

Training Manual

*Data Analysis - Db2
and Oracle*

Preface

This manual describes basic concepts of structure query language (SQL) for data analysis in various RDBMS like Db2 and Oracle.

It explains how to perform database manipulation tasks and demonstrate data analysis tasks through simple examples.

It consists of commands and process to restore database dump and connect two same or different databases for data analysis.

Process of using SQL for data analysis is almost for DB2 and Oracle. Only syntax of the query/functions differs for some of the data analysis conditions in DB2 and Oracle, same has been mentioned in the manual.

Prerequisites

Following database server and client should be installed on the system for data analysis:

- D2 9.7 or DB2 10.5 express C database server on Windows
- Oracle 11g database server on Windows
- IBM data studio client to connect DB2 database server
- SQL developer client to connect Oracle database server

Table of Contents

1.	Getting Started with IBM Data Studio Client.....	8
1.1.	Restore DB2 database using data studio client	8
1.2.	Restore DB2 8.x database dump	12
1.3.	Uninstall DB2 server.....	16
1.4.	Import Oracle database dump	18
1.5.	Using data studio client for data analysis.....	19
2.	Data Export in DB2.....	27
2.1	Export using data studio client.....	27
2.2	Export using DB2 command line (db2cmd)	30
3.	Data IMPORT in DB2	33
3.1	Import using data studio client	33
3.2	IMPORT using DB2 command line (db2cmd).....	37
4.	Data EXPORT in ORACLE	39
4.1	Export using SQL Developer	39
5.	Data IMPORT in ORACLE.....	44
5.1	Import using SQL Developer	44
6.	Data Analysis - DB2/Oracle Database.....	50
6.1	Create tables and views.....	50
6.2	Insert Data	51
6.3	Using SQL to access data	52
6.3.1	Basic Functions	52
6.3.2	Conditional Selection	54
6.3.3	In, Between and Like Clause.....	55
6.3.4	Joins	56
6.3.5	Order By, Group By and Having Clause	58
6.3.6	Union, Except, Intersect Clause	60
6.3.7	Case	61
6.3.8	Sub-query.....	62
6.3.9	Xmlquery.....	64
6.3.10	Index.....	65
7.	Difference in DB2 and Oracle Data Analysis.....	68

7.1	Syntax difference of common commands/functions	68
8.	Connect two database in DB2 for data analysis	70
9.	Connect Oracle database with Db2 database for data analysis.....	82
10.	Troubleshooting in DB2.....	92
11.	General issues on information security in RDBMS.....	94
12.	Appendix - Sample Queries	96

Getting Started with IBM Data Studio Client



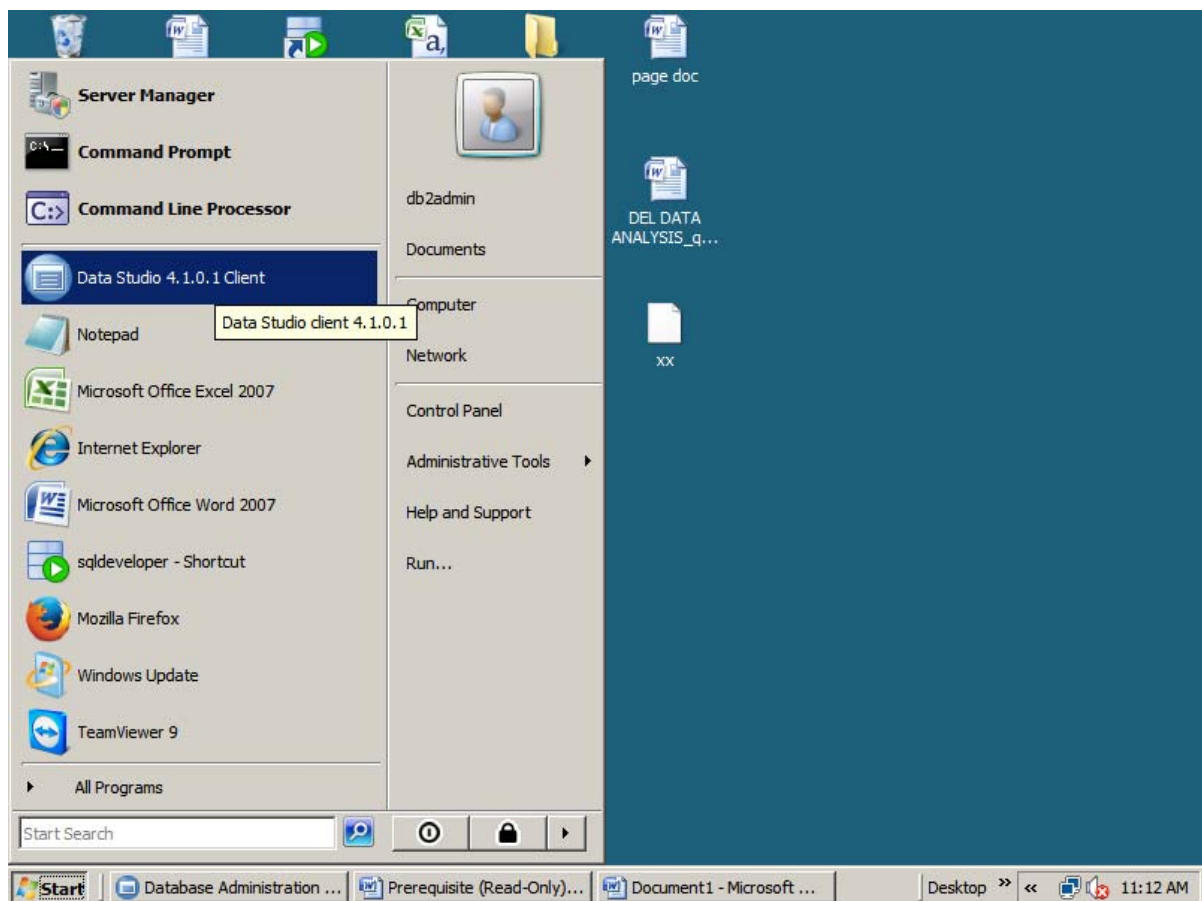
1. *Getting Started with IBM Data Studio Client*

IBM data studio is a client to connect DB2 database server for database administration tasks, and query analysis.

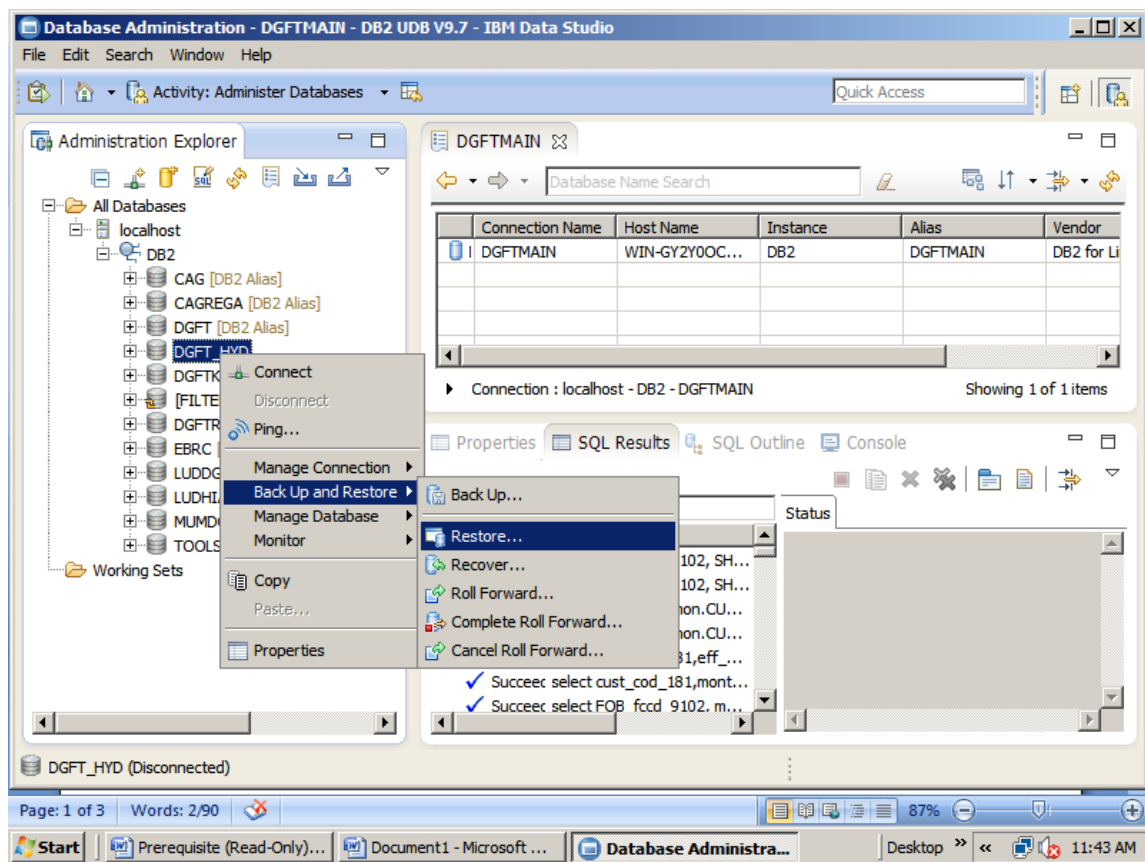
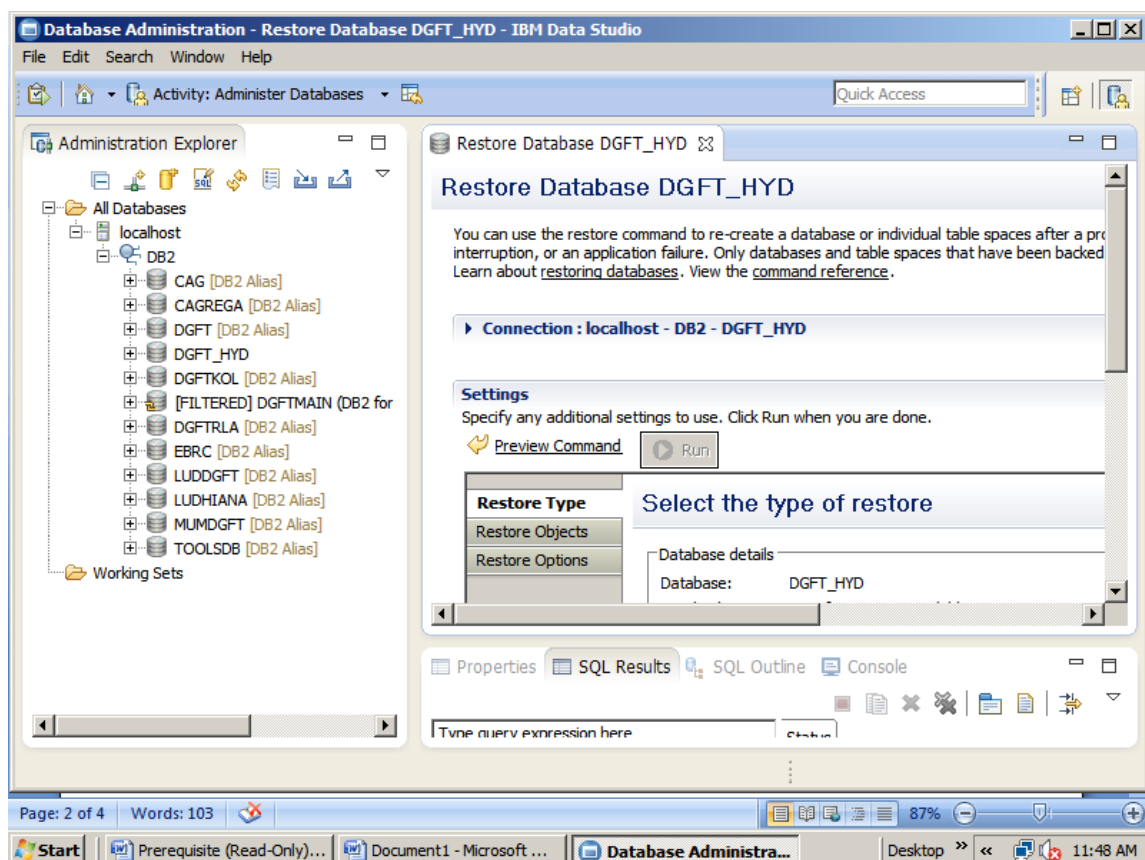
1.1. *Restore DB2 database using data studio client*

Following are the steps to restore a db2 database dump:

- a) Click on start and click data studio 4.1.0.1 client to open the client.

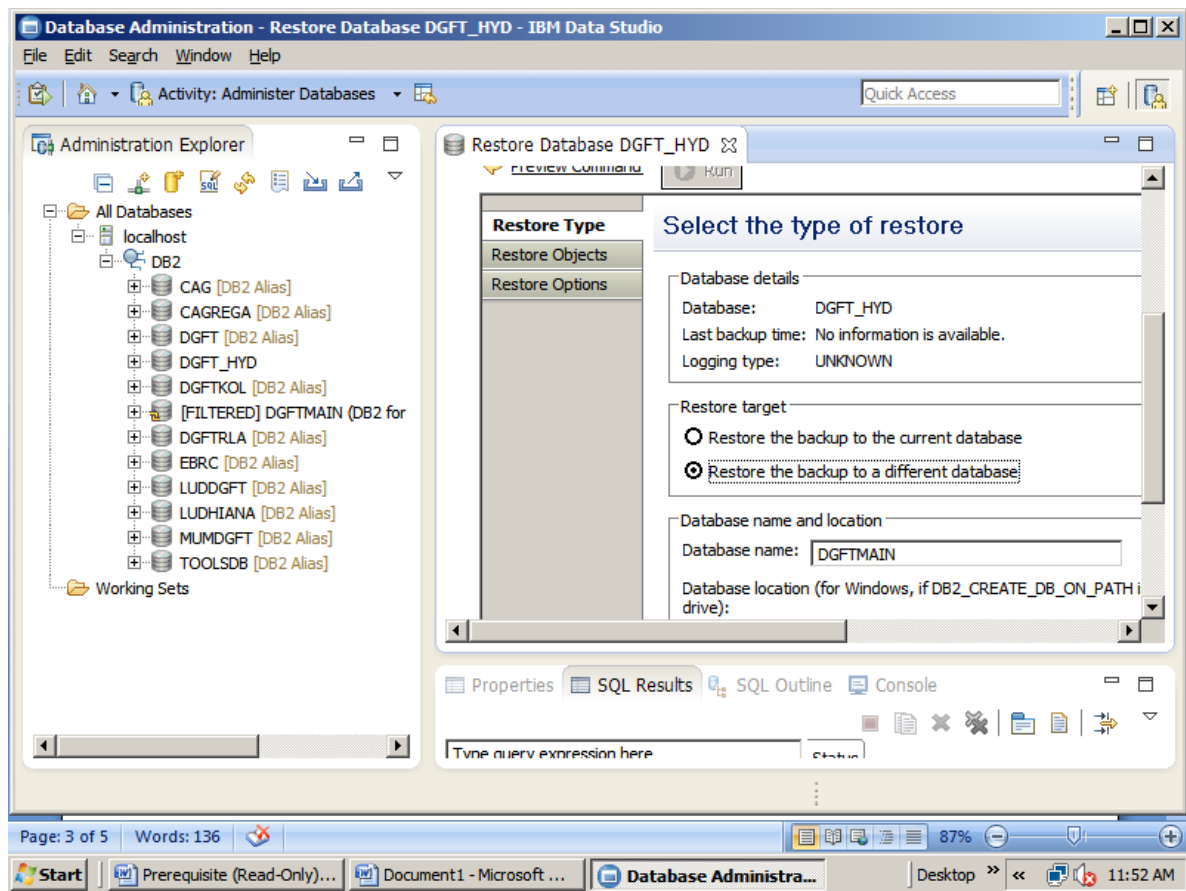


- b) Right click on a database option, click Backup & Restore option and then Restore, restore database window will appear.



- c) Click the **restore type** tab on left side and select option -- restore the backup to a different database and provide database name as name for the database being restored.

Eg. DGFTMAIN if the name of the database being restored is DGFTMAIN.



- d) Click **restore objects** tab on left side, tick mark Backup is from another data base option and provide database name as name for the database being restored.

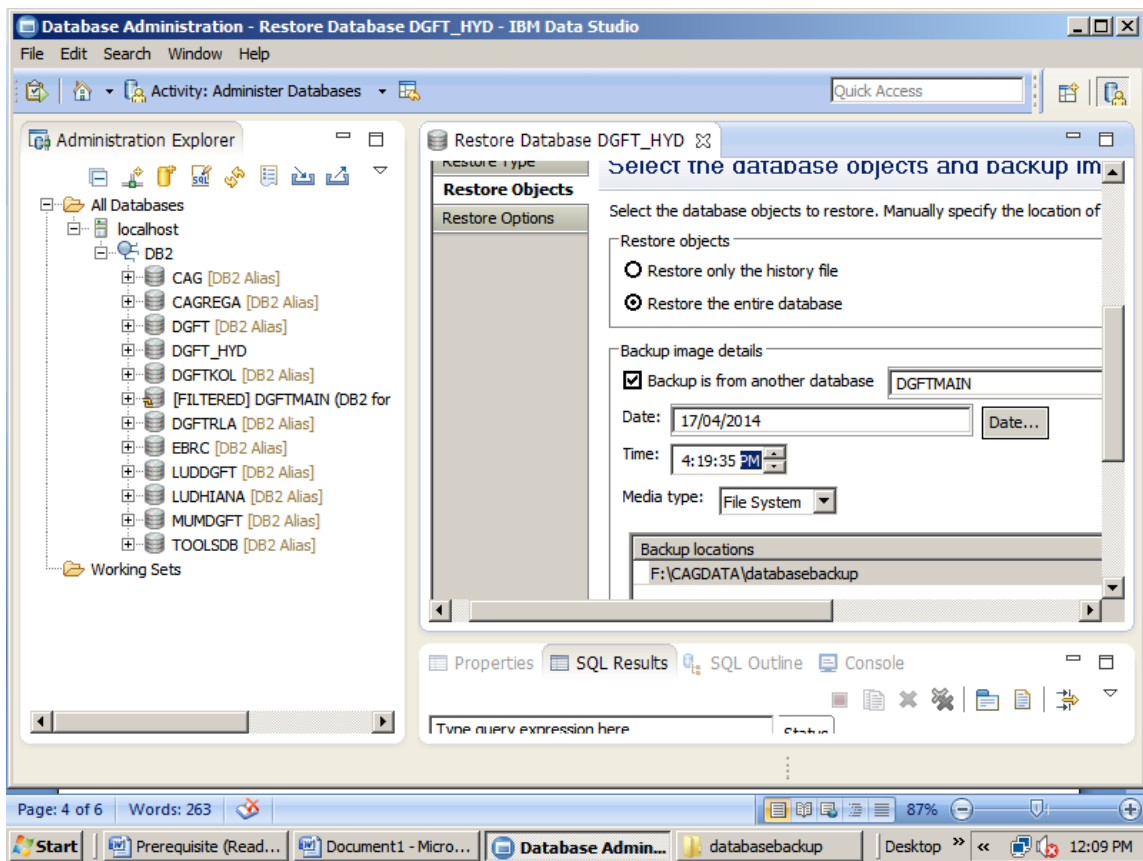
Eg. DGFTMAIN if the name of the database being restored is DGFTMAIN.

- e) Provide **date and time** as mentioned in the database file.

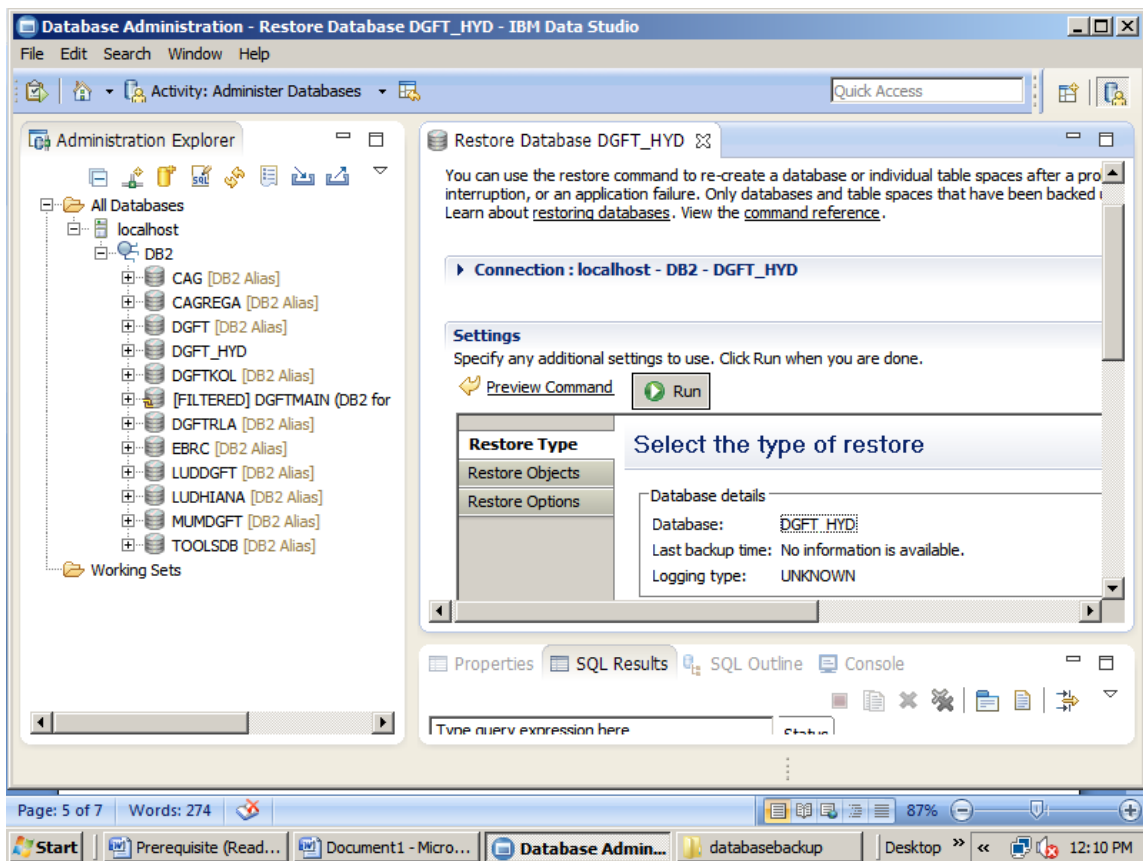
Eg. If the file name of database being restored is "DGFTMAIN.0.DB2.NODE0000.CATN0000.20140417161935.001" then provide date as 20140417 and time as 4:19:35 PM

- f) Select **media type** as file system and provide backup location as the folder/directory location where database dump file is kept on the system.

Eg. If dump file is kept at location " F:\CAGDATA\databasebackup", provide F:\CAGDATA\databasebackup as backup location by clicking the add/browse button.



g) Click **Run** button to start and restore the database.



Note: Name of DB2 database dump file should be like:

< database name>.0.DB2.NODE0000.CATN0000.20140717174509.001

Eg. EBRC.0.DB2.NODE0000.CATN0000.20140717174509.001

1.2. Restore DB2 8.x database dump

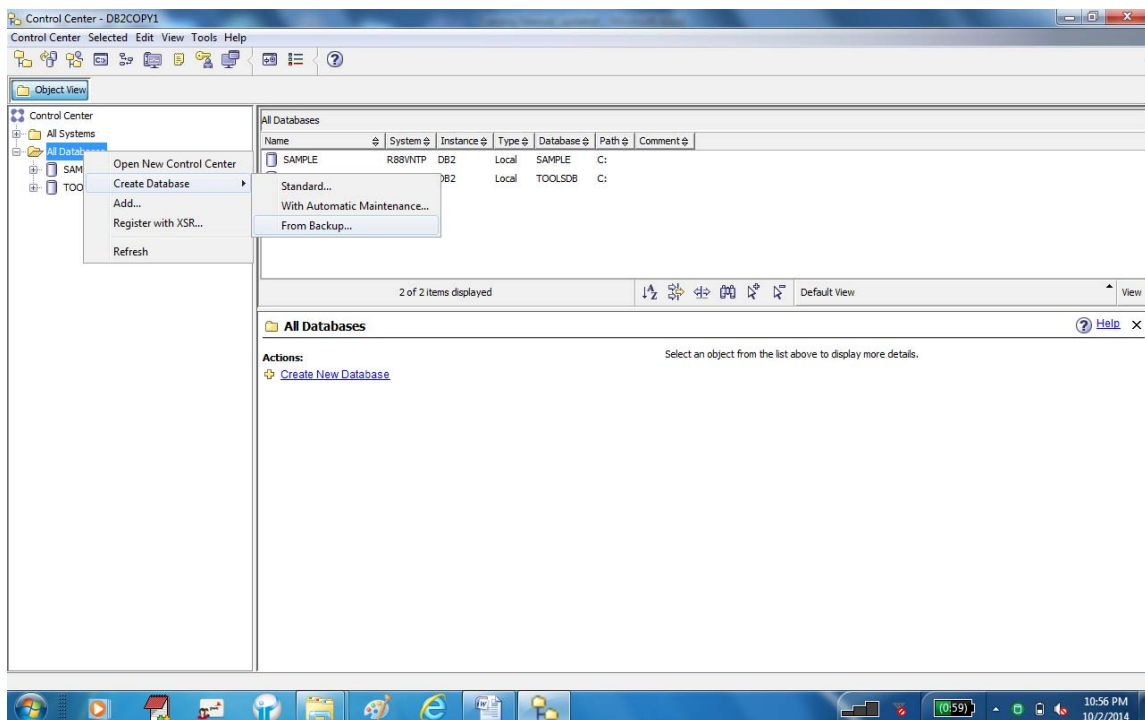
In case the DB2 database dump is from an older version (8.x) of Db2 database, we need to restore the dump first in 9.7 DB2 database server and then from 9.7 to 10.5 database server. Following steps are followed:

- a) Make a directory tree structure as mentioned below and save the database dump file in the last folder (date)

C:\DGFT.0\DB2\NODE0000\CATN0000\20141002

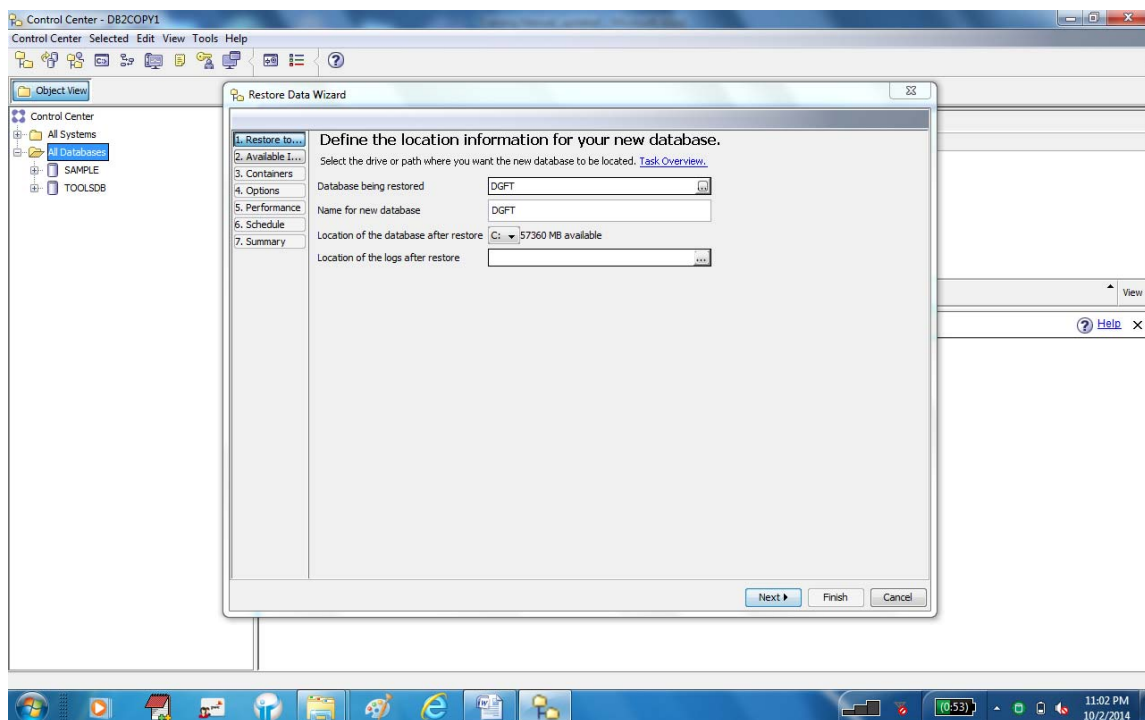
Where DGFT is the name of the database dump to be restored and 20141002 is the date when the backup was taken.

- b) Click on start and type db2cc in search/run to open the DB2 control centre.
- c) Right click on All database option, click create database option and then From backup, restore data wizard will appear.



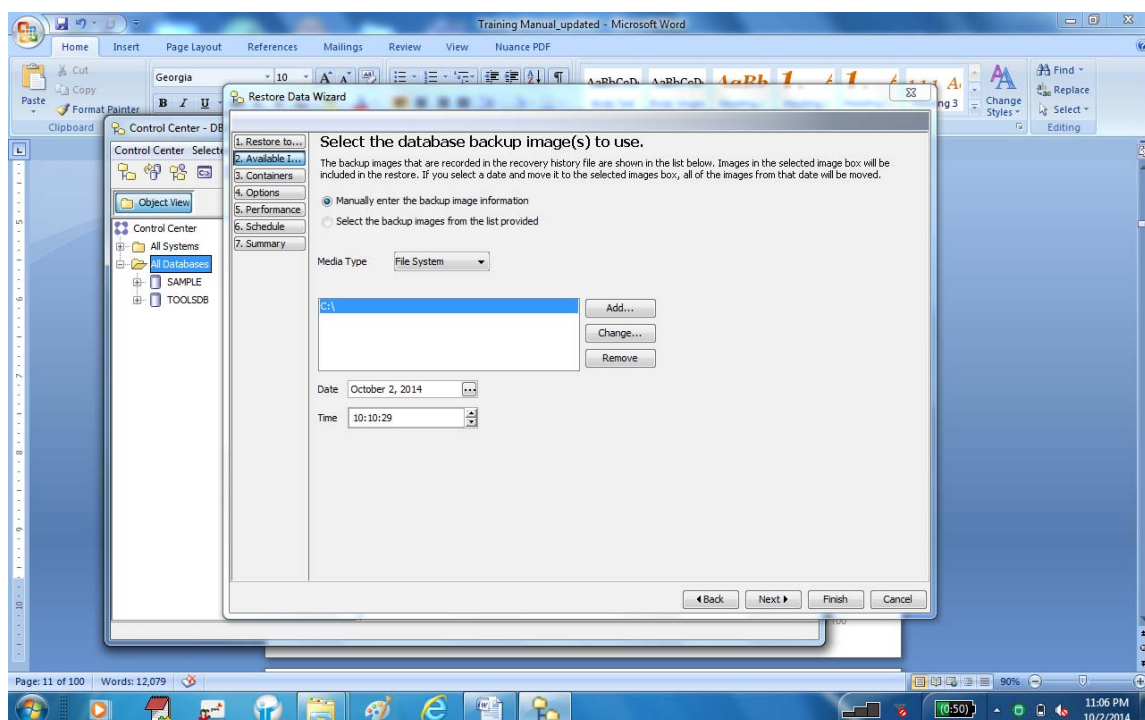
- d) Type the name of database being restored and name of the new database. Click next

Eg. DGFT if the name of the database being restored with database name as DGFT.

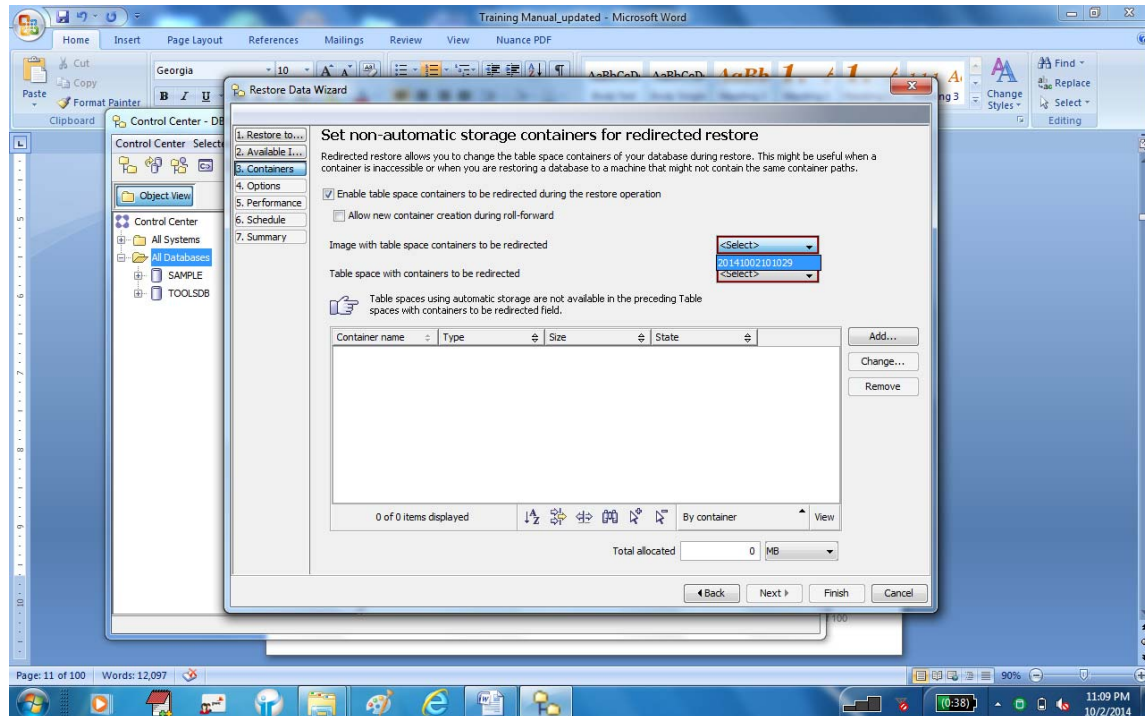


- e) Add the name of the drive in which we created the tree structure and saved the dump file, provide the date and timestamp of dump to be restored. Click next

Eg. C:\ drive is the drive in which we created tree structure mentioned above, 20141002 is the date folder and 10:10:29 is the timestamp mentioned in dump filename.



- f) In case we want to change the location of containers click enable table space container option and select timestamp and table space to change the location .

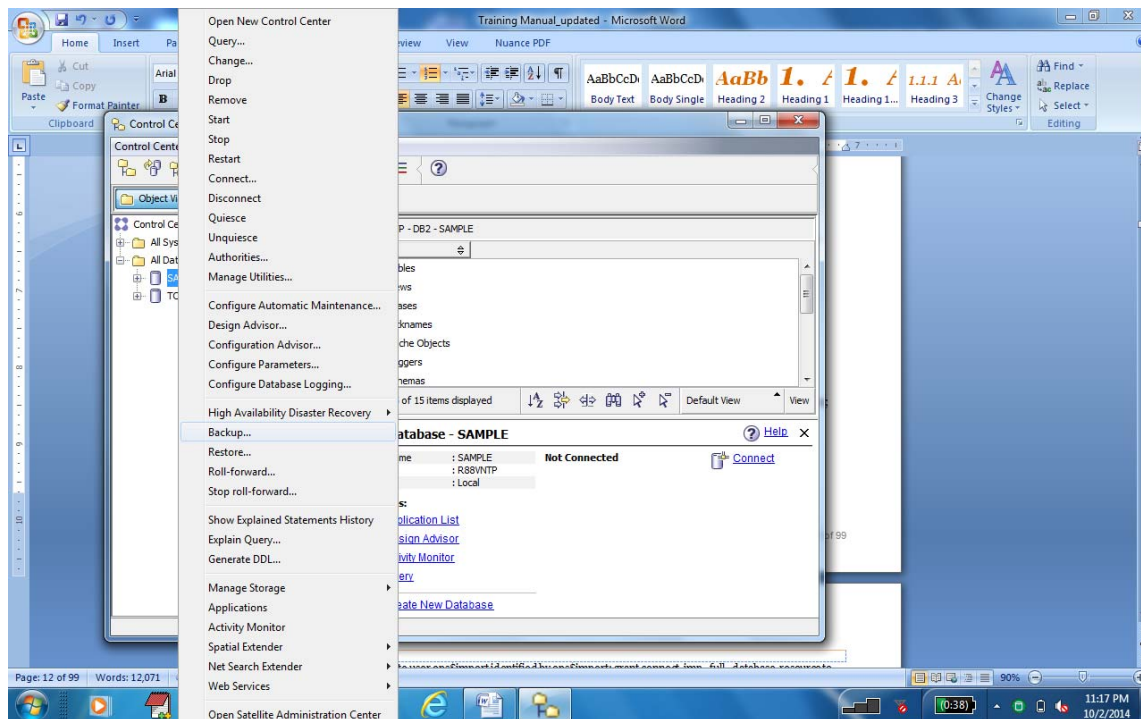
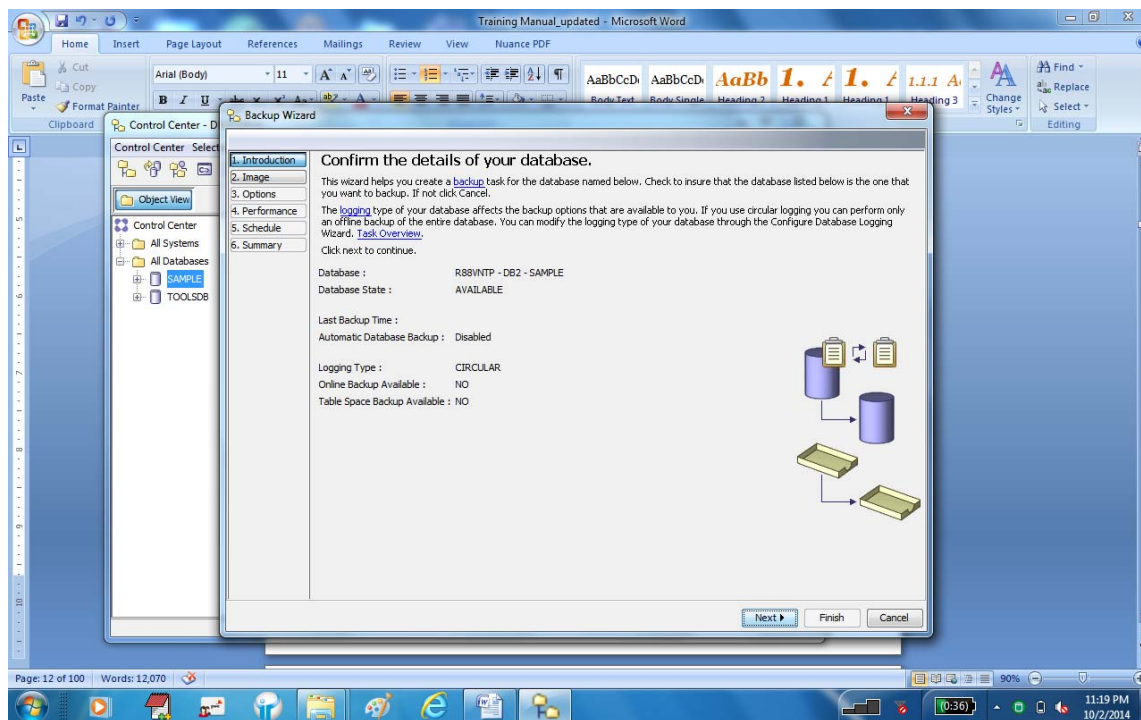


- g) Click **Finish** button to start and restore the database.

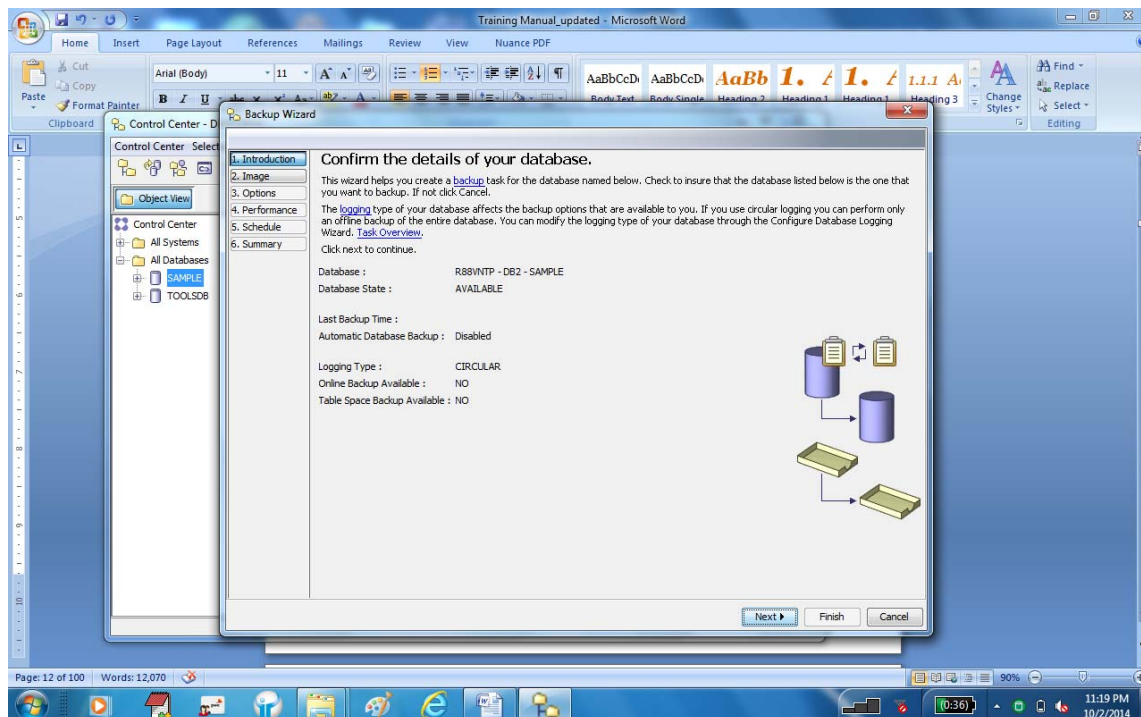
Note: Name of DB2 database dump file should be like:

101029.001 where 101029 is the timestamp of the database dup file.

- h) Once the database restore is complete, take the backup of the database restored by right click the database and click backup option.



- h) Click next and provide the location where we want to take the database backup, click next and then finish.

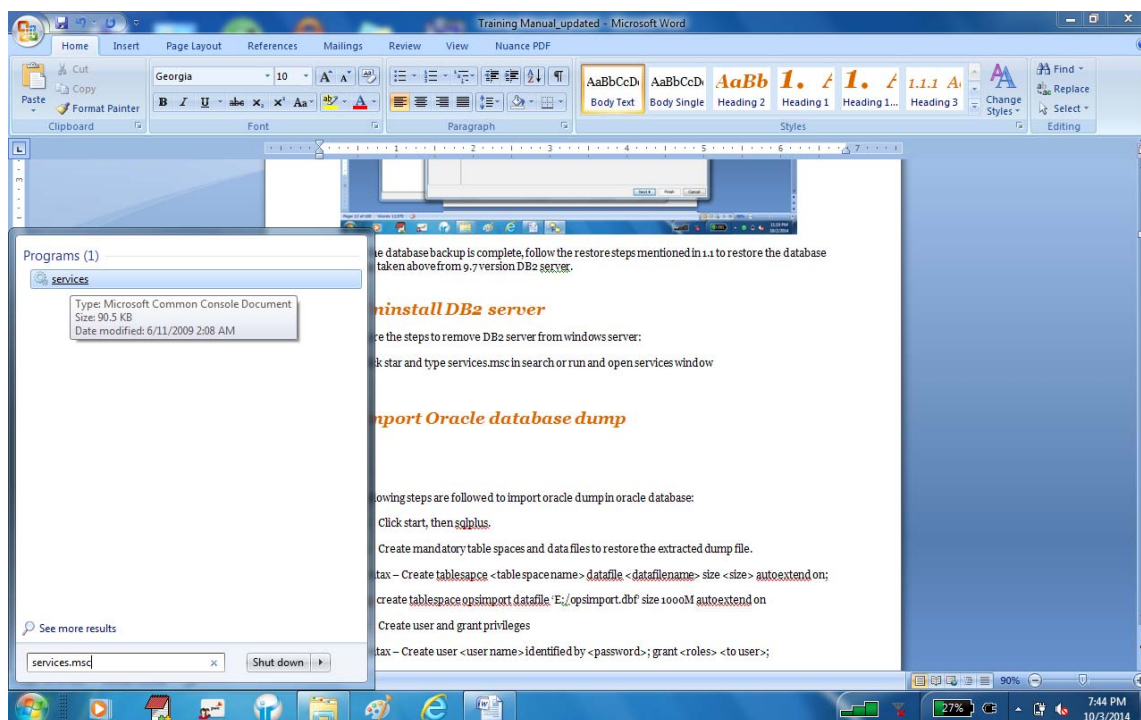


Once the database backup is complete, follow the restore steps mentioned in 1.1 to restore the database backup taken above from 9.7 version DB2 server.

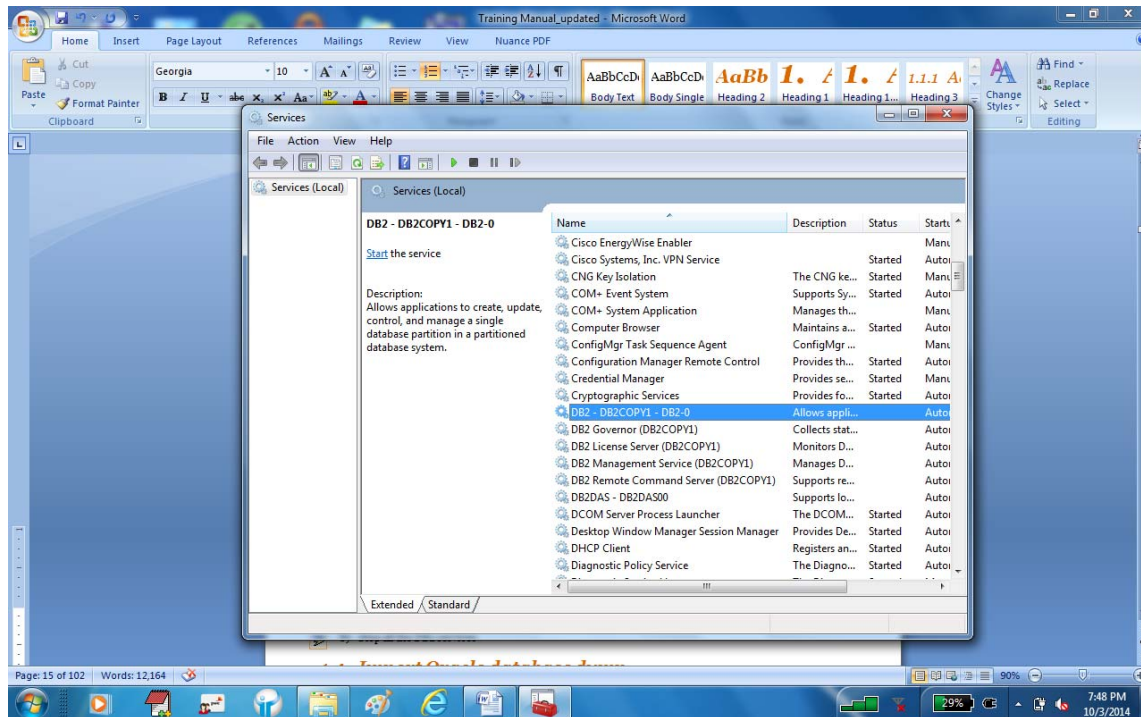
1.3. Uninstall DB2 server

Following are the steps to remove DB2 server from windows server:

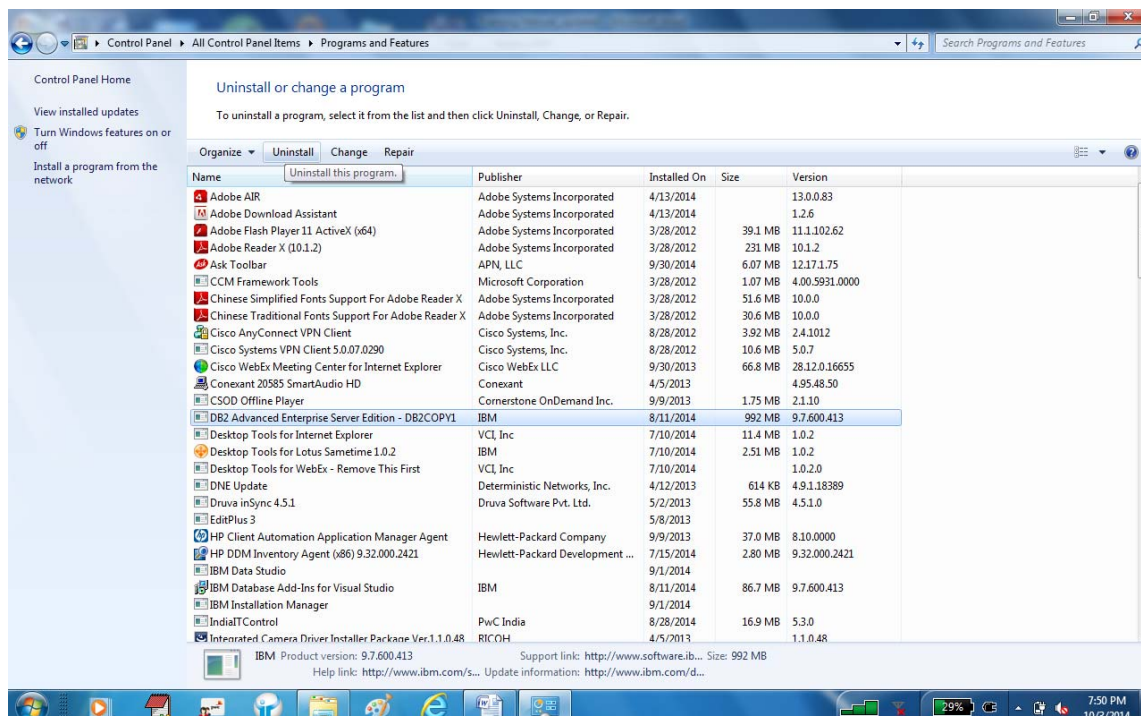
- Click star and type services.msc in search or run and open services window.



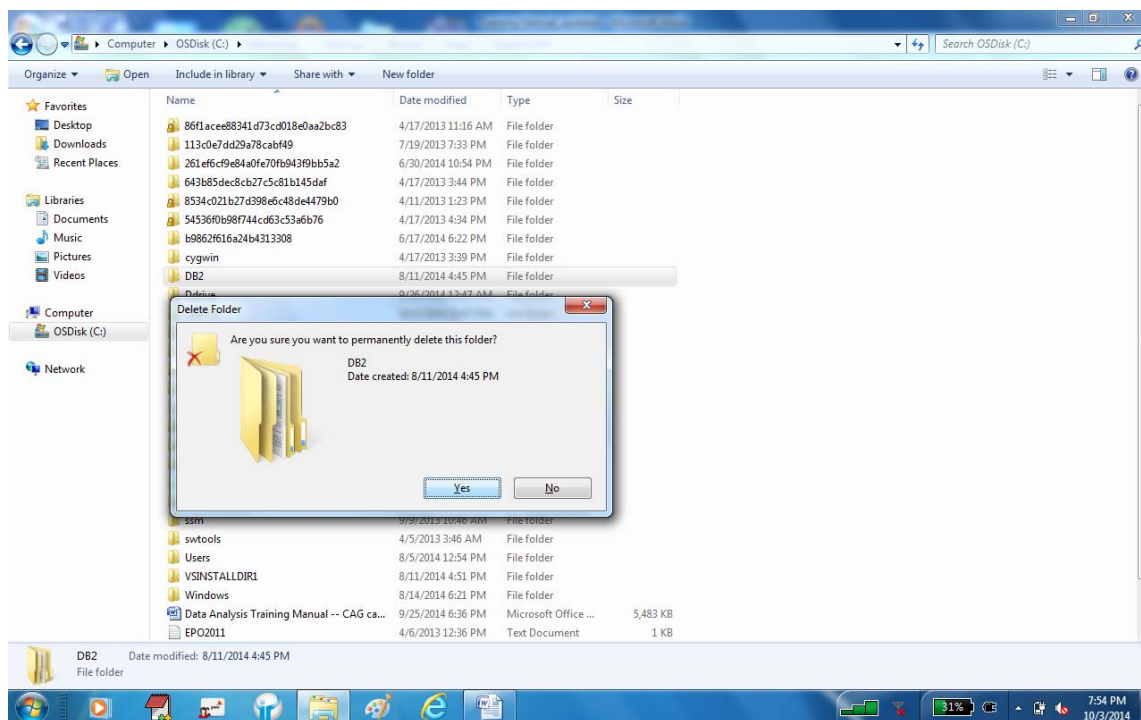
b) Stop all the services starting with name DB2.



c) Open control panel, click program and features and uninstall DB2 server.



d) After uninstall the DB2 server, delete all the files and folders which were created at the time of installation. Folder would either start from IBM or DB2 in any of the drives where you installed DB2 server. Post deletion Db2 server will be removed from the windows machine.



1.4. Import Oracle database dump

Following steps are followed to import oracle dump in oracle database:

- a) Click start, then sqlplus.
- b) Create mandatory table spaces and data files to restore the extracted dump file.

Syntax – Create tablespace <table space name> datafile <datafilename> size <size> autoextend on;

Eg: create tablespace opsimport datafile 'E:/opsimport.dbf' size 1000M autoextend on

- c) Create user and grant privileges

Syntax – Create user <user name> identified by <password>; grant <roles> <to user>;

Eg: create user ops\$import identified by ops\$import; grant connect, imp_full_database, resource to ops\$import.

- d) Create directory for storing the file from where import will happen

Syntax – create directory <directory name> as <path name>;

Eg: create directory import as 'E:\import'

- e) Grant privileges on the directory to user

Syntax – grant read on, write on directory <directory name> to <user>

Eg: grant read on, write on directory import to ops\$import

- f) Go to command prompt by clicking start, then run and type cmd.

g) Type impdp command to import the dump.

Syntax -- IMPDP <username>/<password> DIRECTORY=<DIRECTORY NAME> DUMPFILE
=<filename.DMP> LOGFILE=<filename.LOG> FULL=Y;

Eg: impdp ops\$import/ops\$import Directory=import DUMPFILE=impcga_010213310313.dmp
LOGFILE=import.log FULL=Y;

DEFAULT DIRECTORY:

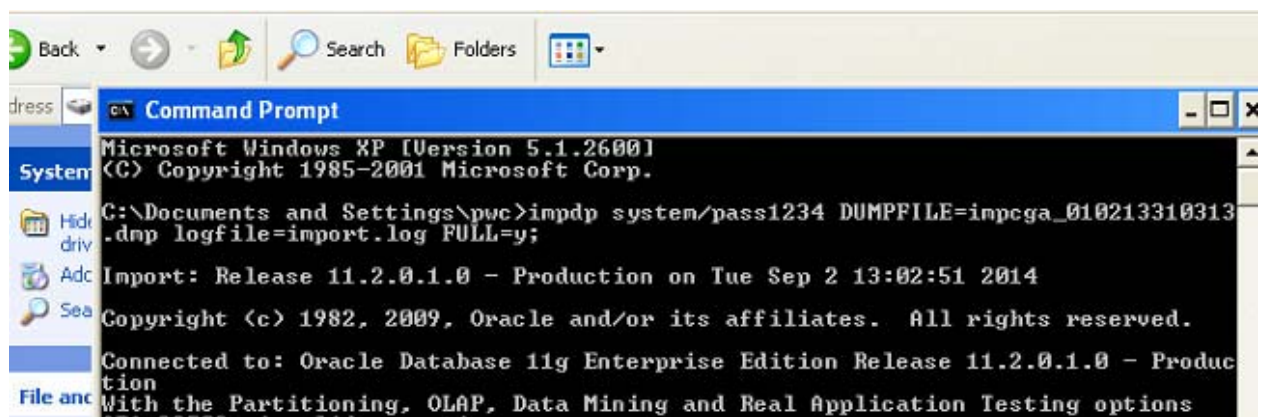
In case you do not want to create a directory, make sure your dump file as well as the log file are in the default directory which is specified below. You can skip the directory parameter in the import command in this case.

Default directory: "C:\app\pwc\admin\orcl\dpdump\impcga_010213310313.dmp"

Where "c:\app\pwc\" is the path where oracle is installed".

Import command when dumpfile is in default directory –

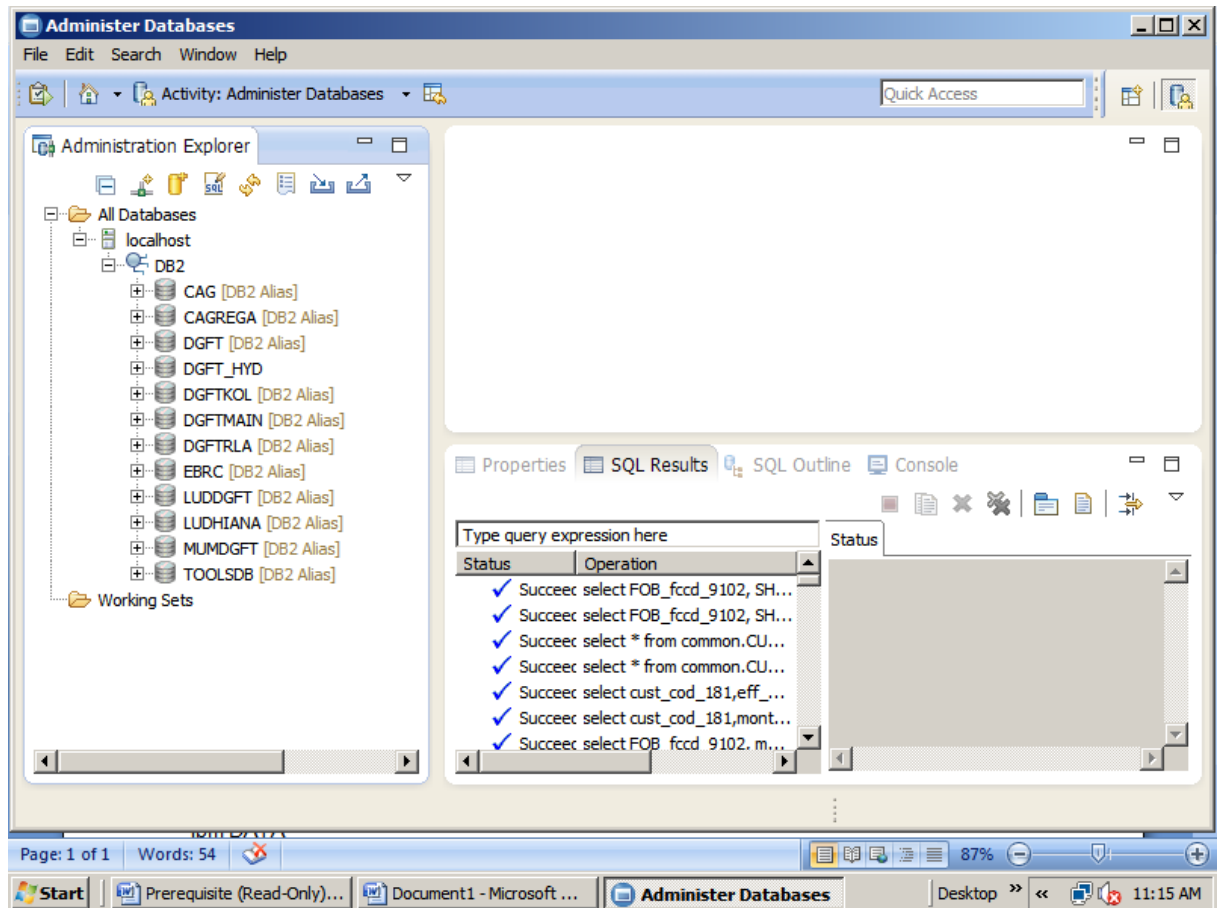
Impdp system/pass1234 DUMPFILE=impcga_010213310313.dmp logfile=import.log FULL=Y;



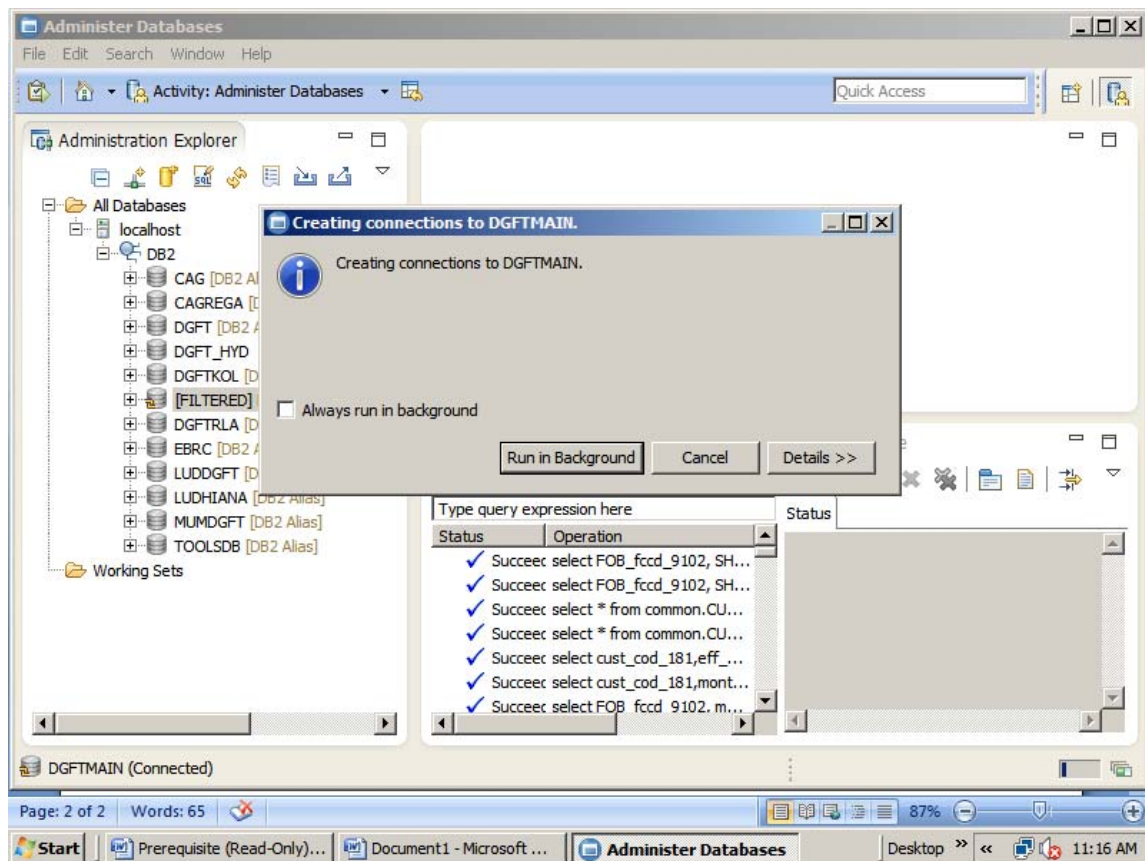
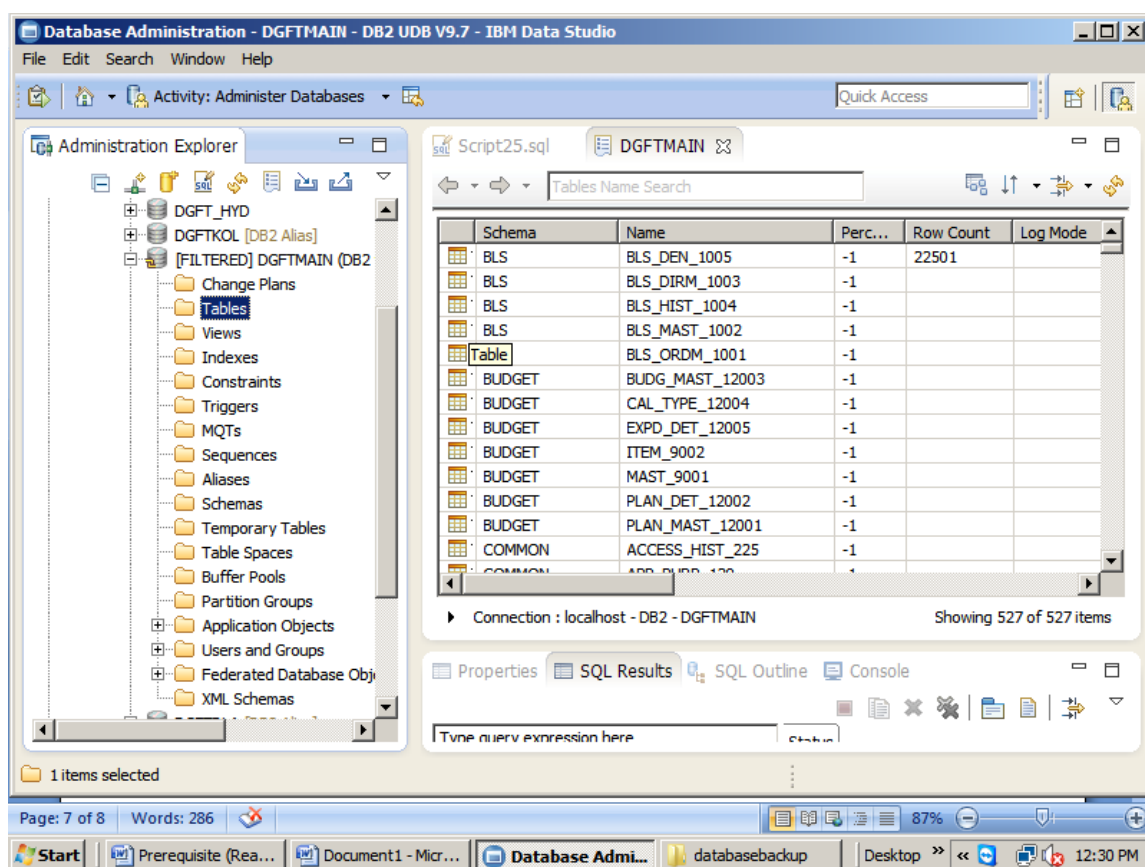
1.5. Using data studio client for data analysis

IBM data studio is a client which can be used for connecting db2 databases and performs data analysis. Basic functionality is mentioned below.

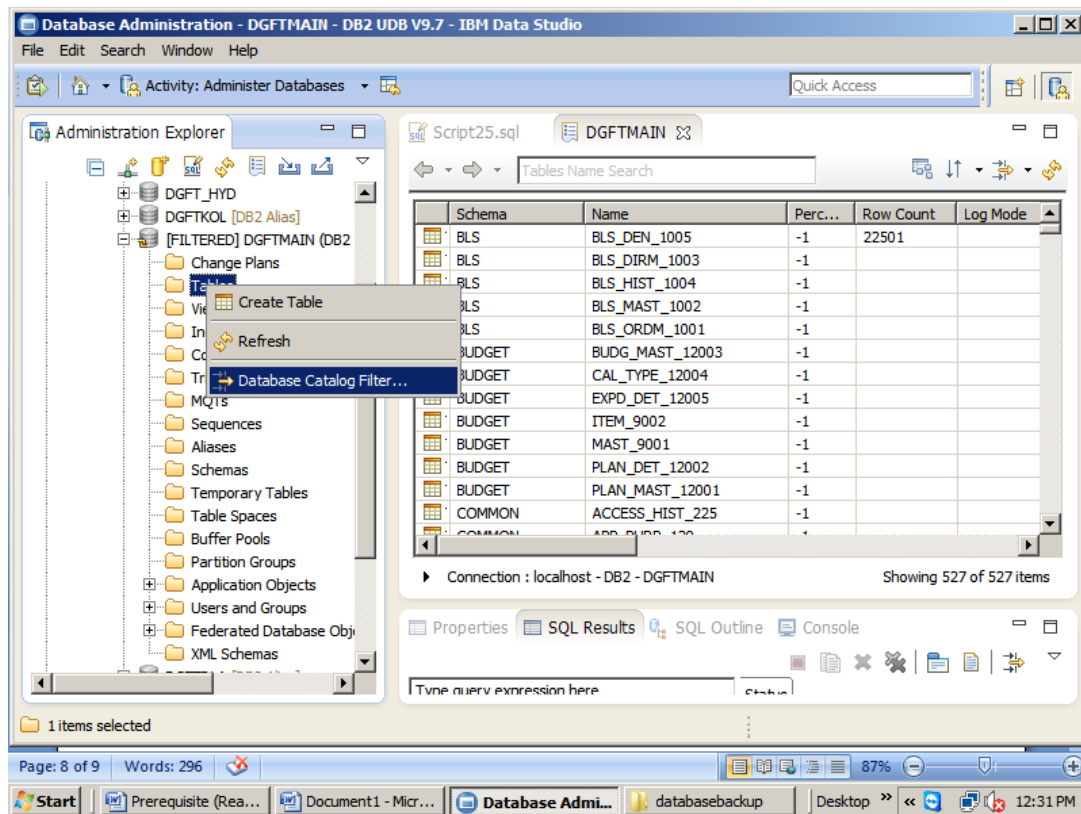
- Open the data studio client and on the right hand side we can all the db2 databases.



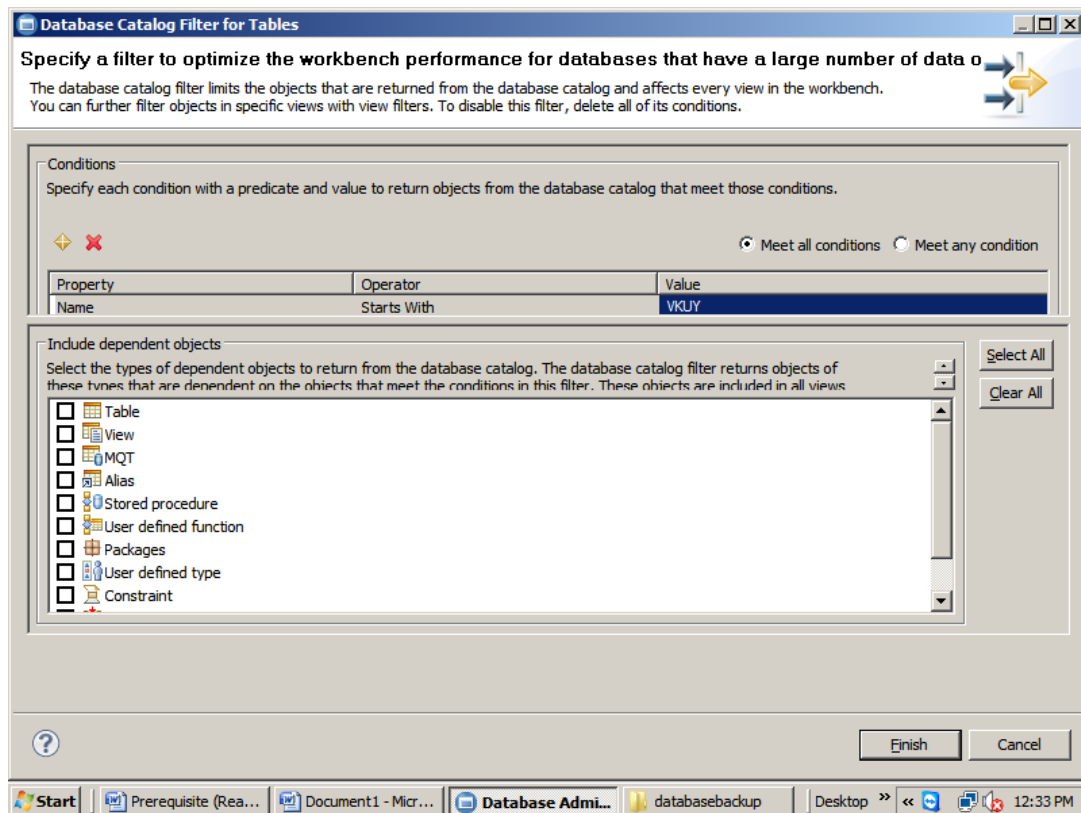
- Double click on any database to connect to a database, click on tables to see list of available tables in the database.



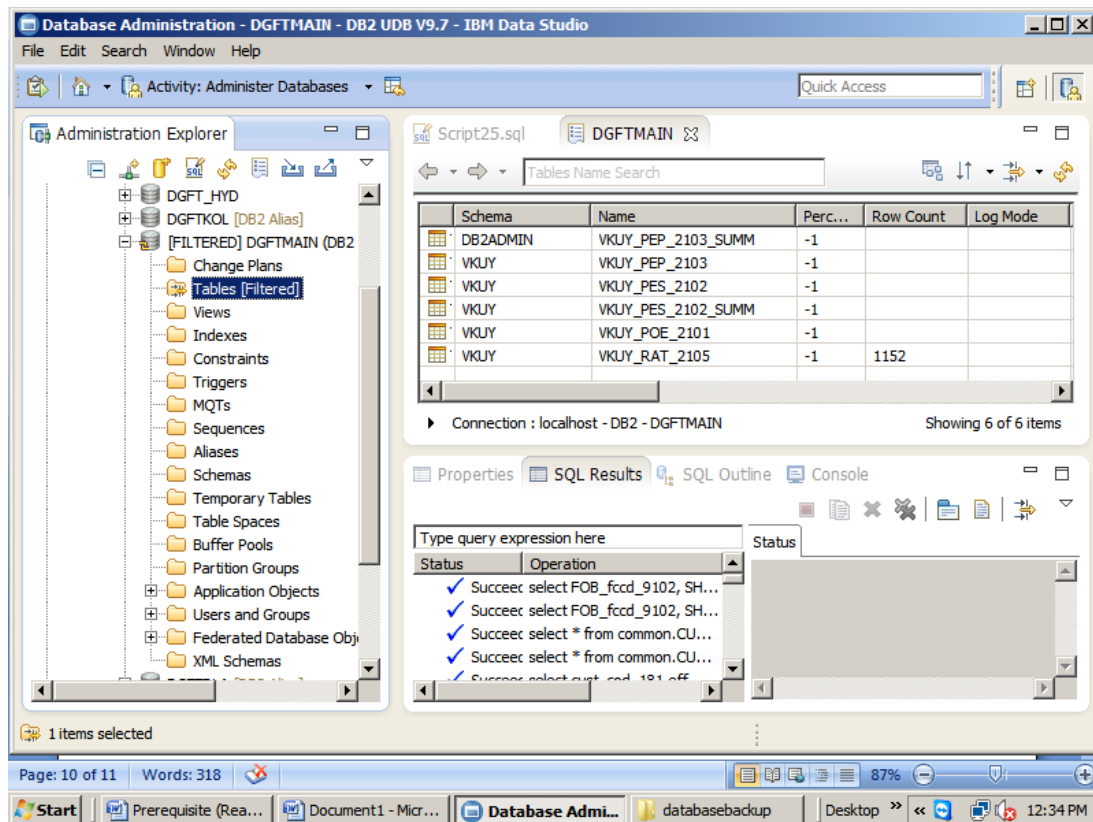
- To see selected list of tables -- Right click tables' option to put a filter and see specific tables.



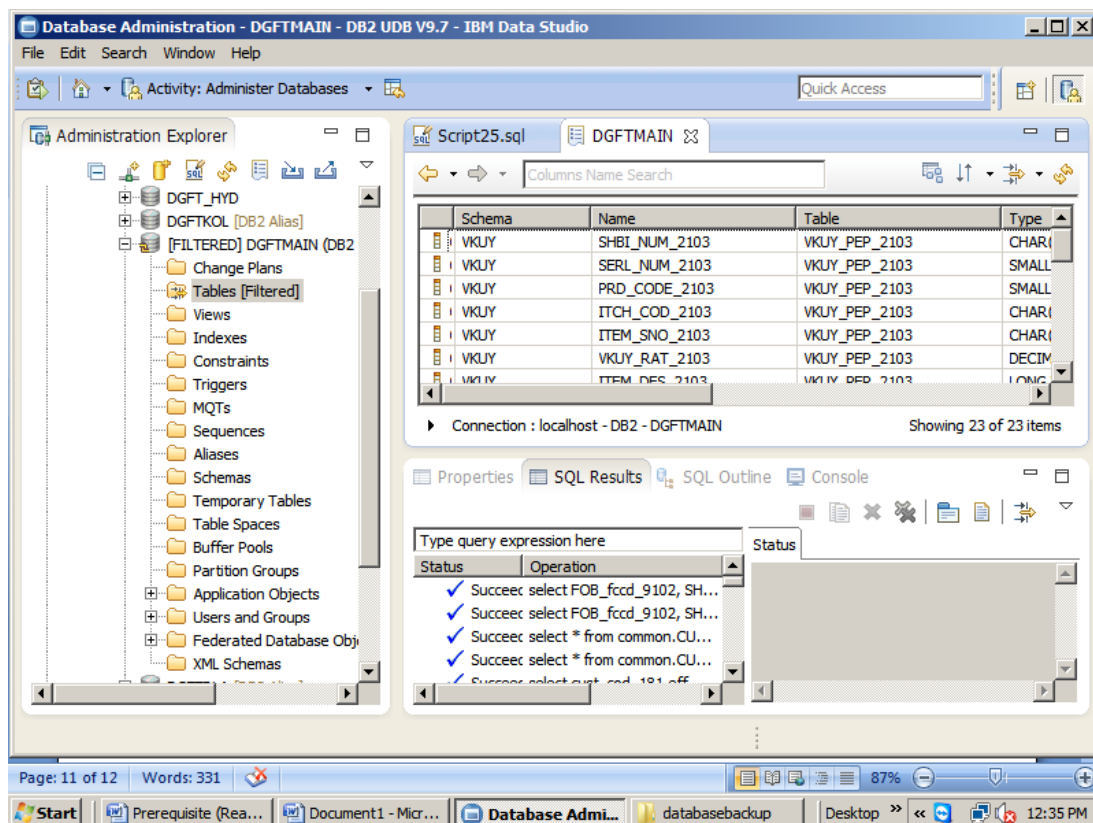
Eg. If we want to see list of tables starting with VKUY



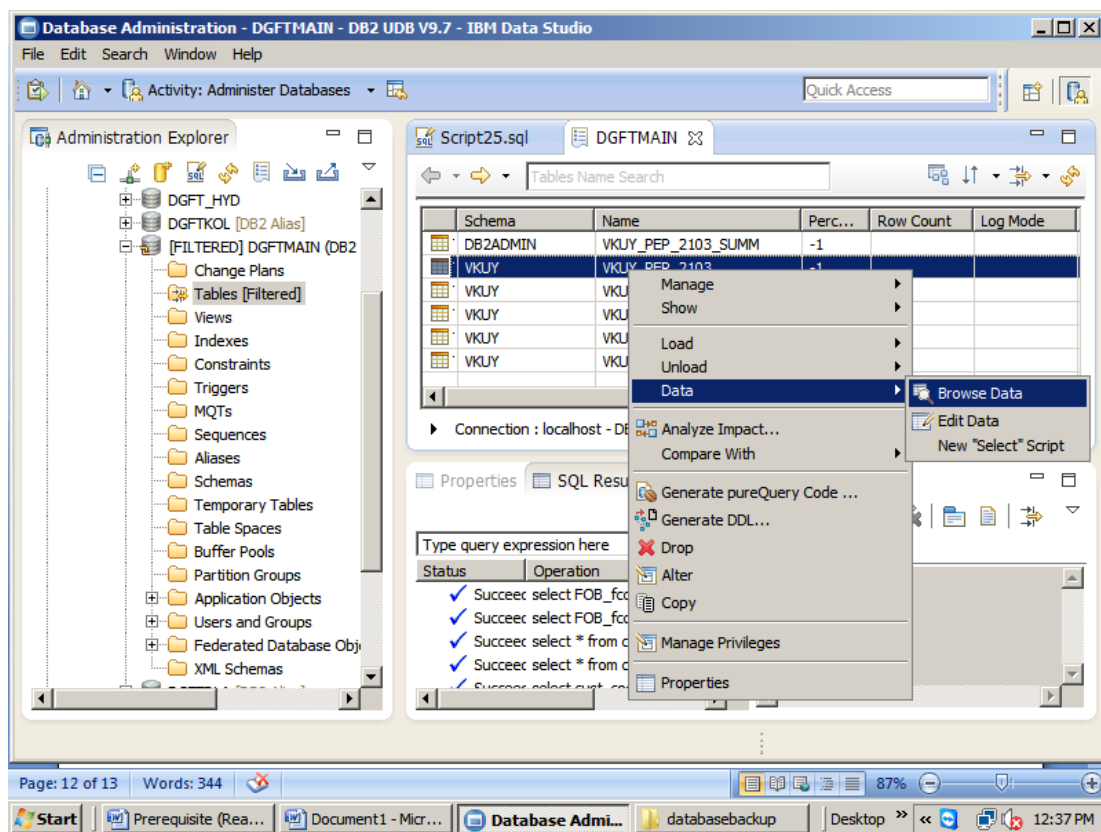
- Click finish to see tables starting with VKUY.



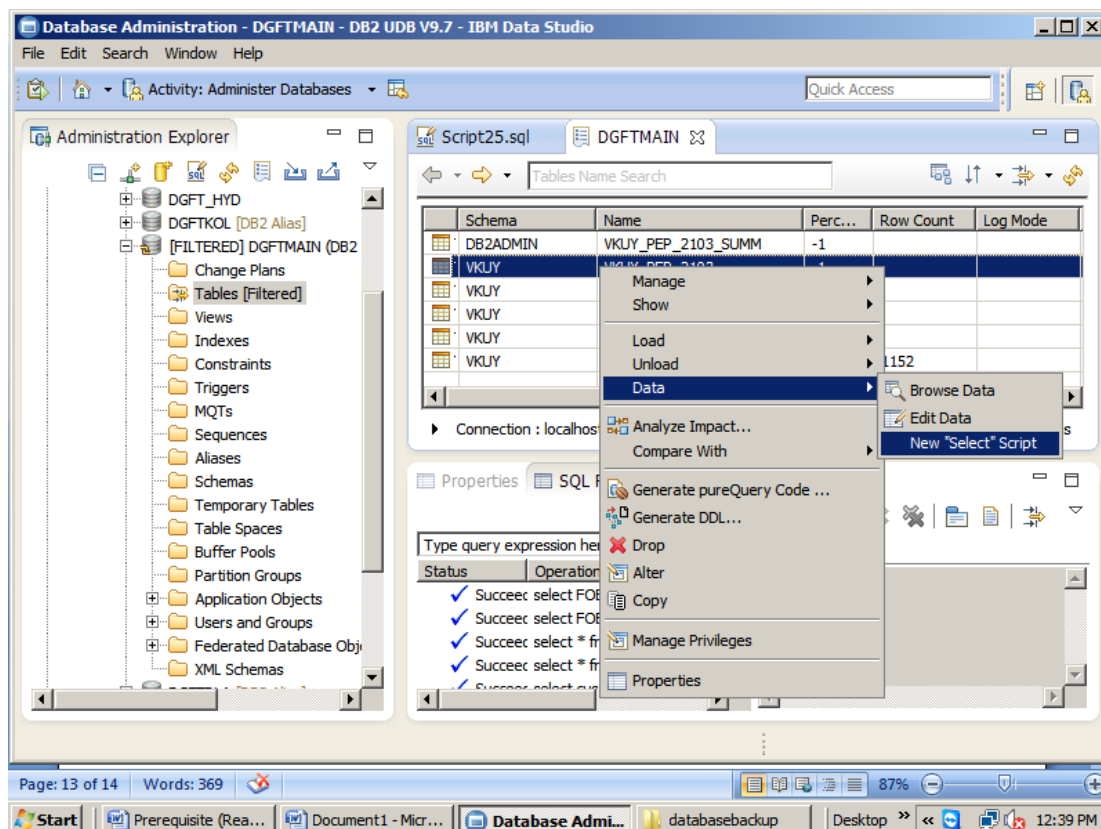
- List of available columns in table -- By double clicking a table, we can see the columns of the table.

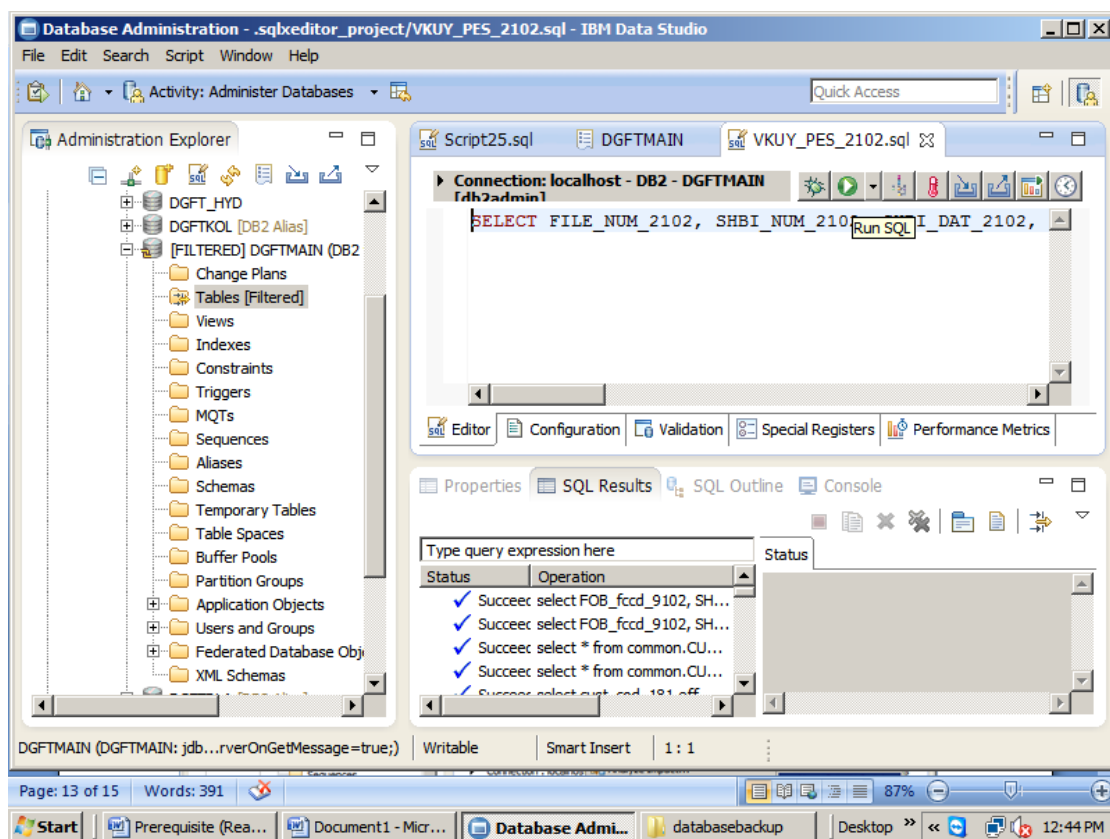


- To see sample data in a table -- Right click on table, click data and then **Browse data** option.



- To run a query using a table -- Right click on table, click **data** and the **new script** option to run a query in sql query window.





Data Export in DB2

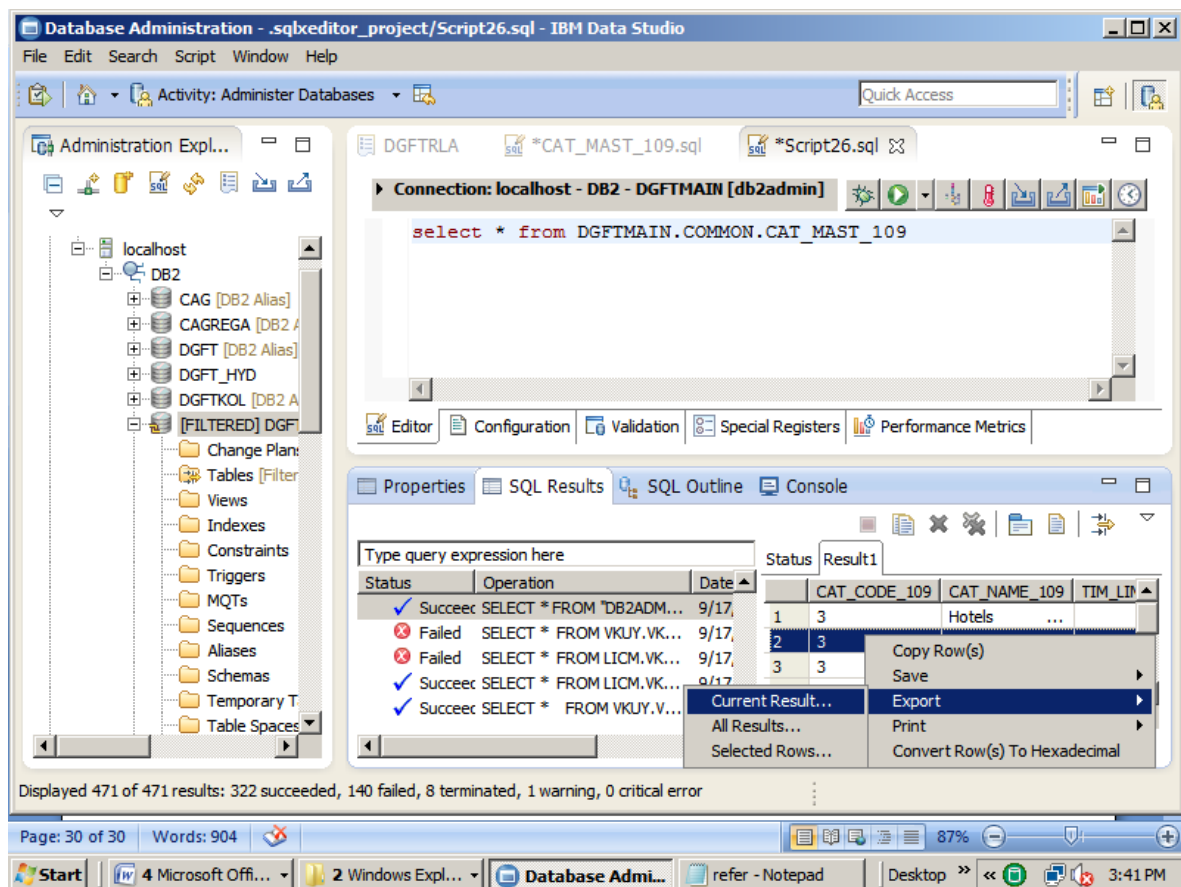


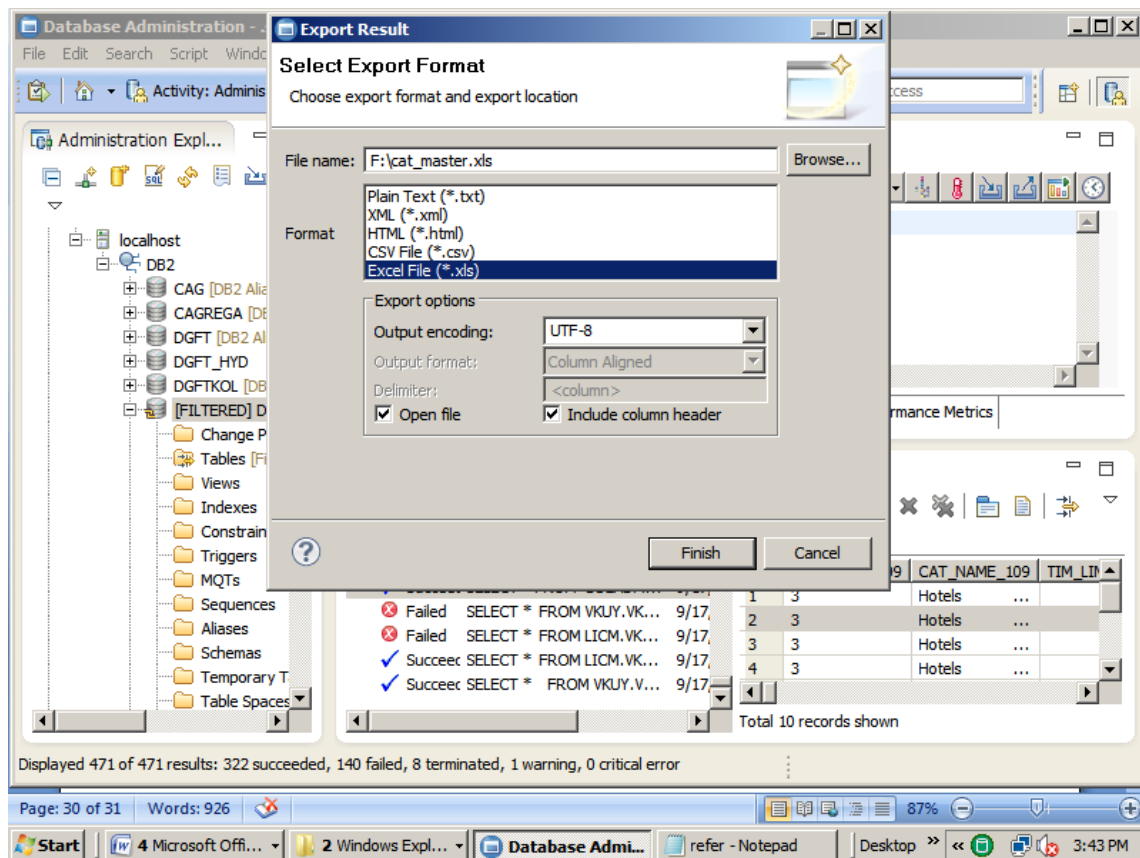
2. Data Export in DB2

We can export data for a table or output of a query in xls, csv, txt format using data studio. There are two ways to export the data.

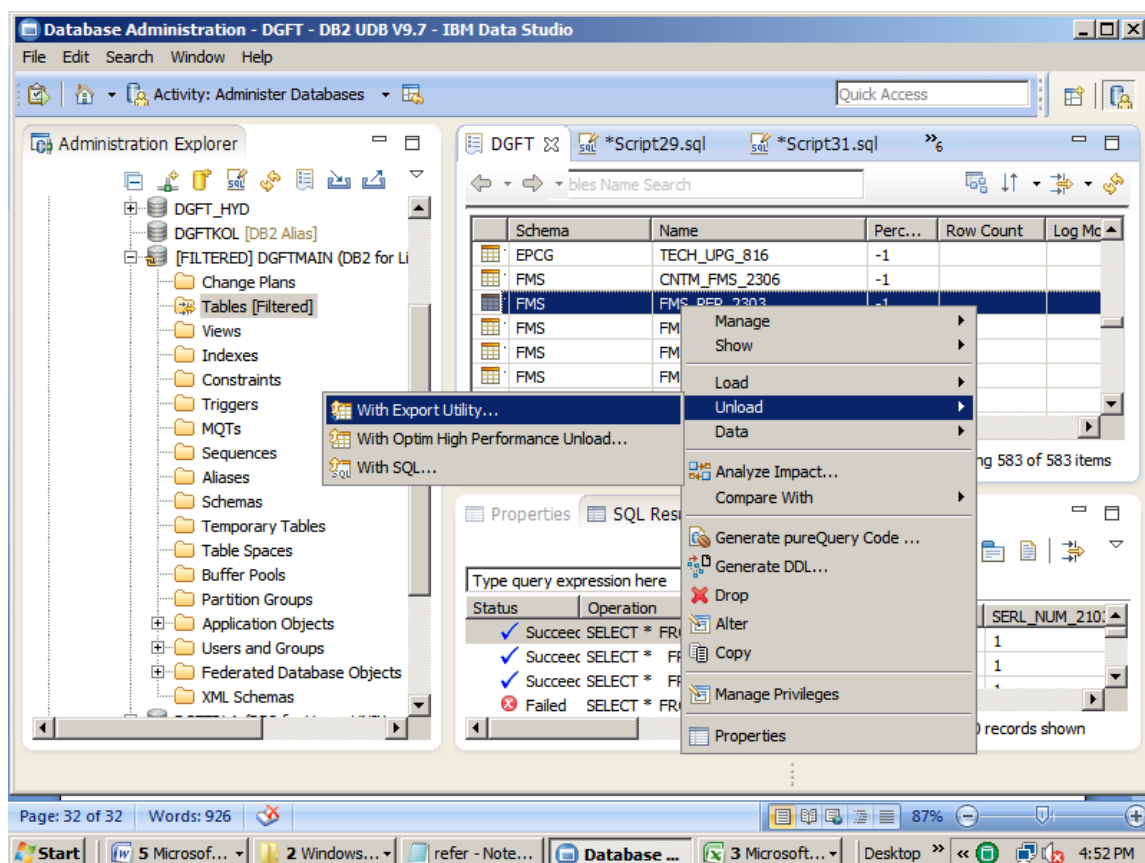
2.1 Export using data studio client

- a) Execute the query, right click on any row of the result displayed then click export and then current results to export the results of query.

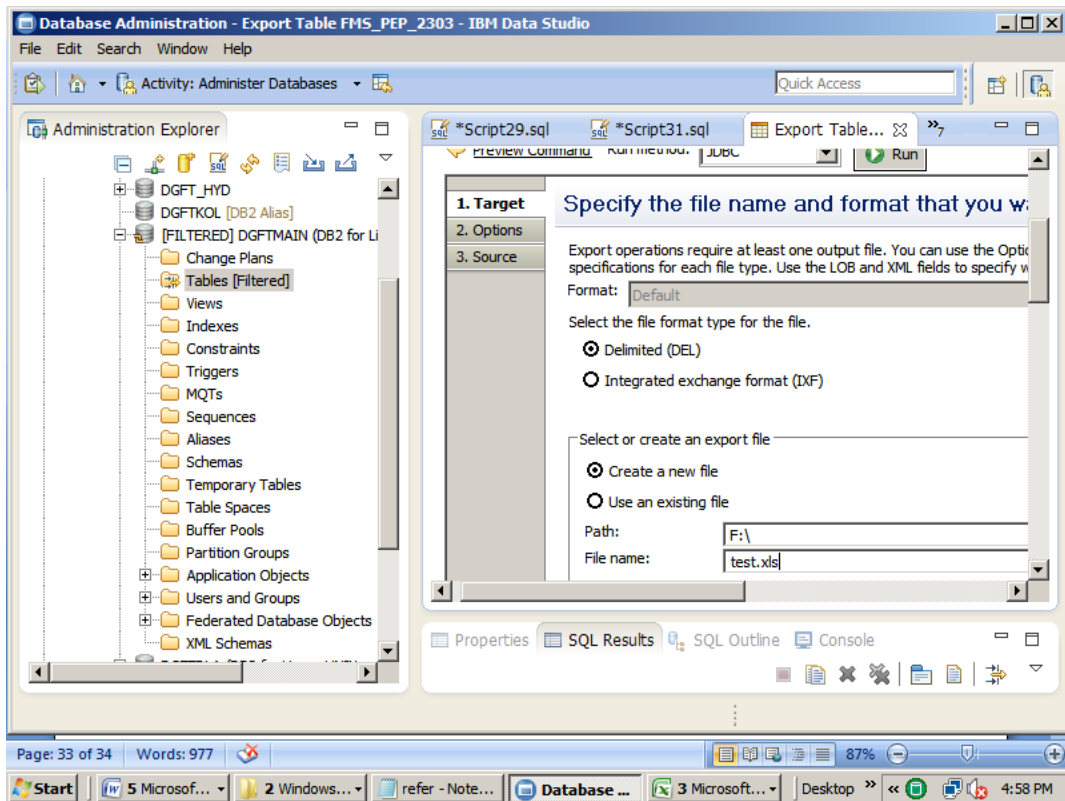




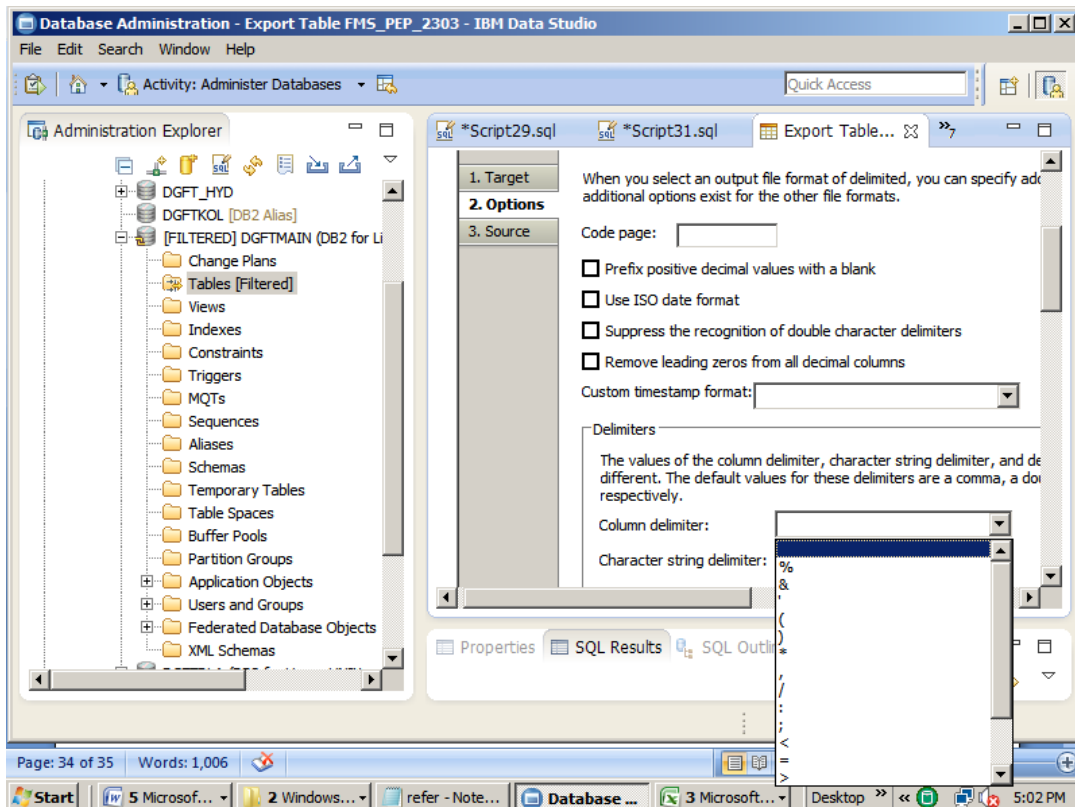
- b) Right click on a selected table, click unload and then click export utility, it will open the export utility window.



- i. Click target tab on right side, select the format as delimited (del), and provide the path and filename in which you want to export the data of the table.

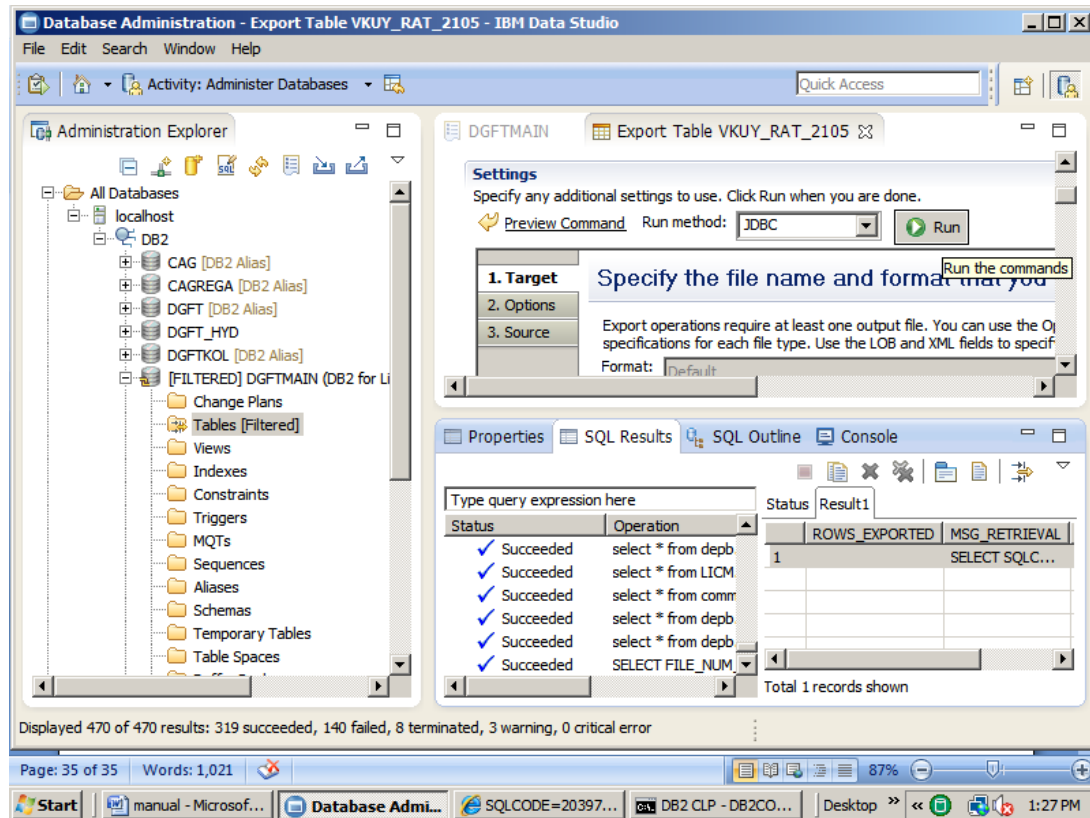


- ii. We can also use option tab to change the column and string delimiters, by default it will take comma and double quotation as column and string delimiter respectively.



- iii. Click run button with method as JDBC after giving filename and path to export the data from the table

Note: Export done through this method extract data in csv format and columns headers are not exported



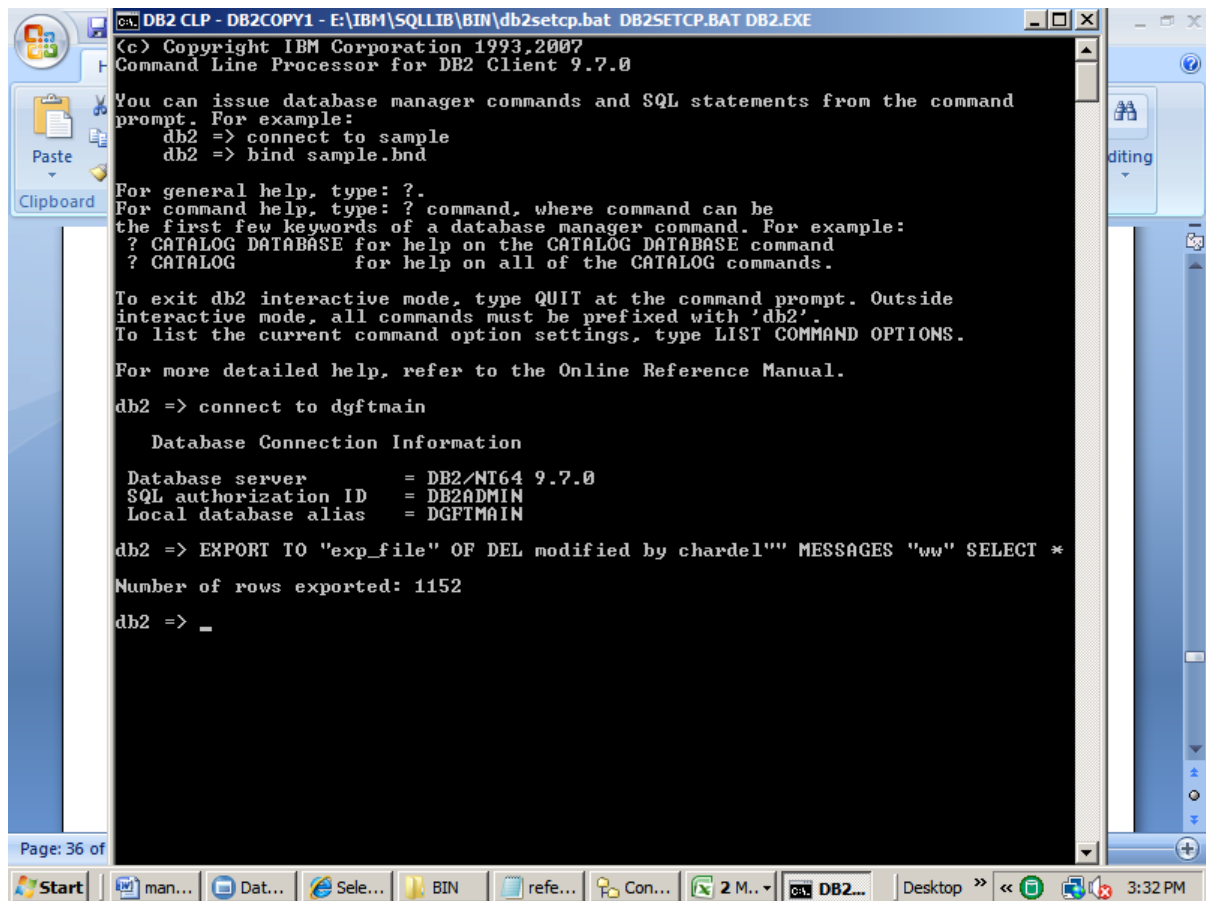
2.2 Export using DB2 command line (db2cmd)

Export command -- db2 EXPORT TO "<filename>" modified by codel<column delimiter> chardel<string delimiter> OF DEL <query whose o/p needs to be exported>

By default column delimiter is , and string delimiter is ""

Eg. db2 EXPORT TO "exp_file" OF DEL MESSAGES "ww" SELECT * FROM VKUY.VKUY_RAT_2105;

Export file will be saved at "C:\IBM\SQLLIB\BIN"



```
c:\> DB2 CLP - DB2COPY1 - E:\IBM\SQLLIB\BIN\db2setcp.bat DB2SETCP.BAT DB2.EXE
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
    ? CATALOG DATABASE for help on the CATALOG DATABASE command
    ? CATALOG           for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftmain

      Database Connection Information

Database server      = DB2/NT64 9.7.0
SQL authorization ID = DB2ADMIN
Local database alias = DGFTMAIN

db2 => EXPORT TO "exp_file" OF DEL modified by chardel"" MESSAGES "vw" SELECT *
Number of rows exported: 1152

db2 => _
```

Page: 36 of

Start | man... | Dat... | Sele... | BIN | refe... | Con... | 2 M... | DB2... | Desktop » << 3:32 PM

Data IMPORT in DB2



3. Data IMPORT in DB2

Before importing data from any file in DB2, a table must be created in which data will be imported.

How to create table?

- a) **Command** -- create table <table name> (<column name> <data type>, <column name 2> <data type>...)

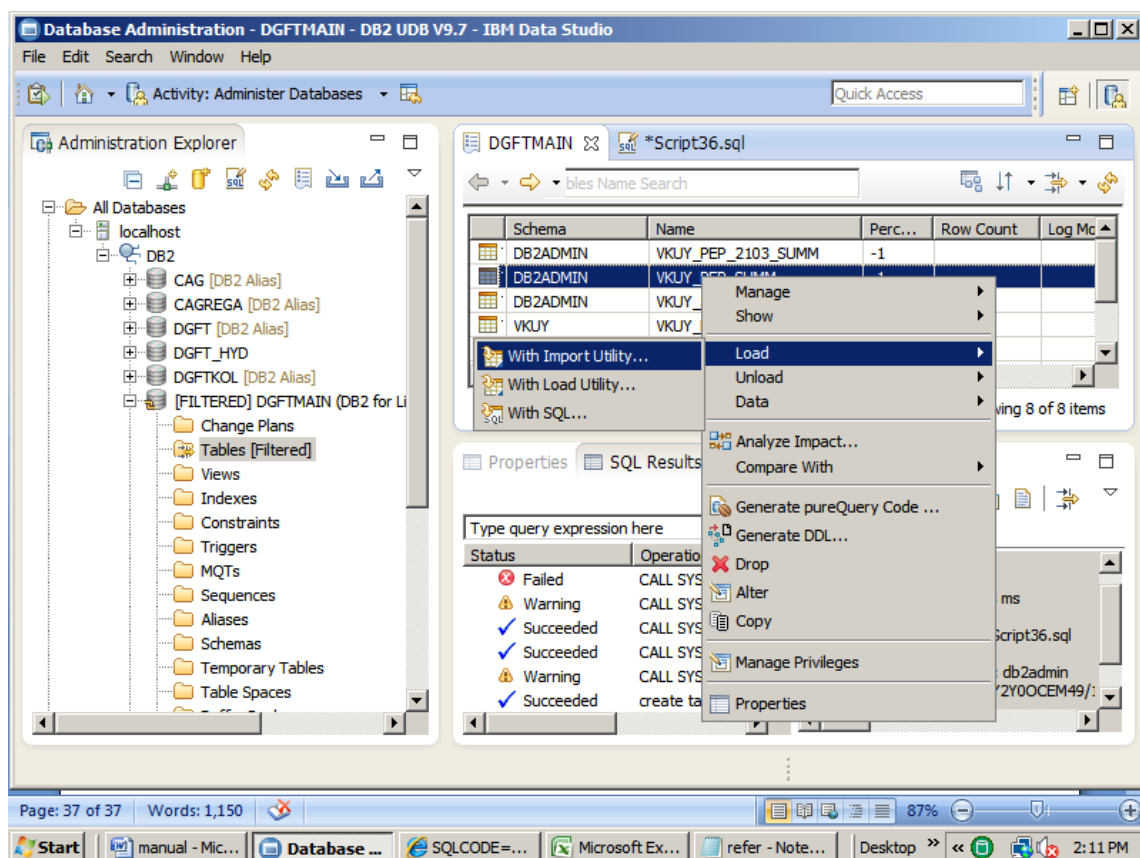
Eg. create table vkuy_summ (cont int, iec varchar(2))

- b) **Command** -- create table <table name> (<any select query, column name and data type of the table will be same as the result of the query>) with no data

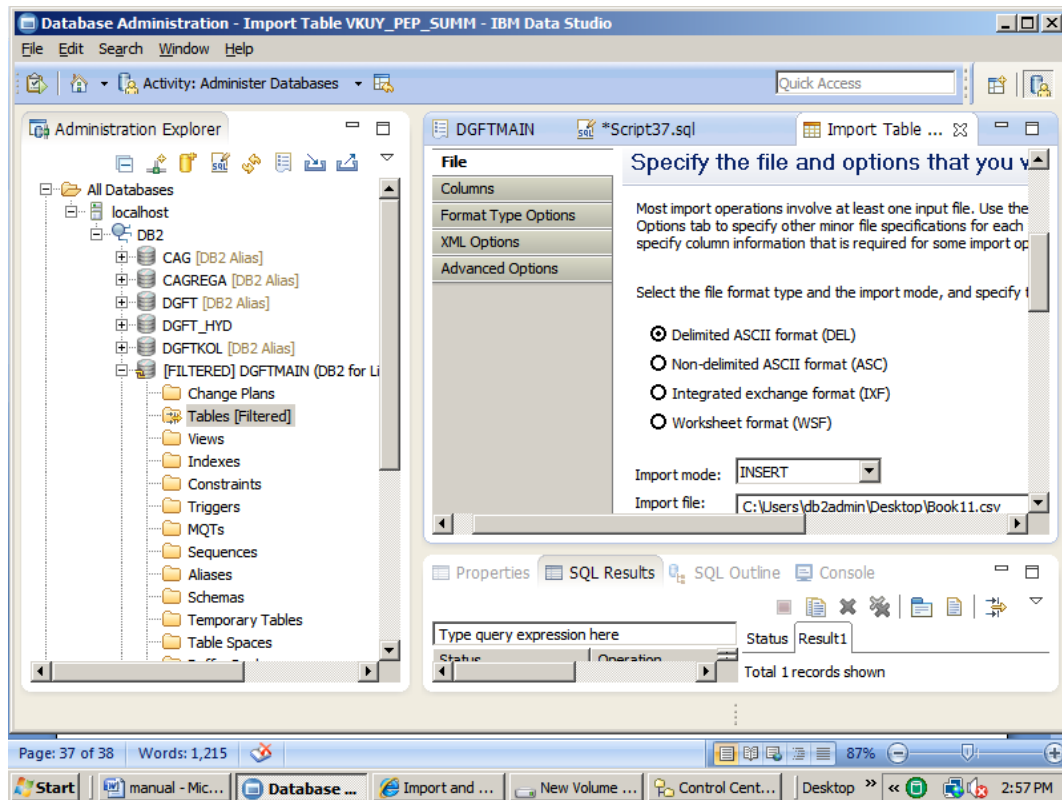
Eg. create table VKUY_PEP_summ as (select shbi_num_2103,FOB_FCCD_2103, conv_rat_2103 from VKUY.VKUY_PEP_2103 where conv_rat_2103>1 group by shbi_num_2103,FOB_FCCD_2103, conv_rat_2103) with no data
IMPORT steps:

3.1 Import using data studio client

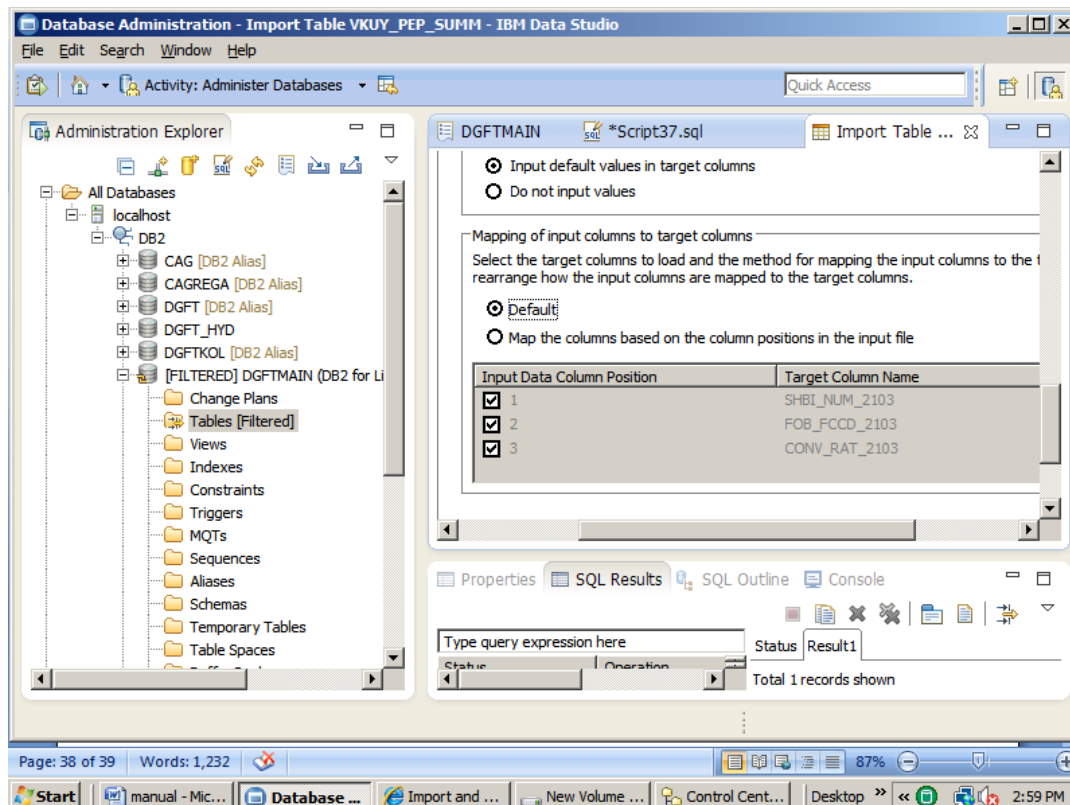
- a) Once table has been created, right click on table, click load and then import utility, it will open the import utility wizard

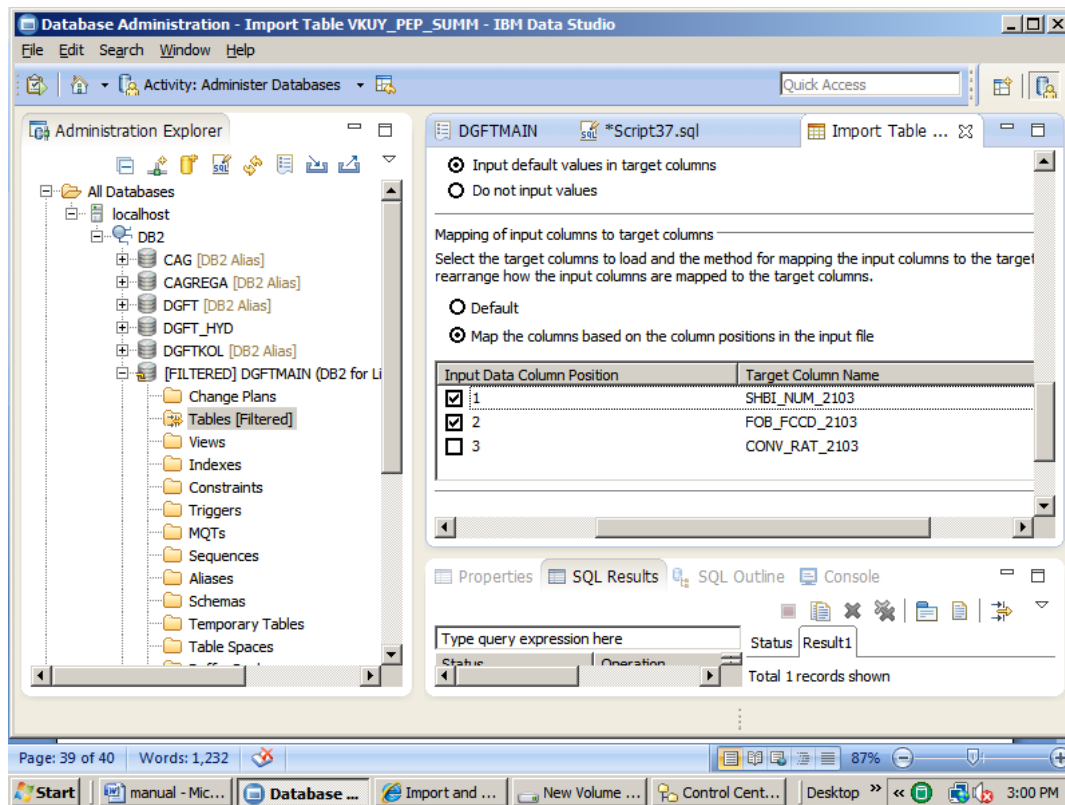


- b) Click on file tab on right side and select DEL format. select import mode as insert and mention the file name of the file to be imported. (Note: File should be saved in .csv file, in case of excel file, save excel in CSV format and then import)

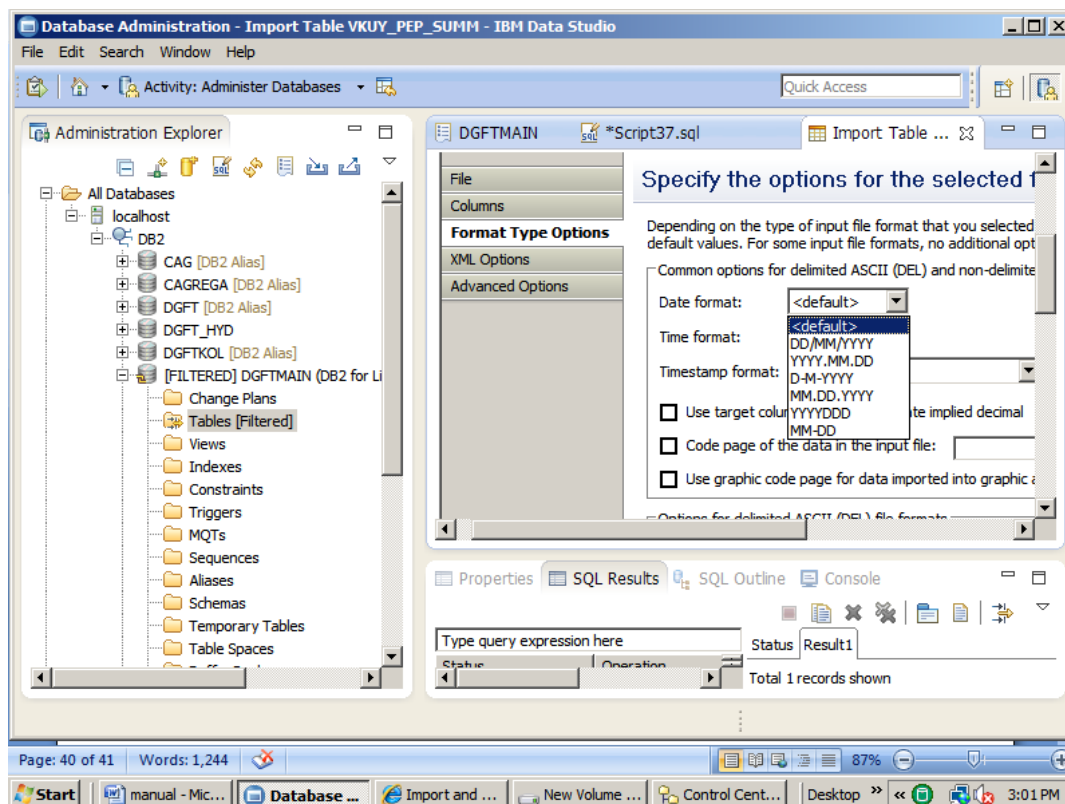


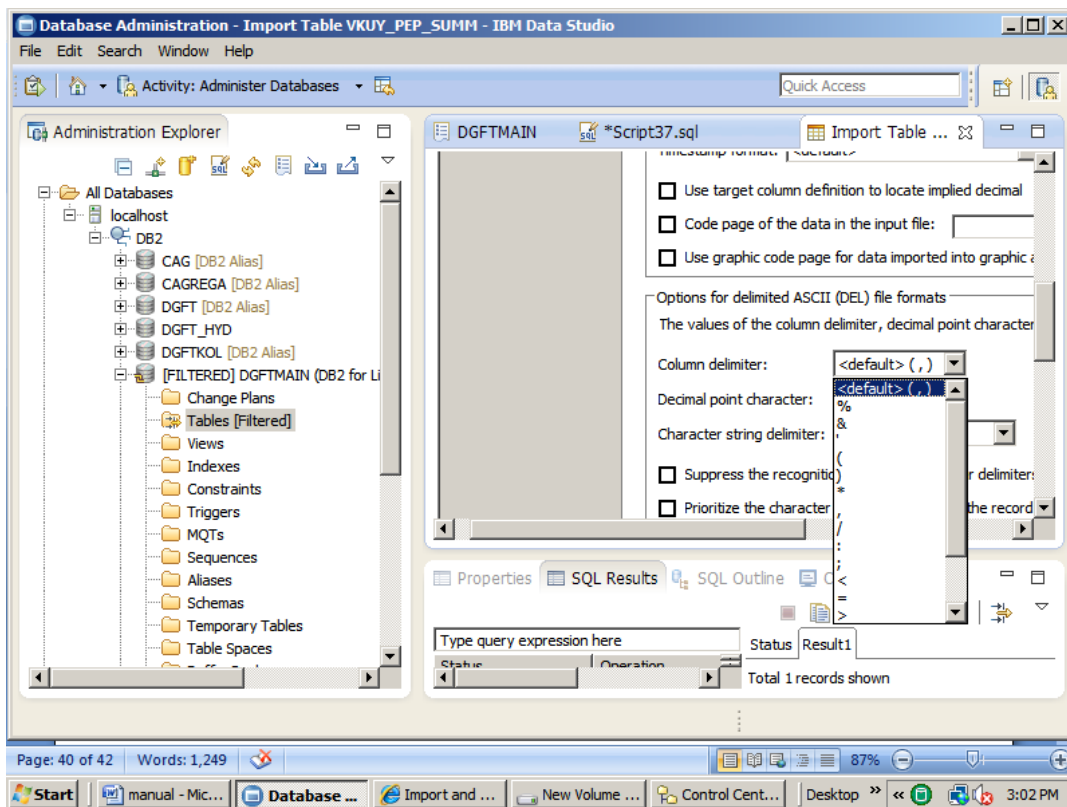
- c) Use column tab to select the column mapping of table and file to be imported.



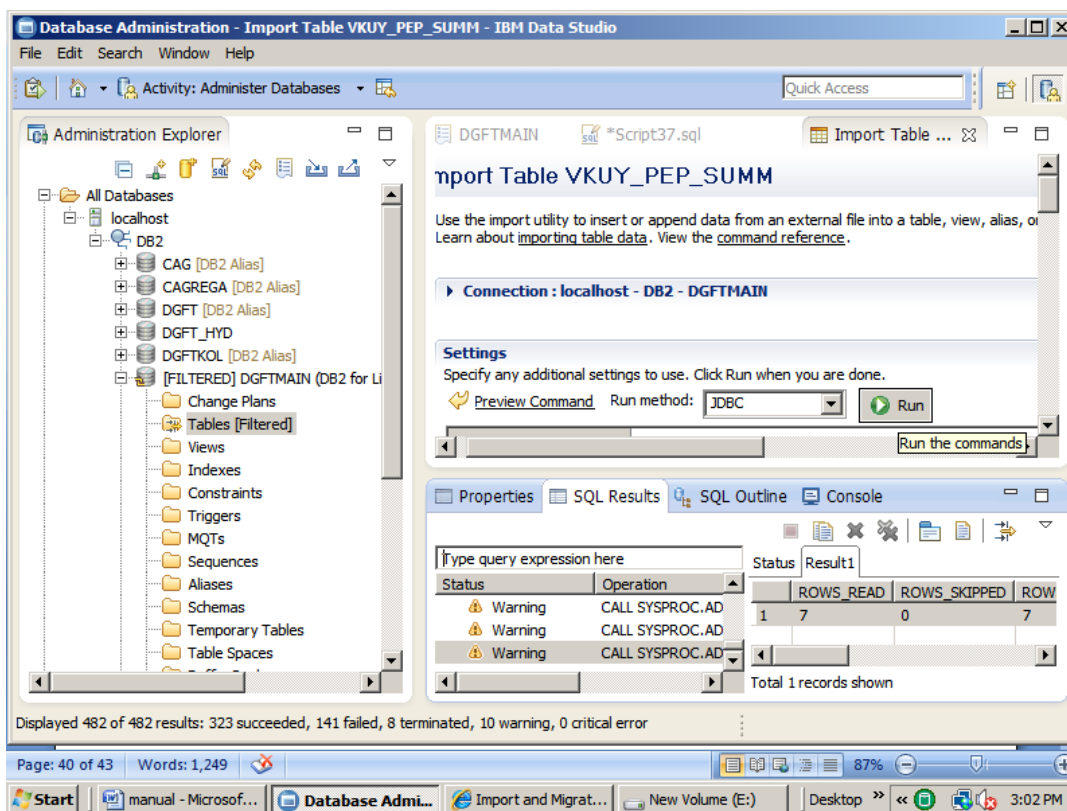


d) Use format type options tab to change the date or time format and column or string delimiter.





e) Click run with method as JDBC to import the file.



In result tab we can see how many rows have been read, skipped and loaded.

3.2 *IMPORT using DB2 command line (db2cmd)*

Import command -- DB2 IMPORT FROM "<filename>" OF DEL INSERT INTO <table name> (column name1, column name2, column name3 ..)

Eg.- DB2 IMPORT FROM "C:\Users\db2admin\Desktop\Book11.csv" OF DEL INSERT INTO DB2ADMIN.VKUY_PEP_SUMM (SHBI_NUM_2103, FOB_FCCD_2103, CONV_RAT_2103);

```

Local database alias = DGFTMAIN

db2 => EXPORT TO "exp_file" OF DEL modified by charde1"" MESSAGES "ww" SELECT *
Number of rows exported: 1152

db2 => EXPORT TO "exp_file" OF DEL modified by coldel: charde1"" MESSAGES "ww" S
Number of rows exported: 1152

db2 => IMPORT FROM "< C:\Users\db2admin\Desktop\Book11.csv>" OF DEL INSERT INTO
SQL3109N The utility is beginning to load data from file "<
C:\Users\db2admin\Desktop\Book11.csv>".

SQL3196N The input file was not found.

SQL3110N The utility has completed processing. "0" rows were read from the
input file.

db2 => IMPORT FROM "C:\Users\db2admin\Desktop\Book11.csv" OF DEL INSERT INTO DB2
SQL3109N The utility is beginning to load data from file
"C:\Users\db2admin\Desktop\Book11.csv".

SQL3125W The character data in row "1" and column "2" was truncated because
the data is longer than the target database column.

SQL3110N The utility has completed processing. "7" rows were read from the
input file.

SQL3221W ...Begin COMMIT WORK. Input Record Count = "7".

SQL3222W ...COMMIT of any database changes was successful.

SQL3149N "7" rows were processed from the input file. "7" rows were
successfully inserted into the table. "0" rows were rejected.

Number of rows read          = 7
Number of rows skipped       = 0
Number of rows inserted      = 7
Number of rows updated       = 0
Number of rows rejected      = 0
Number of rows committed    = 7

db2 =>
db2 =>

```

Data EXPORT in ORACLE



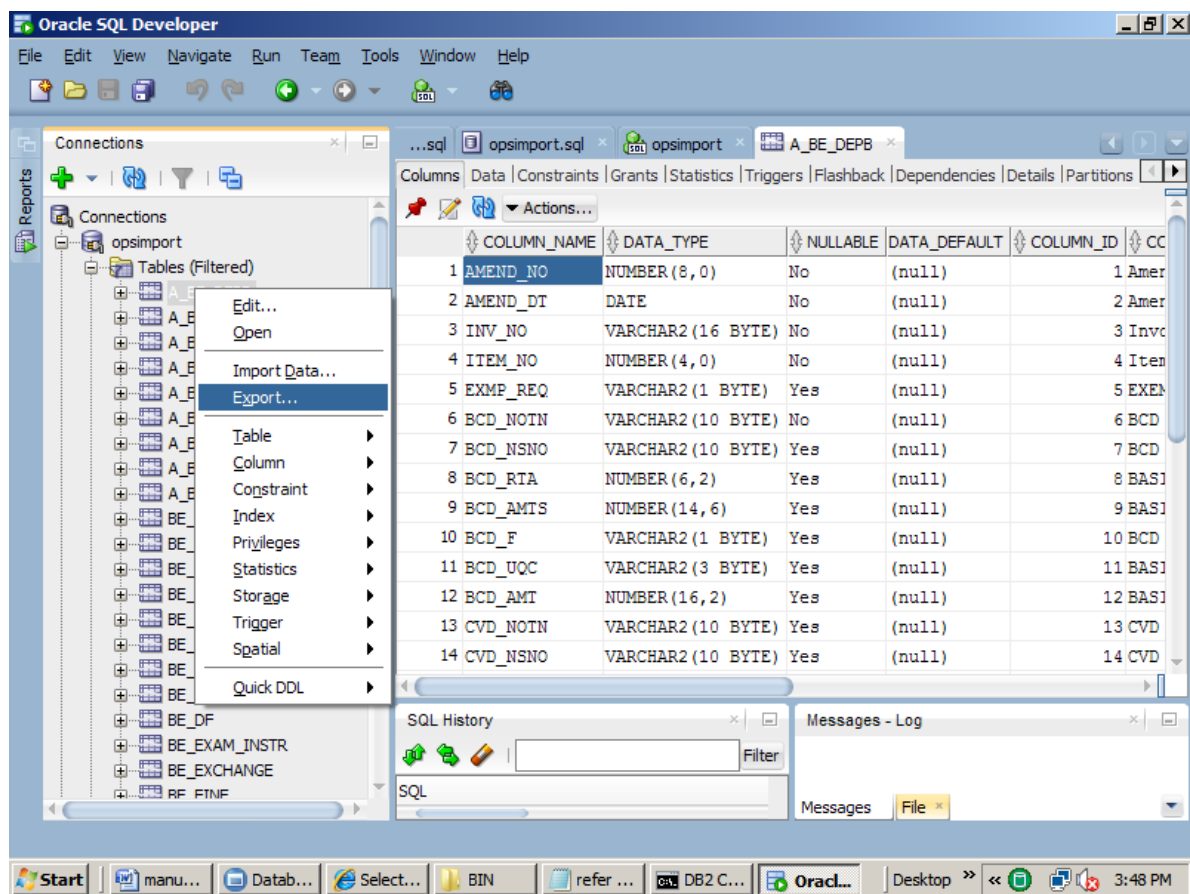
4. Data EXPORT in ORACLE

4.1 Export using SQL Developer

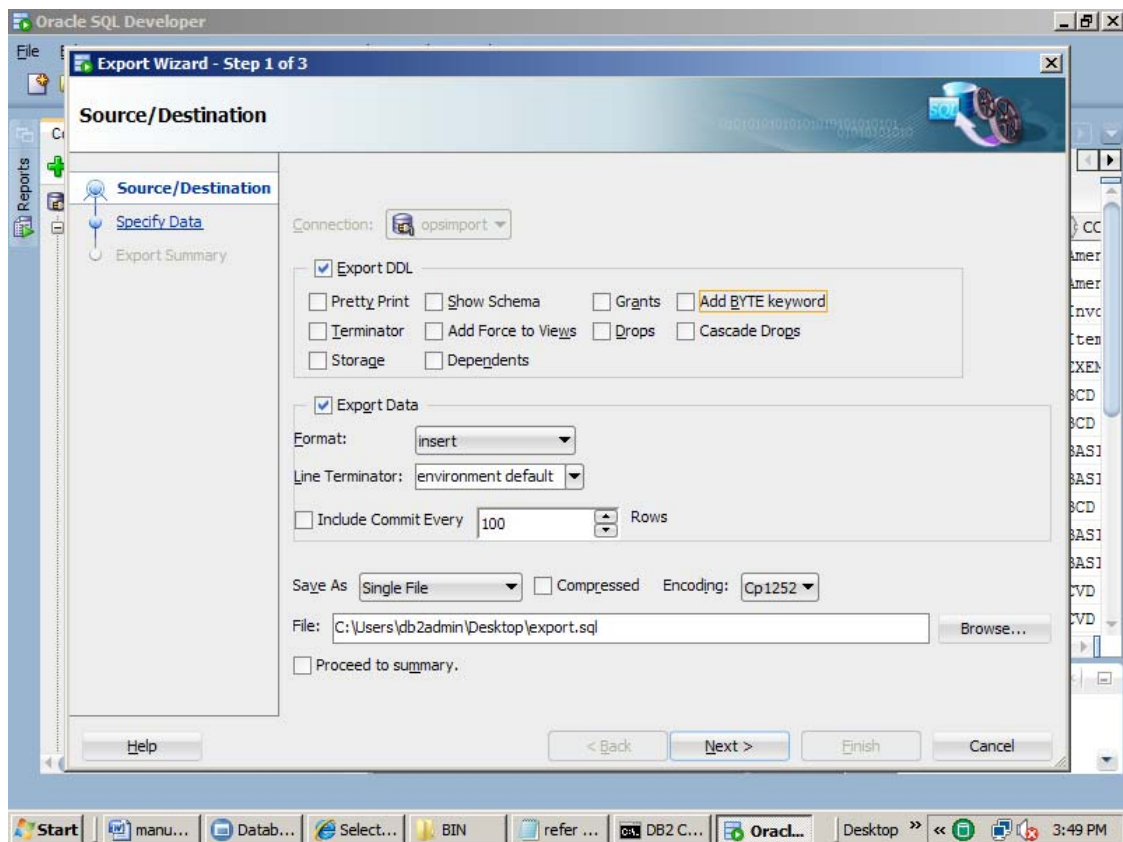
There are two methods to export data using SQL developer

Option 1

- a) Right click on table to be exported and click export, it will open the export wizard



- b) Tick mark export data option and export ddl in case we want to export table structure as well. Click next

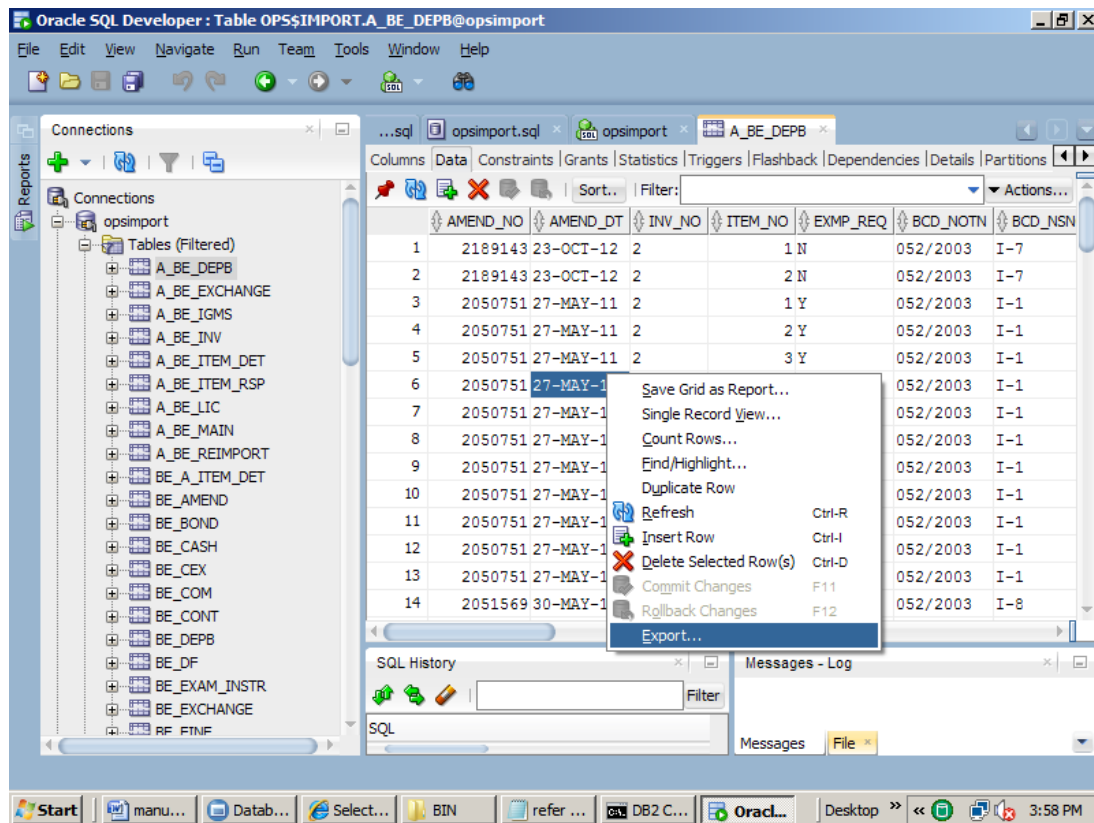


- c) Click next and finish.

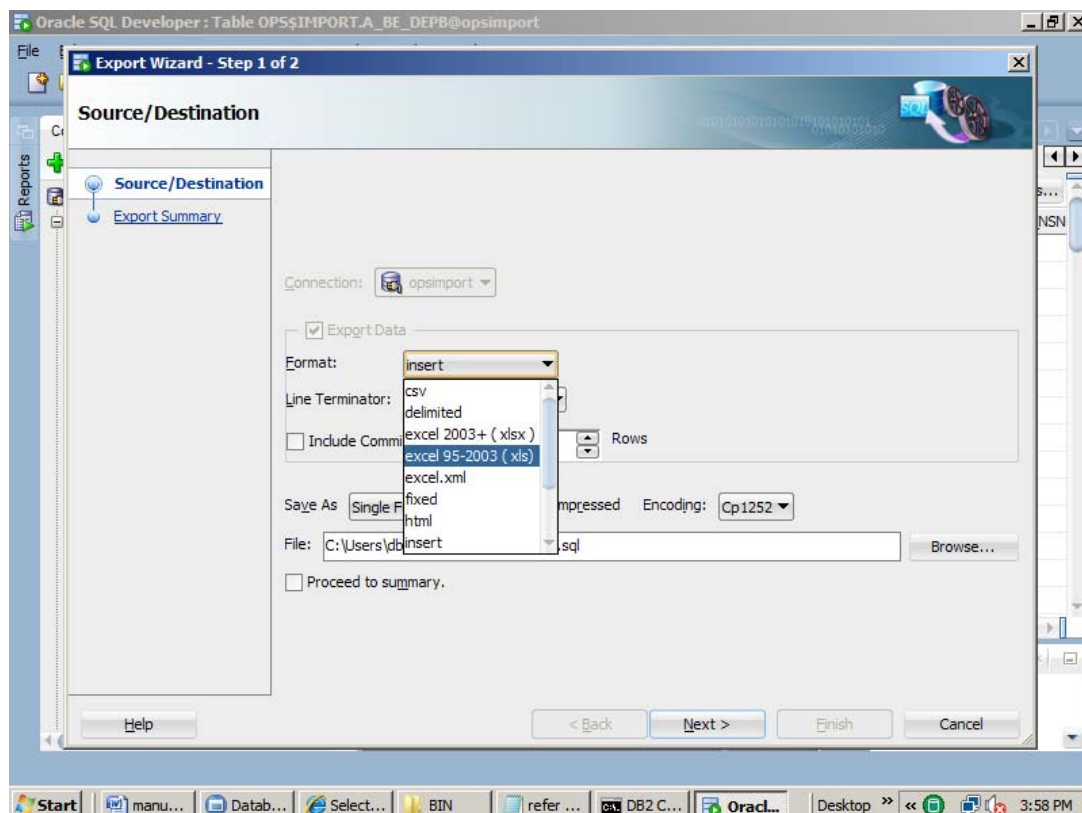
Note: It will import the table DDL and data in the insert statement format. **Content of same file can be used as it is in Db2 to create a table and insert the data in table**

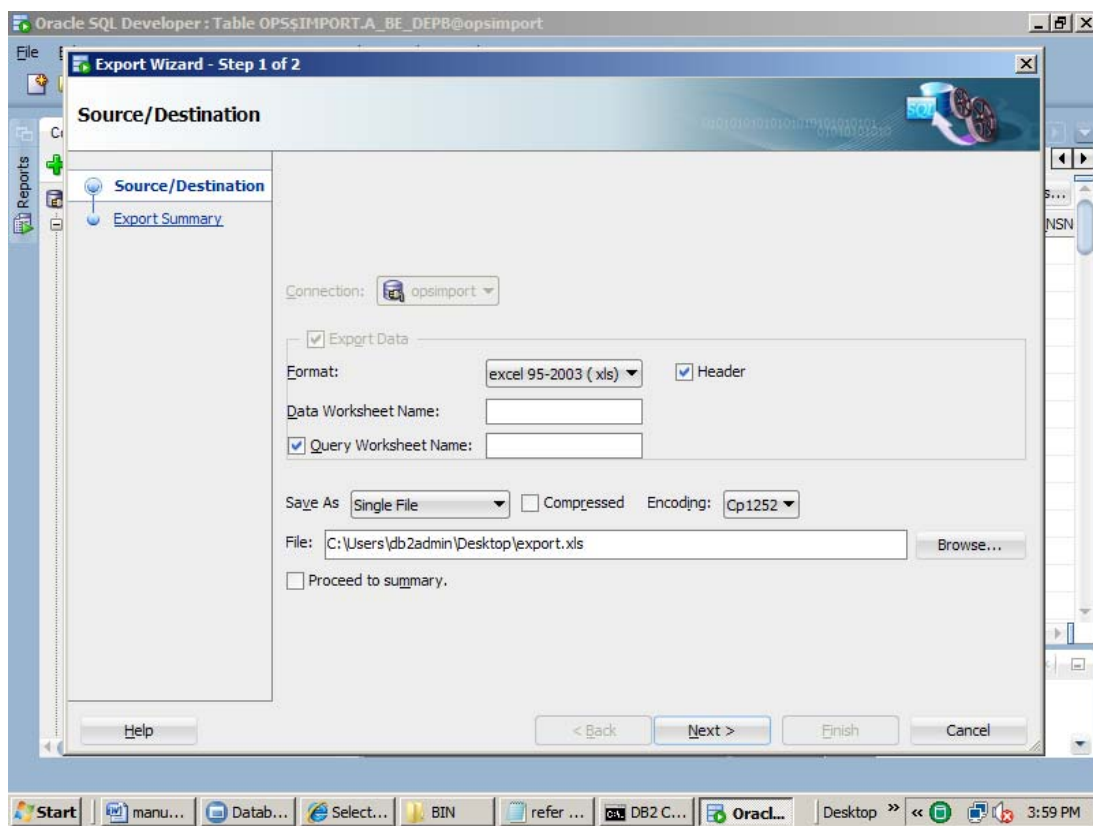
Option 2

- a) Execute the query and right click on the result and then click export. It will open export wizard.



- b) Select the format of the file in which we want to save the data. Select the file name and click next.





Click next and then finish, it will export the o/p data.

Data IMPORT in ORACLE

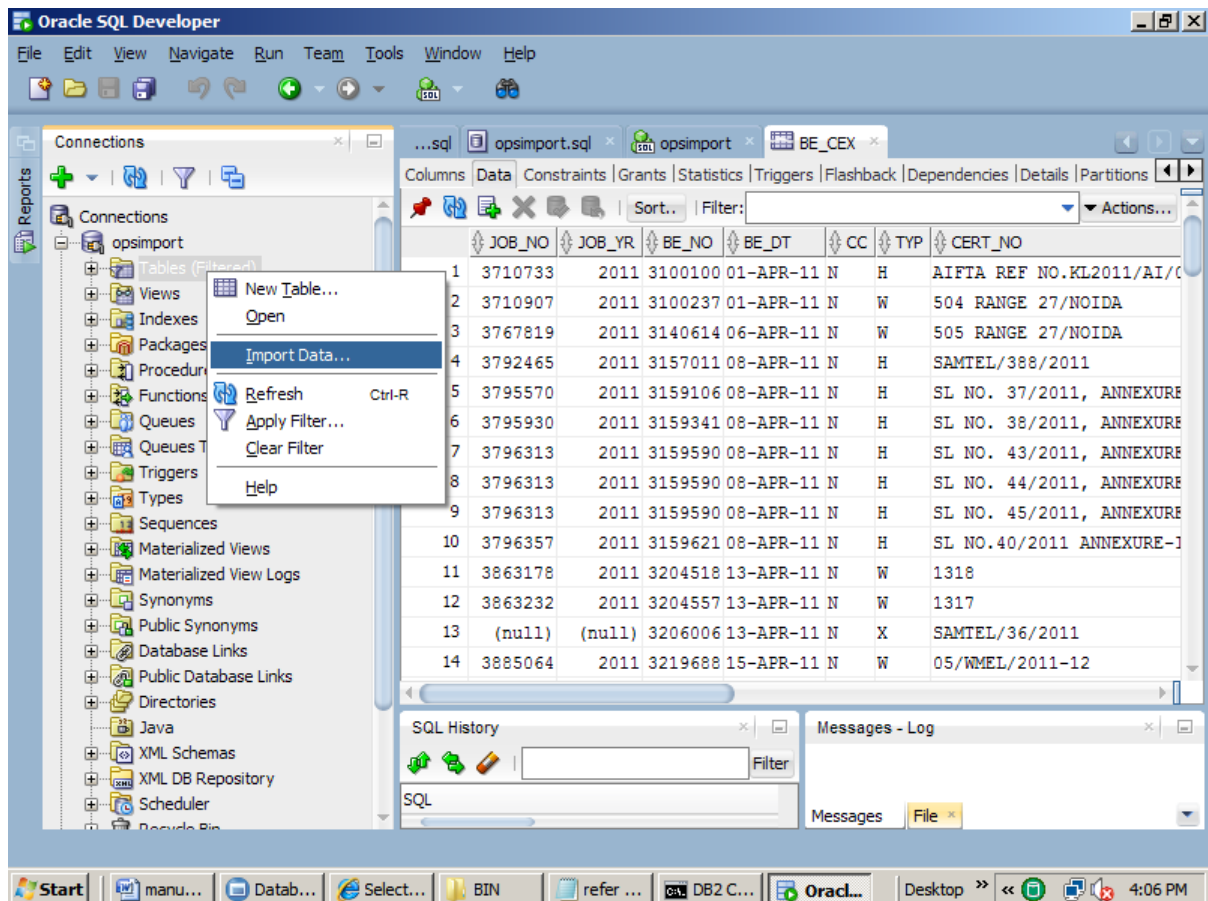


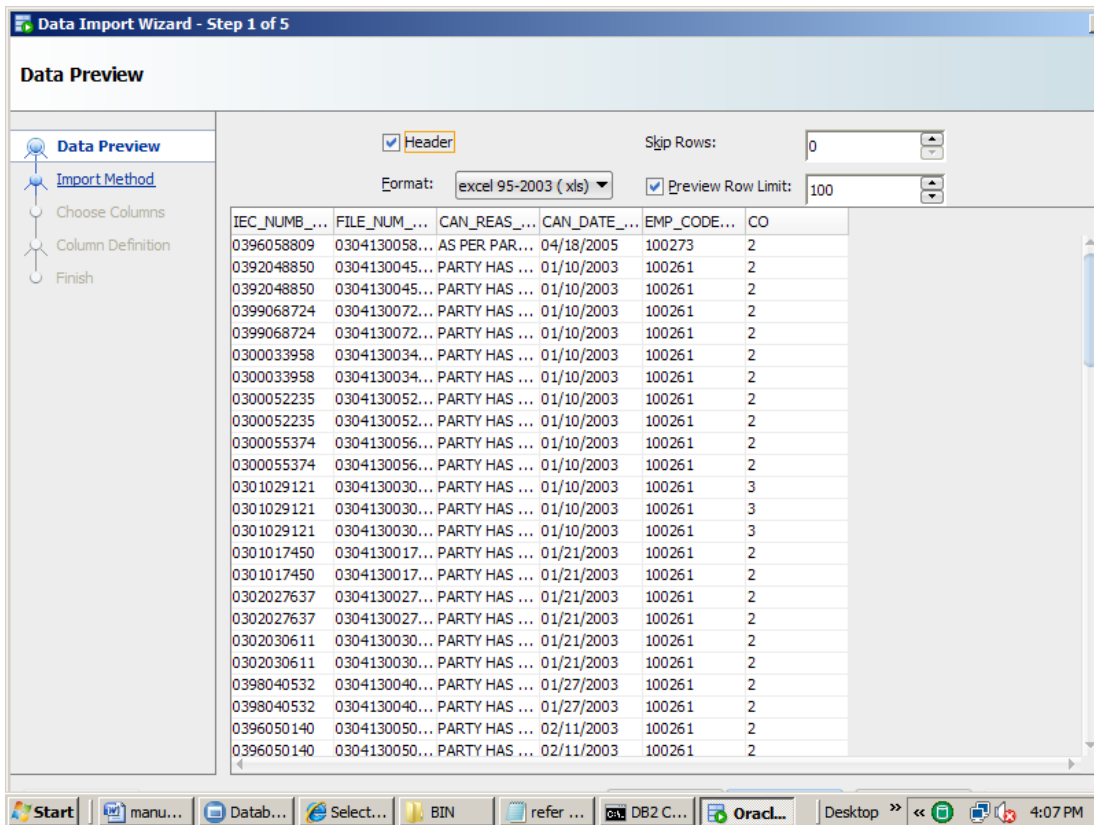
5. Data *IMPORT* in ORACLE

5.1 Import using SQL Developer

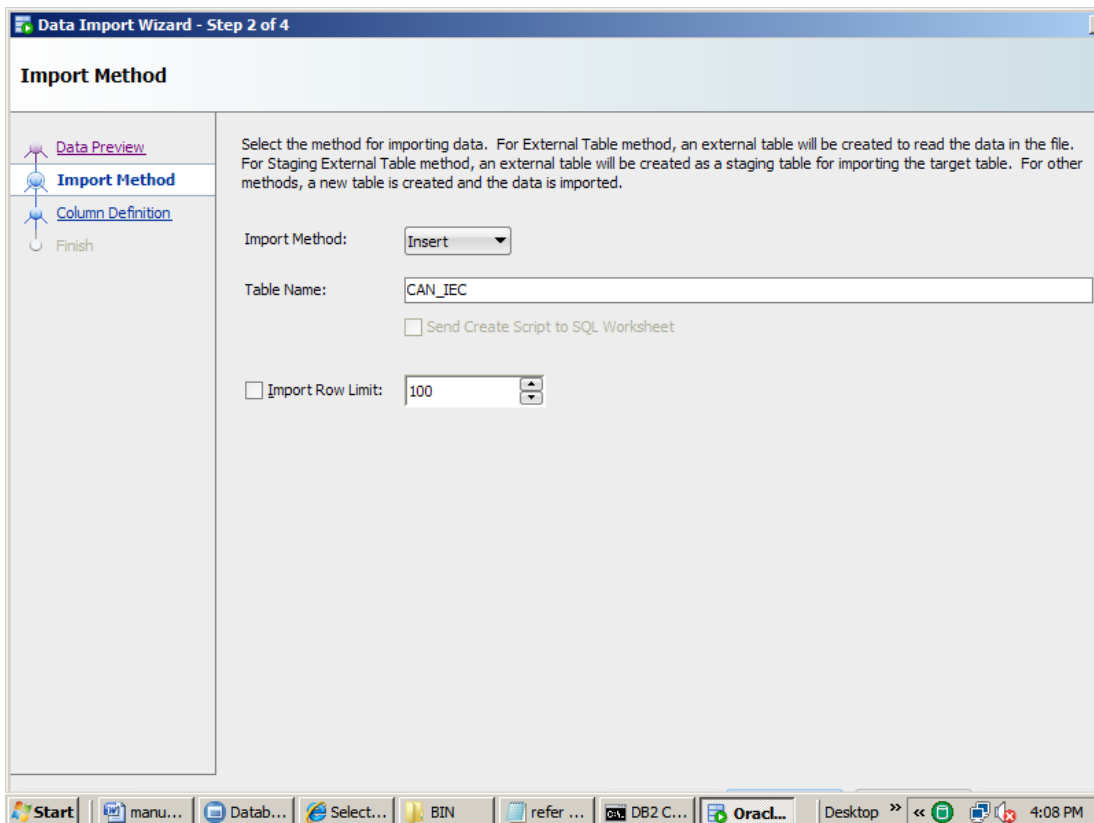
- a) Right click of tables and click import; it will open the import wizard.

Note: In Oracle there is no need to create table before import and we can import data in any file format (xls, csv, txt).

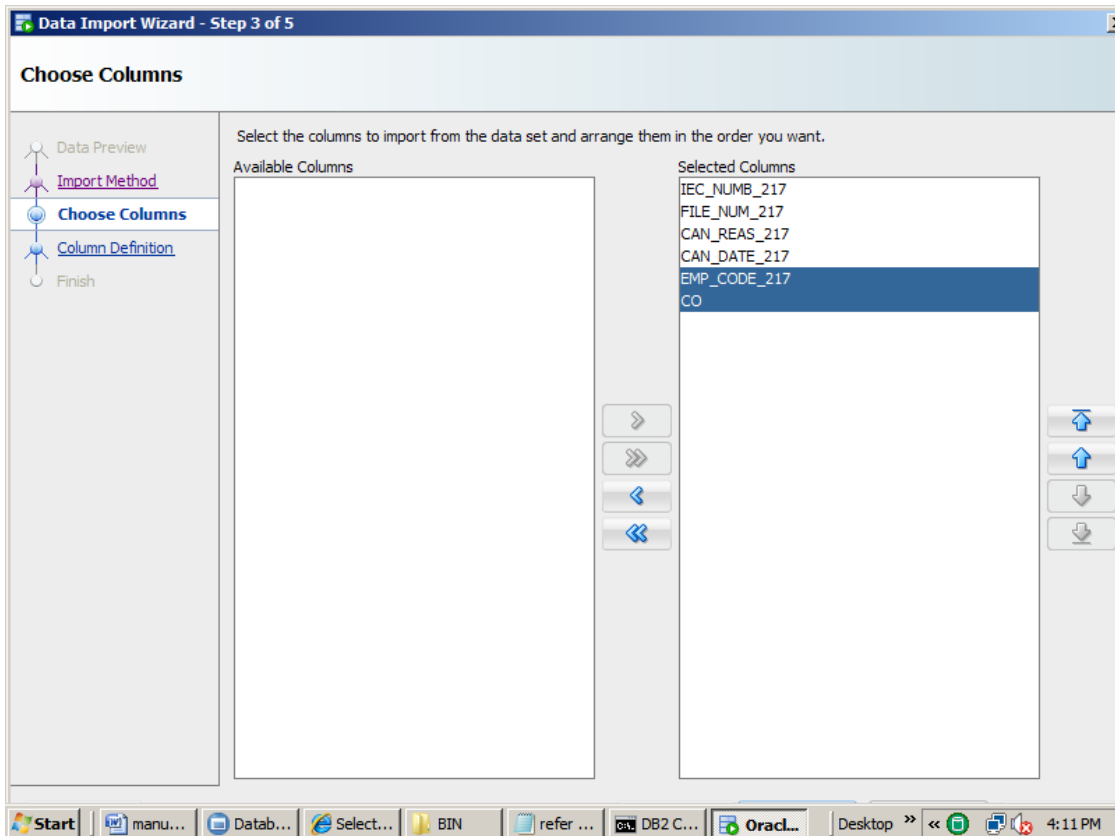
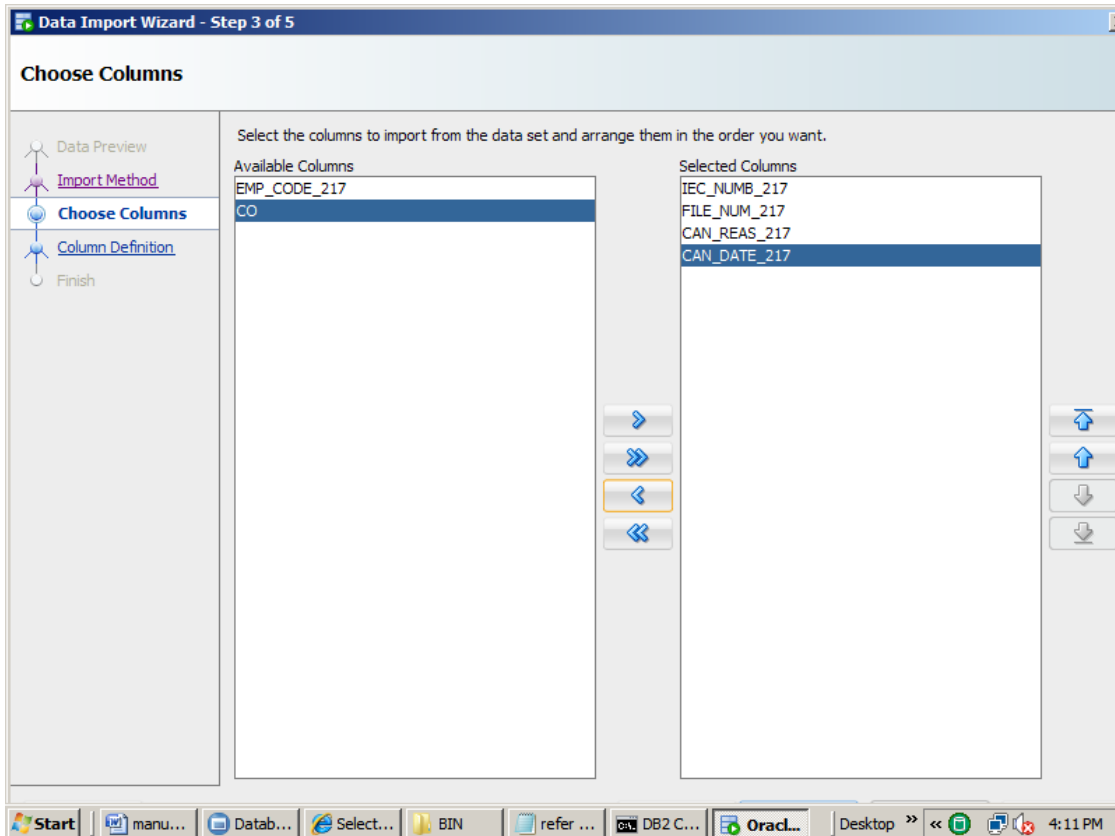




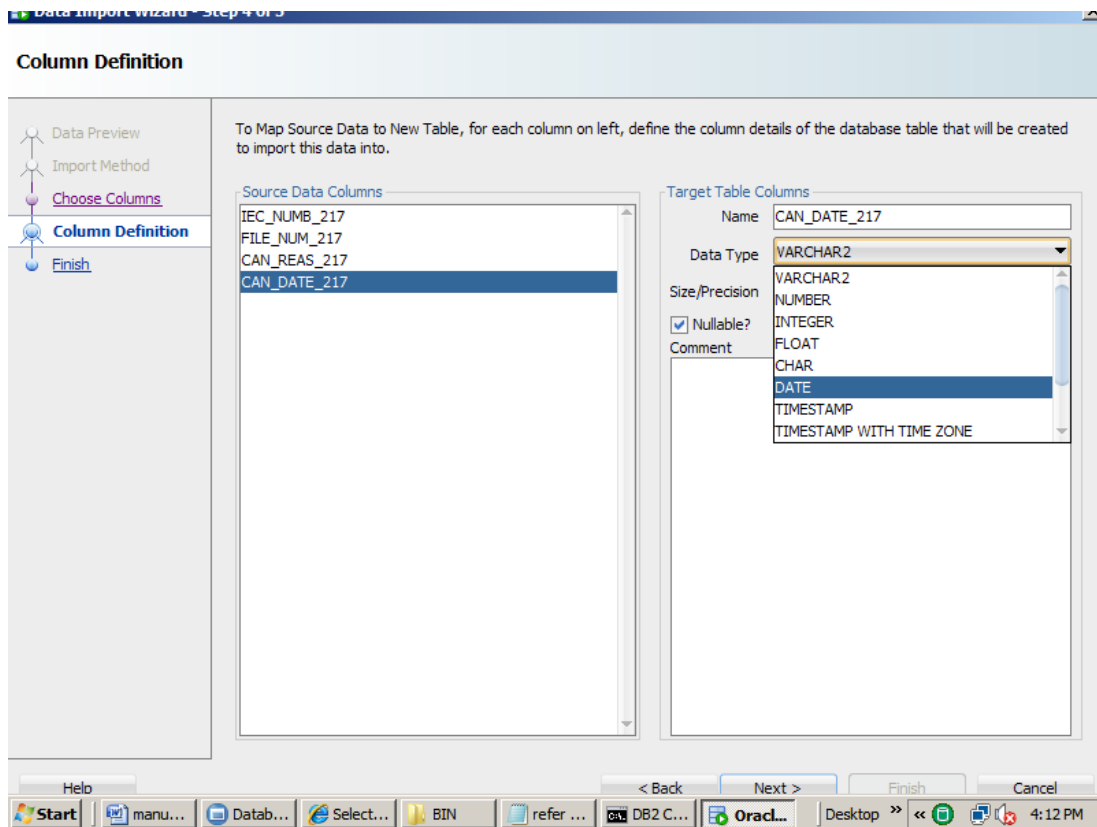
- b) Click on import method tab and provide table name. Select insert option as import method and click next.



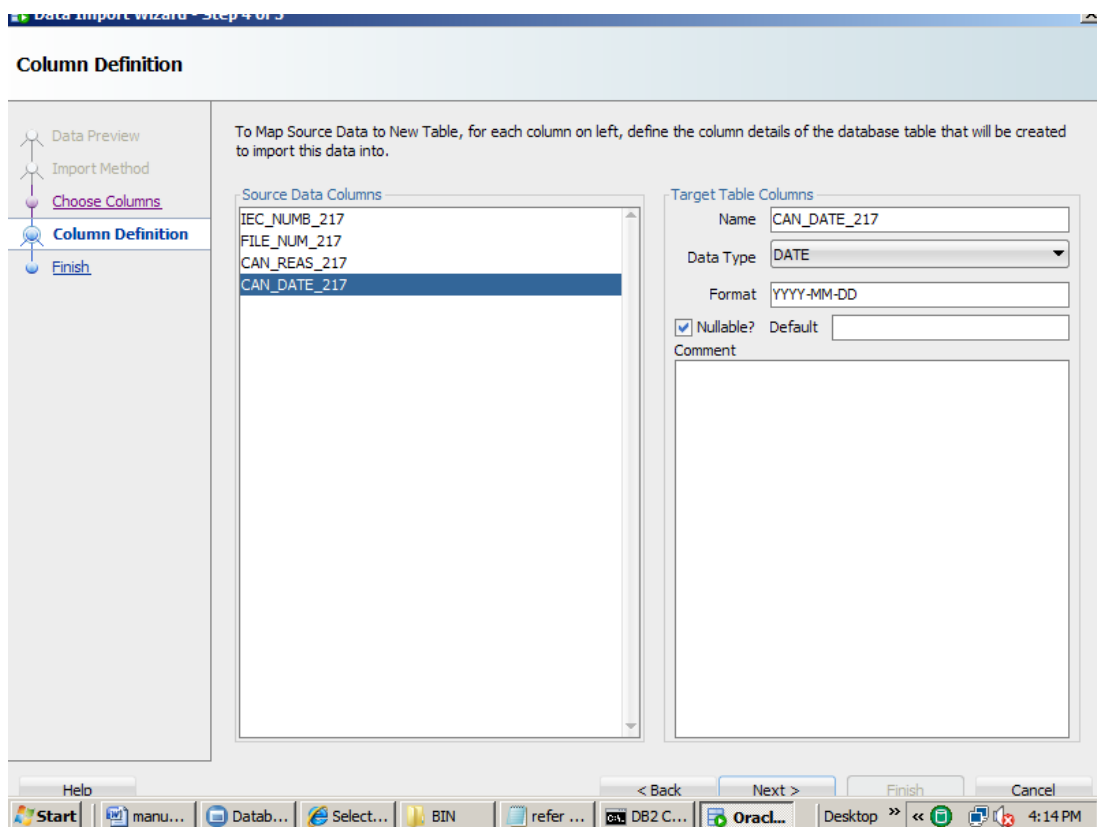
- c) Use choose columns tab to select the columns you want to import from the file.



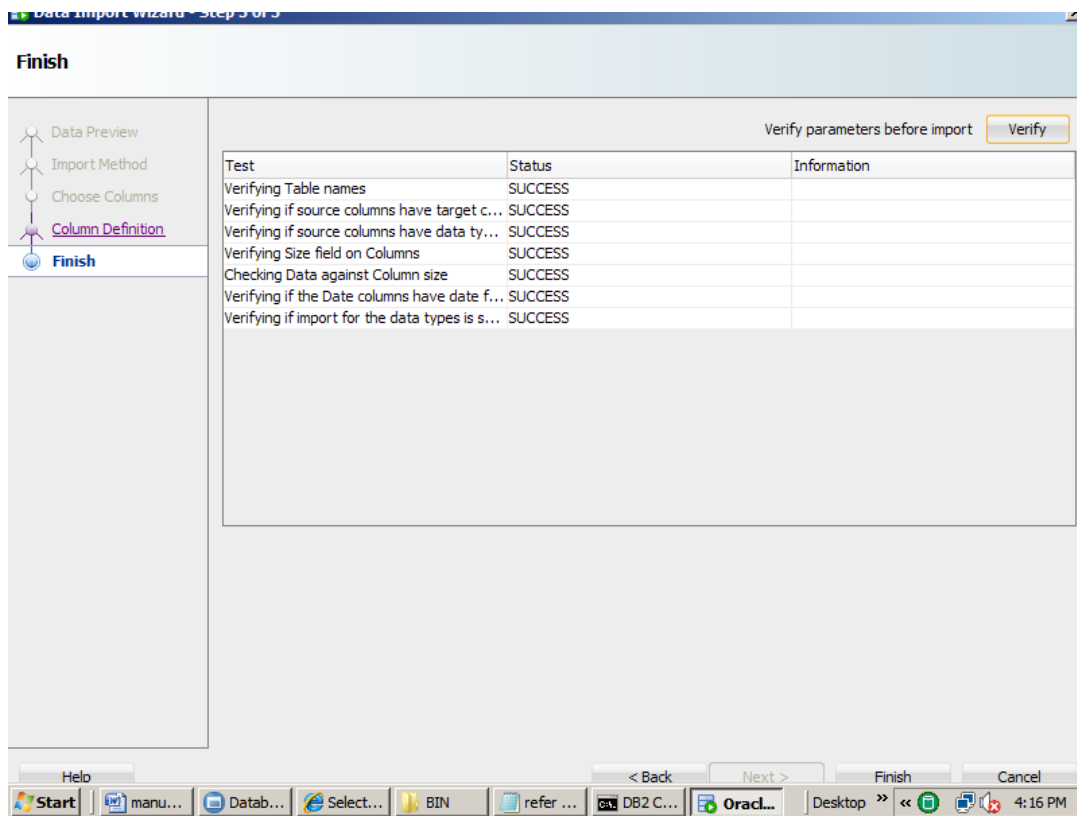
- d) Click next to provide the column definitions. It will take data type automatically from the file. Check if the correct data type is selected, especially in date format columns/data.



- e) Provide the date format as per the format showed in data preview tab.



f) Click next to verify and then finish importing the data.



Data Analysis - DB2/Oracle Database



6. *Data Analysis - DB2/Oracle Database*

SQL is used for performing data analysis of the data stored in RDBMS like DB2 and Oracle. While the concept, query syntax and basic command remain same in both Db2 and Oracle for most of the data analysis logic. Only some of the syntax and functions differs which is mentioned in chapter 7.

6.1 *Create tables and views*

This section describes how you can create and manipulate tables and views. The relationship of tables and views is explored

Tables

The CREATE TABLE statement is used to define a table. Two methods to create a table

- a) **Command** -- create table <table name> (<column name> <data type>, <column name 2> <data type>...)

```
Eg. -- CREATE TABLE employee
(empno CHAR NOT NULL
,firstnme VARCHAR NOT NULL
,midinit CHAR NOT NULL
,lastname VARCHAR NOT NULL
,workdept CHAR
,phoneno CHAR
,hiredate DATE
,job CHAR WITH DEFAULT X
,edlevel SMALLINT NOT NULL
,SEX CHAR
,birthdate DATE
,salary DECIMAL,bonus DECIMAL(10,2)
,comm DECIMAL (8,2)
,primary key (empno, birthdate)
)
```

- b) **Command** -- create table <table name> (<any select query, column name and data type of the table will be same as the result of the query>) with no data

```
Eg. create table VKUY_PEP_summ as (select shbi_num_2103,FOB_FCCD_2103, conv_rat_2103
from VKUY.VKUY_PEP_2103 where conv_rat_2103>1 group by
shbi_num_2103,FOB_FCCD_2103, conv_rat_2103) with no data
```

View

A view is another way to look at the data in one or more tables (or other views).

Command – create view <view name> as <select statement for which you want to create a view>

Eg, a user of the following view will only see those rows (and certain columns) in the EMPLOYEE table where the salary of a particular employee is greater than or equal to the average salary for their particular department.

```
CREATE VIEW employee_view AS
SELECT a.empno, a.firstnme, a.salary, a.workdept
FROM employee a
```

```
WHERE a.salary >=
(SELECT AVG(b.salary)
FROM employee b
WHERE a.workdept = b.workdept);
```

A view need not always refer to an actual table. It may instead contain a list of values:

```
CREATE VIEW silly (c1, c2, c3)
AS VALUES (11, 'AAA', SMALLINT(22))
,(12, 'BBB', SMALLINT(33))
,(13, 'CCC', NULL);

SELECT * FROM silly ORDER BY c1 ASC;
```

```
O/P
C1 C2 C3
-- --- --
11 AAA 22
12 BBB 33
13 CCC -
```

Note that these rows are not stored anywhere in the database - they are instead created on the fly when the view is queried. Once the View is created, it can be queried in the same way as tables.

6.2 Insert Data

The INSERT statement is used to insert rows into a table.

Command 1 - INSERT INTO <table-name> (<column 1>,<column 2>..) values (value 1, value 2..)

Eg. INSERT INTO emp_act (projno, emendate, actno, empno) VALUES ('ABC',DATE(CURRENT TIMESTAMP),123,'500000');

```
INSERT INTO emp_act VALUES
('400000','ABC',10,NULL,DEFAULT,CURRENT DATE);
```

Command 2 – Insert into <table-name> <query whose o/p we want to insert into table>

```
Eg. INSERT INTO emp_act
SELECT id
,SUBSTR(UCASE(name),1,6)
,salary / 229
,123
,CURRENT DATE
,'2003-11-11'
FROM staff
WHERE id < 50;
```

If only some columns are inserted using the query, they need to be explicitly listed:

```
Eg. INSERT INTO emp_act (empno, actno, projno)
SELECT id
,MINUTE(CURRENT TIME)
,'DEF'
FROM staff
WHERE id < 40;
```

Note: One has to provide a list of the columns (to be inserted) if the set of values provided does not equal the complete set of columns in the target table, or are not in the same order as the columns are defined in the target table.

If multiple rows are inserted in one statement, and one of them violates a unique index check, all of the rows are rejected.

The NULL and DEFAULT keywords can be used to assign values to columns. One can also refer to special registers, like the current date and current time.

6.3 Using SQL to access data

6.3.1 Basic Functions

Following functions can be used to derive desired results of data analysis:

- **Distinct** – To select unique values of a column in a table.
Syntax – select distinct (<column name>) from <table name> where <if required>
- **COALESCE** -- To substitute any numeric or character value for a null, based on the column data type.
Syntax – select COALESCE(<column name>,0) from <table name> where <if required>
- **Avg** – To get the average (mean) value of a set of non-null rows. The columns(s) must be numeric.
Syntax -- select avg (<column name>) from <table name> where <if required>
- **COUNT** -- Get the number of values in a set of rows. The result is an integer. The value returned depends upon the options used:
 - COUNT(*) gets a count of matching rows.
 - COUNT(expression) gets a count of rows with a non-null expression value.
 - COUNT(DISTINCT expression) gets a count of distinct non-null expression values.
 Syntax -- select count (<column name>/*) from <table name> where <if required>
- **MAX, MIN** – To get the maximum/minimum value of a set of rows.
Syntax – select max(<column name>) from <table name> where <if required>
select min(<column name>) from <table name> where <if required>
- **SUM** – To Get the sum of a set of numeric values If DISTINCT is used, duplicate values are ignored. Null values are always ignored. If no rows match, the result is null.
Syntax – select sum(<column name>) from <table name> where <if required>
- **ABS** -- Returns the absolute value of a number (e.g. -0.4 returns + 0.4). The output field type will equal the input field type.
- **CEIL** -- Returns the next smallest integer value that is greater than or equal to the input (e.g. 5.5 returns 6.000). The output field type will equal the input field type
Syntax – select ceil(<column name>) from <table name> where <if required>
- **DATE** -- Converts the input into a date value. The nature of the conversion process depends upon the input type and length:
 - Timestamp and date input have the date part extracted.
 - Char or varchar input that is a valid string representation of a date or a timestamp (e.g."1997-12-23") is converted as is.
 - Char or varchar input that is seven bytes long is assumed to be a Julian date value in the format yyyynnn where yyyy is the year and nnn is the number of days since the start of the year (in the range 001 to 366).

- Numeric input is assumed to have a value which represents the number of days since the date "0001-01-01" inclusive. All numeric types are supported, but the fractional part of a value is ignored (e.g. 12.55 becomes 12 which converts to "0001-01-12").
Syntax – select date(<column name>) from <table name> where <if required>
- DAY/MONTH/YEAR -- Returns the day/month/year part of a date (or equivalent) value. The output format is integer.
Syntax – select day/month/year(<column name>) from <table name> where <if required>
- DAYNAME -- Returns the name of the day (e.g. Friday) as contained in a date (or equivalent) value. The output format is varchar(100).
Syntax – select dayname(<column name>) from <table name> where <if required>
- DAYOFWEEK -- Returns a number that represents the day of the week (where Sunday is 1 and Saturday is 7) from a date (or equivalent) value. The output format is integer.
Syntax – select dayofweek(<column name>) from <table name> where <if required>
- FLOOR -- Returns the next largest integer value that is smaller than or equal to the input (e.g. 5.945 returns 5.000). The output field type will equal the input field type
Syntax – select floor(<column name>) from <table name> where <if required>
- HOUR -- Returns the hour (as in hour of day) part of a time value. The output format is integer.
Syntax – select hour(<column name>) from <table name> where <if required>
- LOWER/UPPER -- Converts a mixed or upper-case/lower-case string to lower/upper case respectively. The output is the same data type and length as the input.
Syntax – select lower/upper(<column name>) from <table name> where <if required>
- LEFT/RIGHT -- The LEFT/RIGHT function has two arguments: The first is an input string of type char, varchar, clob, or blob. The second is a positive integer value. The output is the left/right most characters (equal to the number of positive integer value) in the string. Trailing blanks are not removed.
Syntax – select left/right(<column name>, <positive integer value, eg. 1/4/6>) from <table name> where <if required>
- LENGTH -- Returns an integer value with the internal length of the expression (except for double-byte string types, which return the length in characters). The value will be the same for all fields in a column, except for columns containing varying-length strings.
Syntax – select length (<column name>) from <table name> where <if required>
- LOCATE -- Returns an integer value with the absolute starting position of the first occurrence of the first string within the second string. If there is no match the result is zero. The optional third parameter indicates where to start the search.
Syntax – select locate(<find string, eg. 'ABC' >, <column name>, <start position (by default it is 1)>) from <table name> where <if required>
- TRIM/LTRIM/RTRIM -- Remove blanks / leading blanks / trailing blanks, from the column.
Syntax – select trim/ltrim/rtrim(<column name>) from <table name> where <if required>
- NULLIF -- Returns null if the two values being compared are equal, otherwise returns the first value.
Syntax -- select nullif(<column name 1>, <column name 2>) from <table name> where <if required>
- REPLACE -- Replaces all occurrences of one string with another. The output is of type varchar(4000).
Syntax -- select replace(<column name >, <search for string, eg. 'AB'>,<replace with string, eg. 'CD'>) from <table name> where <if required>

- **SUBSTR** -- Returns part of a string. If the length is not provided, the output is from the start value to the end of the string. If the length is provided, and it is longer than the field length, a SQL error results.
Syntax – select substr(<column name>, <start position, by default it is 1>, <length of string required>) from <table name> where <if required>
- **PLUS/MINUS/MULTIPLY/DIVIDE** -- The PLUS/MINUS/MULTIPLY/DIVIDE function is same old algebraic mathematics functions. One can use it the old fashioned way. Both the PLUS and MINUS functions can be used to add and subtract numbers, and also date and time values.
Syntax – select <column name1>+/-*//<column name 2> from <table name> where <if required>
- **Relational Operators**

=	Equal
<> or !=	Not Equal
<	Less Than
>	Greater Than
>=	Greater Than Equal To
<=	Less Than Equal To

6.3.2 Conditional Selection

The WHERE clause is used to specify that only certain rows of the table are displayed, based on the criteria described in that WHERE clause.

Eg. If you wanted to see the EMPLOYEEIDNO's of those making at or over \$50,000, use the following:

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY >= 50000;
```

The WHERE description, SALARY >= 50000, is known as a condition (an operation which evaluates to True or False).

We can use ADD / OR in where clause to select data for two or more specific conditions.

The AND operator joins two or more conditions, and displays a row only if that row's data satisfies.

Eg. To display all staff making over \$40,000, use:

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY > 40000 AND POSITION = 'Staff';
```

The OR operator joins two or more conditions, but returns a row if ANY of the conditions listed hold true.

Eg. To see all those who make less than \$40,000 or have less than \$10,000 in benefits, listed together, use the following query:

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE SALARY < 40000 OR BENEFITS < 10000;
```

AND & OR can be combined, for example:

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEESTATISTICSTABLE
WHERE (POSITION = 'Manager' AND SALARY > 60000) OR (BENEFITS > 12000);
```

6.3.3 In, Between and Like Clause

The **IN** predicate is used to compare a value with several other values. Eg:

```
SELECT NAME
FROM STAFF
WHERE DEPT IN (20, 15)
```

This example is equivalent to:

```
SELECT NAME
FROM STAFF
WHERE DEPT = 20 OR DEPT = 15
```

Similarly, NOT IN lists all rows excluded from the IN list.

The **BETWEEN** predicate compares a single value to an inclusive range of values (named in the BETWEEN predicate). Eg: find the names of employees who earn between \$10, 000 and \$20, 000

```
SELECT LASTNAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 10000 AND 20000
```

This is equivalent to:

```
SELECT LASTNAME
FROM EMPLOYEE
WHERE SALARY >= 10000 AND SALARY <= 20000
```

The next example finds the names of employees who earn less than \$10, 000 or more than \$20, 000:

```
SELECT LASTNAME
FROM EMPLOYEE
WHERE SALARY NOT BETWEEN 10000 AND 20000
```

The **LIKE** predicate to search for strings that have certain patterns. The pattern is specified through percentage signs and underscores.

- The underscore character (_) represents any single character.
- The percent sign (%) represents a string of zero or more characters.
- Any other character represents itself.

Eg: To select employee names that are seven letters long and start with the letter 'S':

```
SELECT NAME
FROM STAFF
WHERE NAME LIKE 'S _ _ _ _ _ _ _'
```

Eg: To select names of employees that do not start with the letter 'S':

```
SELECT NAME
FROM STAFF
WHERE NAME NOT LIKE 'S%'
```

Eg: To select names of employees that contain letter 'S':

```
SELECT NAME
FROM STAFF
WHERE NAME LIKE '%S%'
```

Eg: To select names of employees that ends with letter 'S':

```
SELECT NAME
FROM STAFF
WHERE NAME LIKE '%S'
```

6.3.4 Joins

A join is used to relate sets of rows in two or more logical tables. The tables are always joined on a row-by-row basis using whatever join criteria are provided in the query. In a join, the matching rows are joined side-by-side to make the result table.

The most important data in a relational database is not that stored in the individual rows. Rather, it is the implied relationships between sets of related rows. For example, individual rows in an EMPLOYEE table may contain the employee ID and salary - both of which are very important data items. However, it is the set of all rows in the same table that gives the gross wages for the whole company, and it is the (implied) relationship between the EMPLOYEE and DEPARTMENT tables that enables one to get a breakdown of employees by department and/or division.

Join Types

A generic join matches one row with another to create a new compound row. Joins can be categorized by the nature of the match between the joined rows. Following methods for each join type and how to code it in SQL.

TABLE – PROJECT

Name	Proj
Haas	AD3100
Thompson	PL2100
Walker	MA2112
Lutz	MA2111

TABLE – STAFF

Name	Job
Haas	Pres
Thompson	Manager
Luchi	Sales
Nick	Analyst

Inner Join

An inner-join is another name for a standard join in which two sets of columns are joined by matching those rows that have equal data values. Most of the joins that one writes will probably be of this kind and, assuming that suitable indexes have been created, they will almost always be very efficient.

Eg. To find the list of employees who are assigned to a project.

```
SELECT a.NAME,
a.PROJ, b.NAME, b.JOB
FROM PROJECT A, STAFF B
WHERE A.NAME = B.NAME
```

Result --

NAME	PROJ	NAME	JOB
Haas	AD3100	Haas	PRES
Thompson	PL2100	Thompson	MANAGER

The result of the inner join consists of rows that have matching values for the NAME column in the right and the left tables - both 'Haas' and 'Thompson' are included in the STAFF table that lists all the employees and in the PROJECT table that lists full-time and contract employees assigned to a project.

Left Outer Join

A left outer join is the same as saying that we want all of the rows in the first table listed, plus any matching rows in the second table:

Eg: To find the name and project number of all employees (listed in PROJECT) and their job title, if they are employee (listed in STAFF), STAFF is designated as the right table and PROJECT as the left table.

```
SELECT a.NAME, a.PROJ,
b.NAME, b.JOB
FROM PROJECT A LEFT OUTER JOIN STAFF B
ON a.NAME = b.NAME
```

Result --

NAME	PROJ	NAME	JOB
-----	-----	-----	-----
Haas	AD3100	Haas	PRES
Lutz	MA2111	-	-
Thompson	PL2100	Thompson	MANAGER
Walker	MA2112	-	-

Rows with values in all columns are the result of the inner join. These are rows that satisfy the join condition: 'Haas' and 'Thompson' are listed in both PROJECT (left table) and STAFF (right table). For rows that the join condition was not satisfied, the null value appears on columns of the right table: 'Lutz' and 'Walker' are contract employees listed in the PROJECT table and not in the STAFF table. Note that all rows from the left table are included in the result set.

Right Outer Join

A right outer join is the inverse of a left outer join. One gets every row in the second table listed, plus any matching rows in the first table:

Eg: To find the name and job title of all full-time employees (listed in STAFF) and their project number, if they are assigned to one (listed in PROJECT), STAFF is designated as the right table and PROJECT as the left table.

```
SELECT A.NAME,
A.PROJ, B.NAME, B.JOB
FROM PROJECT A RIGHT OUTER JOIN STAFF B
ON A.NAME = B.NAME
```

Result --

NAME	PROJ	NAME	JOB
-----	-----	-----	-----
Haas	AD3100	Haas	PRES
-	-	Lucchessi	SALESREP
-	-	Nicholls	ANALYST
Thompson	PL2100	Thompson	MANAGER

Rows that satisfy the join condition: 'Haas' and 'Thompson' are listed in both PROJECT (left table) and STAFF (right table). For rows that the join condition was not satisfied, the null value appears on columns of the right table: 'Lucchessi' and 'Nicholls' are the employees that are not assigned to a project. While they are listed in STAFF, they are not in PROJECT. Note that all rows from the right table are included in the result set.

Full Outer Joins

A full outer join occurs when all of the matching rows in two tables are joined, and there is also returned one copy of each non-matching row in both tables.

Eg. To find the name of all full-time employees, including the ones that are not assigned to a project, and contract employees:

```
SELECT A.NAME, A.PROJ,
B.NAME, B.JOB
FROM PROJECT A FULL OUTER JOIN STAFF B
ON A.NAME = B.NAME
```

Result –

NAME	PROJ	NAME	JOB
Haas	AD3100	Haas	PRES
-	-	Lucchessi	SALESREP
-	-	Nicholls	ANALYST
Thompson	PL2100	Thompson	MANAGER
Lutz	MA2111	-	-
Walker	MA2112	-	-

The result includes the left outer join, the right outer join and the inner join. All the employees are listed. Just like left outer join and right outer join, for values where the join condition was not satisfied, the null value appears in the respective column. Every row from STAFF and PROJECT is included in the result set.

Listing non-matching rows only

Use where clause with join to find the rows that do not match.

Eg. In case we find To find the name and project number of all employees listed in PROJECT but not listed in STAFF table.

```
Select * from (SELECT a.NAME, a.PROJ,
b.NAME, b.JOB
FROM PROJECT A LEFT OUTER JOIN STAFF B
ON a.NAME = b.NAME) where b.NAME is null
```

Result --

NAME	PROJ	NAME	JOB
Lutz	MA2111	-	-
Walker	MA2112	-	-

6.3.5 Order By, Group By and Having Clause

ORDER BY clause is used to sort the information by the values in one or more columns.

Syntax – select * from <table name> where <if required> order by <column 1> ASC/DESC

Eg. To get the employees in department = 84 ordered by number of years employed in ascending order:

```
SELECT NAME, JOB, YEARS
FROM STAFF
WHERE DEPT = 84
ORDER BY YEARS ASC
```

Note: Specify ORDER BY as the last clause in the entire SELECT statement. Columns named in this clause can be expressions or any column of the table. The column names in the ORDER BY clause do not have to be specified in the select list. We can order rows in ascending or descending order by explicitly specifying either ASC or DESC within the ORDER BY clause. If neither is specified, the rows are automatically ordered in ascending sequence. We can order rows by character values as well as numeric values.

GROUP BY clause is used to group individual rows into combined sets based on the value in one, or more, columns. The column names in the SELECT clause must be either a grouping column or a column function. Column functions return a value for each group defined by the GROUP BY clause. Each group is represented by a single row in the result set.

Syntax – select <expressions, eg: count(*)/max()/min() if required>, <column >, <column 2>, <column 3> .. from <table name> where <if required> group by <column 1> ,<column 2>, <column 3> ..

Eg. To get the maximum salary for each employee in department number=84:

```
SELECT NAME,ID, MAX(SALARY) AS MAXIMUM
FROM STAFF DEPT = 84
GROUP BY NAME, ID
```

Note: There can only be one GROUP BY per SELECT. Multiple select statements in the same query can each have their own GROUP BY.

Every field in the SELECT list must either be specified in the GROUP BY, or must have a column function applied against it.

The result of a simple GROUP BY is always a distinct set of rows, where the unique identifier is whatever fields were grouped on.

When grouping, all null values in the GROUP BY fields are considered equal. There is no guarantee that the rows resulting from a GROUP BY will come back in any particular order, to resolve this we can use an ORDER BY.

To apply a qualifying condition to groups so that query returns a result only for the groups that satisfy the condition, we use **HAVING** clause after the GROUP BY clause. A HAVING clause can contain one or more predicates connected by ANDs and ORs. Each predicate compares a property of the group (such as AVG(SALARY)) with either:

- Another property of the group
For example: HAVING AVG(SALARY) > 2 * MIN(SALARY)
- A constant
For example: HAVING AVG(SALARY) > 20000

Syntax – select <expression, eg: count(*)/max()/min() >, <column >, <column 2>, <column 3> .. from <table name> where <if required> group by <column 1> ,<column 2>, <column 3> .. having <expression > <logical condition>

Eg: To find the maximum and minimum salary of departments with more than 4 employees:

```
SELECT WORKDEPT, MAX(SALARY) AS MAXIMUM, MIN(SALARY) AS MINIMUM
FROM EMPLOYEE
GROUP BY WORKDEPT
HAVING COUNT(*) > 4
```

Note: It is possible (though unusual) for a query to have a HAVING clause but no GROUP BY clause. In this case, database treats the entire table as one group. Because the table is treated as a single group, you can have at most one result row. If the HAVING condition is true for the table as a whole, the selected result (which must consist entirely of column functions) is returned; otherwise no rows are returned.

6.3.6 Union, Except, Intersect Clause

The UNION, EXCEPT, and INTERSECT set operators enable you to combine two or more outer-level queries into a single query. Each query connected by these set operators is executed and the individual results are combined. Each operator produces a different result.

UNION Operator

The UNION operator derives a result table by combining two other result tables (for example TABLE1 and TABLE2) and eliminating any duplicate rows in the tables. When ALL is used with UNION (that is, UNION ALL), duplicate rows are not eliminated. In either case, each row of the derived table is a row from either TABLE1 or TABLE2.

Eg: Following query returns the names of all persons that have a salary greater than \$21, 000, or that have managerial responsibilities and have been working for less than 8 years:

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
UNION
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
ORDER BY ID
```

The database manager combines the results of both queries, eliminates the duplicates, and returns the final result in ascending order.

ID	NAME
10	Sanders
30	Marenghi
100	Plotz
140	Fraye
160	Molinare
240	Daniels
260	Jones

If the column name in the two tables is different, the combined result table does not have names for the corresponding columns. Instead, the columns are numbered in the order in which they appear. So, if we want the result table to be ordered, we have to specify the column number in the ORDER BY clause.

EXCEPT Operator

The EXCEPT operator derives a result table by including all rows that are in TABLE1 but not in TABLE2, and eliminating all duplicate rows. When you use ALL with EXCEPT (EXCEPT ALL), the duplicate rows are not eliminated.

Eg: Following example of the EXCEPT operator, the query returns the names of all persons that earn over \$21,000 but are not a manager and have been there for 8 or more years.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
EXCEPT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Result:

ID	NAME
260	Jones

INTERSECT operator

The INTERSECT operator derives a result table by including only rows that exist in both TABLE1 and TABLE2 and eliminating all duplicate rows. When you use ALL with INTERSECT (INTERSECT ALL), the duplicate rows are not eliminated.

Eg: Following example of the INTERSECT operator, the query returns the name and ID of employees that earn more than \$21,000, have managerial responsibilities and have been working for fewer than 8 years.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
INTERSECT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Result:

ID	NAME
140	Fraye
160	Molinare

Note: All corresponding items in the select-lists of the queries for the operators must be of compatible data type.

Unlike the UNION and INTERSECT operations, the EXCEPT statement is not commutative. This means that "A EXCEPT B" is not the same as "B EXCEPT A".

6.3.7 Case

CASE expressions enable one to do if-then-else type processing inside of SQL statements. There are two types of the expression. In the first kind, each WHEN statement does its own independent checking. In the second kind, all of the WHEN conditions are used to do "equal" checks against a common reference expression. With both flavors, the first WHEN that matches is the one chosen.

Eg 1: The following CASE is of the kind where each WHEN does an equal check against a common expression - in this example, the current value of SEX.

```
SELECT Lastname, sex, CASE sex
  WHEN 'F' THEN 'FEMALE'
  WHEN 'M' THEN 'MALE'
  ELSE NULL
END AS sexx
FROM employee
WHERE lastname LIKE 'J%'
ORDER BY 1
```

Eg 2: Logically the same as the above, but it uses the alternative form of the CASE notation in order to achieve the same result. In this the equal predicate is explicitly stated rather than implied.

```
SELECT lastname, sex,
CASE
  WHEN sex = 'F' THEN 'FEMALE'
  WHEN sex = 'M' THEN 'MALE'
  ELSE NULL
END AS sexx
FROM employee
```

```
WHERE lastname LIKE 'J%'
ORDER BY 1
```

Eg 3: Use CASE to get multiple counts in one pass

```
SELECT COUNT(*) AS tot
,SUM(CASE sex WHEN 'F' THEN 1 ELSE 0 END) AS #f
,SUM(CASE sex WHEN 'M' THEN 1 ELSE 0 END) AS #m
FROM employee
WHERE lastname LIKE 'J%';
```

The CASE expression can also be used in an UPDATE statement to do any one of several alternative updates to a particular field in a single pass of the data:

```
UPDATE staff
SET comm = CASE dept
WHEN 15 THEN comm * 1.1
WHEN 20 THEN comm * 1.2
WHEN 38 THEN
CASE
WHEN years < 5 THEN comm * 1.3
WHEN years >= 5 THEN comm * 1.4
ELSE NULL
END
ELSE comm
END
WHERE comm IS NOT NULL
AND dept < 50;
```

A CASE expression is used to avoid a divide-by-zero error:

```
SELECT c1,c2,CASE c2
WHEN 0 THEN NULL
ELSE c1/c2
END AS c3
FROM table name;
```

Note: If more than one WHEN condition is true, the first one processed that matches is used. If no WHEN matches, the value in the ELSE clause applies. If no WHEN matches and there is no ELSE clause, the result is NULL. There must be at least one non-null result in a CASE statement. Failing that, one of the NULL results must be inside of a CAST expression. All result values must be of the same type.

6.3.8 Sub-query

A sub-query is a special type of full-select that is used to relate one table to another without actually doing a join. When we write an SQL SELECT statement, we can place additional SELECT statements within the WHERE clause. Each additional SELECT starts a subquery. A subquery can then, in turn, include another separate subquery, whose result is substituted into the original subquery's WHERE clause. In addition, a WHERE clause can include subqueries in more than one search condition. The subquery can refer to tables and columns that are different than the ones used in the main query.

Eg: To select the division and location from the ORG table of the employee whose ID in the STAFF table is 280:

```
ID NAME
SELECT DIVISION, LOCATION
FROM ORG
WHERE DEPTNUMB = (SELECT DEPT
FROM STAFF
WHERE ID = 280)
```

When processing a statement, DB2 first determines the result of the subquery. The result from this example's subquery is from STAFF table. The final result is then taken from the row of the ORG table whose DEPTNUMB column has the value of 66.

Eg: To select the employee number and name of employees in department 'A00' with a salary greater than the average salary of the department:

```
SELECT EMPNO, LASTNAME
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
AND SALARY > (SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE WORKDEPT = 'A00')
```

Eg: To select the average salary for every department, the subquery needs to be evaluated once for every department to list all the employees whose salary is higher than the average salary of their department:

```
SELECT E1.EMPNO, E1.LASTNAME, E1.WORKDEPT
FROM EMPLOYEE E1
WHERE SALARY > (SELECT AVG(SALARY)
FROM EMPLOYEE E2
WHERE E2.WORKDEPT = E1.WORKDEPT)
ORDER BY E1.WORKDEPT
```

To write a query with a correlated subquery, use the same basic format of an ordinary outer query with a subquery. However, in the FROM clause of the outer query, just after the table name, place a correlation name. The subquery may then contain column references qualified by the correlation name. We can use joins in a sub-query, list the tables to be joined in the FROM clause, and place the correlation name next to the appropriate table name.

Eg: To select the department's name instead of its number along with employee name whose education level is greater than the average education level in employee table

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
FROM EMPLOYEE E1, DEPARTMENT
WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
AND EDLEVEL > (SELECT AVG(EDLEVEL)
FROM EMPLOYEE E2
WHERE E2.WORKDEPT = E1.WORKDEPT)
```

Suppose that some departments have only a few employees and therefore their average education level may be misleading. You might decide that in order for the average level of education to be a meaningful number to compare an employee against, there must be at least five employees in a department.

So now we have to list the employees whose level of education is higher than the average for that employee's department, and only consider departments with at least five employees. The problem implies another subquery because, for each employee in the outer-level query, the total number of employees in that person's department must be counted:

```
SELECT COUNT(*)
FROM EMPLOYEE E3
WHERE E3.WORKDEPT = E1.WORKDEPT
Only if the count is greater than or equal to 5 is an average to be computed:
SELECT AVG(EDLEVEL)
FROM EMPLOYEE E2
WHERE E2.WORKDEPT = E1.WORKDEPT
AND 5 <= (SELECT COUNT(*)
FROM EMPLOYEE E3
WHERE E3.WORKDEPT = E1.WORKDEPT)
```

And then only those employees whose level of education is greater than the average for that department are included:

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
FROM EMPLOYEE E1, DEPARTMENT
WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
AND EDLEVEL >
(SELECT AVG(EDLEVEL)
FROM EMPLOYEE E2
WHERE E2.WORKDEPT = E1.WORKDEPT
AND 5 <=
(SELECT COUNT(*)
FROM EMPLOYEE E3
WHERE E3.WORKDEPT = E1.WORKDEPT))
```

We can use a subquery to test for the existence of a row that satisfies some condition. In this case, the subquery is linked to the outer-level query by the predicate EXISTS or NOT EXISTS. When we link a subquery to an outer query by an EXISTS predicate, the subquery does not return a value. Rather, the EXISTS predicate is true if the answer set of the subquery contains one or more rows, and false if it contains no rows. The EXISTS predicate is often used with correlated subqueries.

Eg: To select the departments that currently have no entries in the PROJECT table:

```
SELECT DEPTNO, DEPTNAME
FROM DEPARTMENT X
WHERE NOT EXISTS
(SELECT *
FROM PROJECT
WHERE DEPTNO = X.DEPTNO)
ORDER BY DEPTNO
```

Eg: To select the job from staff table whose id is less than 50:

```
SELECT id, job
FROM staff a
WHERE EXISTS
(SELECT *
FROM staff b
WHERE b.id = a.id
AND b.id < 50)
ORDER BY id;
```

We can connect the EXISTS and NOT EXISTS predicates to other predicates by using AND and OR in the WHERE clause of the outer-level query.

We can also use IN in sub-query:

- If any row in the sub-query result matches, the answer is true.
- If no row in the sub-query result matches, the answer is also false.
- If all of the values in the sub-query result are null, the answer is false.

Eg: SELECT *
FROM table1
WHERE t1a IN (SELECT t2a FROM table2);

6.3.9 Xmlquery

In case any column of a table contains xml data, then we can use xmlquery to fetch the required field from that xml file.

Syntax -- Select xmlquery('for \$e in \$d/<Node_name>/<Sub_node_name, if exists> return \$e/<sub_node for which we want to extract data value, eg. BRCNO, STATUS, IEC>/<data_type, eg-number, text>()') passing <column_name, which contains xml data> as "d") from <table_name>

Eg: If column of a table contains xml data like

```
<EBRC>
<BRCNO>HDFC0000003120500014</BRCNO>
<BRCDATE>2012-05-02</BRCDATE>
<STATUS>F</STATUS>
<IEC>0388064510</IEC>
<EXPNAME>WELSPRING UNIVERSAL</EXPNAME>
<IFSC>HDFC0000003</IFSC>
<BILLID>560BC50121250013</BILLID>
<SNO>TKD62652451</SNO>
<SPORT>INTKD6</SPORT>
<SDATE>2011-02-28</SDATE>
<SCC>USD</SCC>
<SVALUE>109846.30</SVALUE>
<RCC>USD</RCC>
<RVALUE>109846.30</RVALUE>
<RDATE>2012-05-02</RDATE>
<RMTBANK>HDFC</RMTBANK>
<RMTCTRY>INDIA</RMTCTRY>
</EBRC>
```

- To select IEC, SVALUE and RVALUE from this xml column "EBRC_XML_1001" SQL will be:

```
Select xmlquery('for $e in $d/EBRC return $e/IEC/number()') passing EBRC_XML_1001 as "d"),
xmlquery('for $e in $d/EBRC return $e/RVALUE/number()') passing EBRC_XML_1001 as "d"),
xmlquery('for $e in $d/EBRC return $e/SVALUE/number()') passing EBRC_XML_1001 as "d")
from EBRC_INFO_1001
```

- To select IEC, SVALUE and RVALUE from this xml column "EBRC_XML_1001" where SDATE=2011-02-28 SQL will be:

```
Select xmlquery('for $e in $d/EBRC return $e/IEC/number()') passing EBRC_XML_1001 as "d"),
xmlquery('for $e in $d/EBRC return $e/RVALUE/number()') passing EBRC_XML_1001 as "d"),
xmlquery('for $e in $d/EBRC return $e/SVALUE/number()') passing EBRC_XML_1001 as "d")
from EBRC_INFO_1001 where XMLEXISTS('$d/EBRC[@SDATE = "2011-02-28"]' passing
EBRC_XML_1001 as "d");
```

6.3.10 Index

An index is a set of pointers that are logically ordered by the values of one or more keys. We can create indexes on table columns to speed up query processing. An index uses pointers to actual data to more efficiently access specific pieces of data.

When creating indexes, we must also take into account the structure of the tables and the type of queries that are most frequently performed on them. For example, columns appearing in the WHERE clause of a frequently issued query is good candidates for indexes.

Similarly, columns that figure in a GROUP BY clause of a frequent query might benefit from the creation of an index, particularly if the number of values used to group the rows is small relative to the number of rows being grouped.

Syntax – create index <index name> on <table name / nick name> (<column name 1>, <column name 2>...)

Eg 1: Create an index named UNIQUE_NAM on the PROJECT table. The purpose of the index is to ensure that there are not two entries in the table with the same value for column project name (PROJNAME).

```
CREATE UNIQUE INDEX UNIQUE_NAM ON PROJECT (PROJNAME)
```

Eg 2: Create an index named JOB_BY_DPT on the EMPLOYEE table. Arrange the index entries in ascending order by job title (JOB) within each department (WORKDEPT).

```
CREATE INDEX JOB_BY_DPT ON EMPLOYEE (WORKDEPT, JOB)
```

Eg 3: The nickname EMPLOYEE references a data source table called CURRENT_EMP. After this nickname was created, an index was defined on CURRENT_EMP. The columns chosen for the index key were WORKDEBT and JOB.

```
CREATE INDEX JOB_BY_DEPT ON EMPLOYEE (WORKDEPT, JOB)
```

To remove or delete indexes, you can use the DROP INDEX <index name> command.

Guidelines and considerations when designing indexes:

- Although the order of the columns making up an index key does not make a difference to index key creation, it might make a difference to the optimizer when it is deciding whether or not to use an index.
For example, if a query has an ORDER BY col1,col2 clause, an index created on (col1,col2) could be used, but an index created on (col2,col1) will be of no help. Similarly, if the query specified a condition such as where col1 >= 50 and col1 <= 100 or where col1=74, then an index on (col1) could be helpful, but an index on (col2,col1) is far less helpful.

Note: Whenever possible, order the columns in an index from the most distinct to the least distinct. This provides the best performance.

- Any number of indexes can be defined on a particular table, to a maximum of 32 767, and they can have a beneficial effect on the performance of queries. Large index keys can also slow down processing of requests. Therefore, use indexes only where a clear advantage for frequent access exists.
- Column data which is not part of the unique index key but which is to be stored or maintained in the index is called an include column. Include columns can be specified for unique indexes only. When creating an index with include columns, only the unique key columns are sorted and considered for uniqueness. The use of include columns can enable index only access for data retrieval, thus improving performance.
- If the table being indexed is empty, an index is still created, but no index entries are made until the table is loaded or rows are inserted.

Difference in DB2 and Oracle Data Analysis



7. *Difference in DB2 and Oracle Data Analysis*

Logic of SQL remains same for both DB2 and ORACLE, only syntax differs for some of the SQL commands/functions.

7.1 *Syntax difference of common commands/functions*

- Create table in Db2:
Syntax -- create table <table name1> as select * from <table name2> with no data
- Create table in Oracle:
Syntax –create table <table name1> as select * from <table name2> where 1<>1
- Limiting result sets in Db2:
Syntax – select * from table where<...> fetch first n rows only
- Limiting result sets in Oracle:
Syntax – select * from table where rownumber<=n
- Concatenation sets in Db2:
Syntax – select column 1 || column 2 from table
- Concatenation result sets in Oracle:
Syntax – select concat(column 1, column2) from table
- To see list of all tables in a schema in Db2:
Syntax – select * from sysibm.systables
- To see list of all tables in a schema in Oracle:
Syntax – select * from tab
- Dummy table in Db2: sysibm.sysdummy1, it is used to check any logical condition/date type
Eg: select current date from sysibm.sysdummy1
- Dummy table in Oracle: dual, it is used to check any logical condition/date type
Eg: select sysdate from dual

Connect two database in DB2 for data analysis

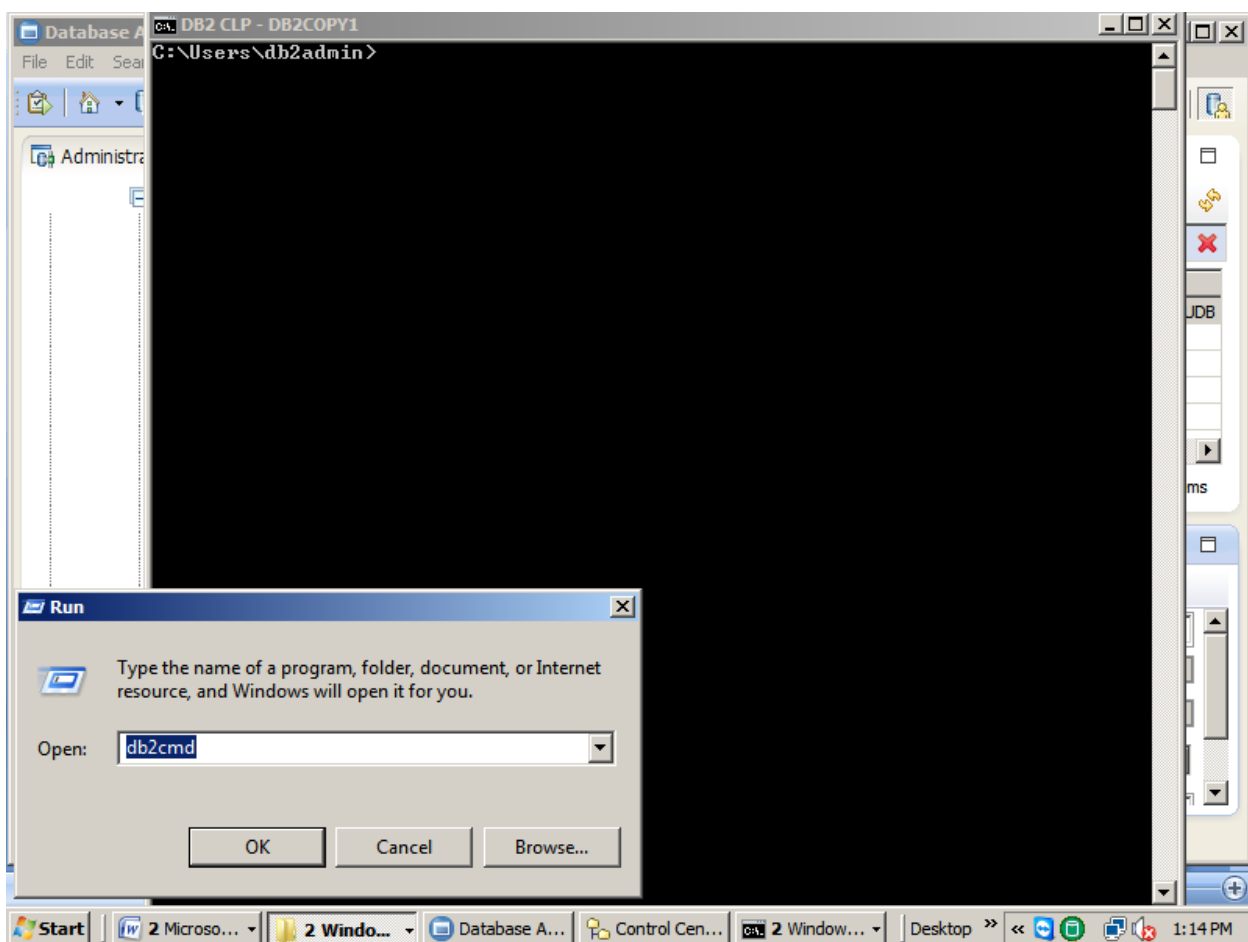


8. *Connect two database in DB2 for data analysis*

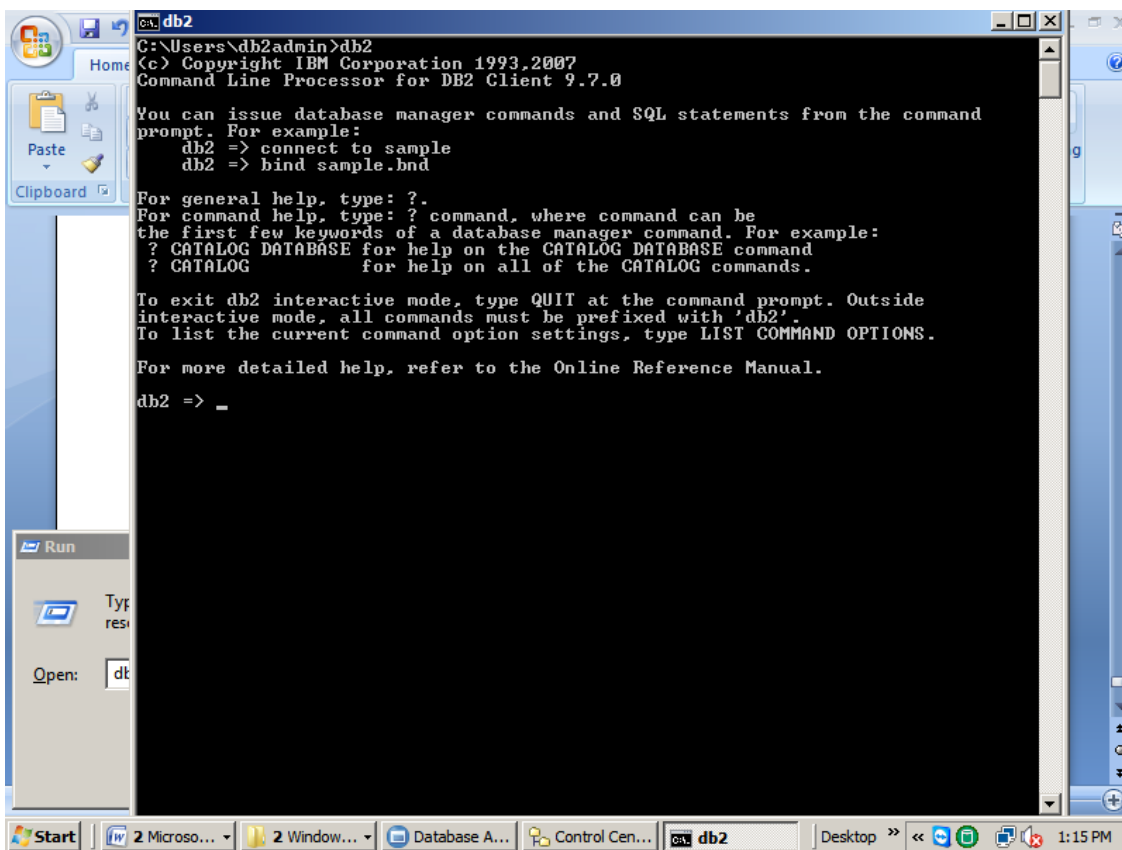
We can connect 2 or more db2 database and analyze tables present in database with tables in another db2 database. If we want to analyze a table in DGFTMAIN with a table in DGFTRLA we need to create a federated database connection from DGFTRLA database to DGFTMAIN database.

Perform following steps to create a federated connection from one DB to another. Eg. from DBDGFTRLA to DGFTMAIN

1. Click start then run and type db2cmd to open db2 command prompt window.



2. Type db2 and press enter



```
C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
  db2 => connect to sample
  db2 => bind sample.bnd

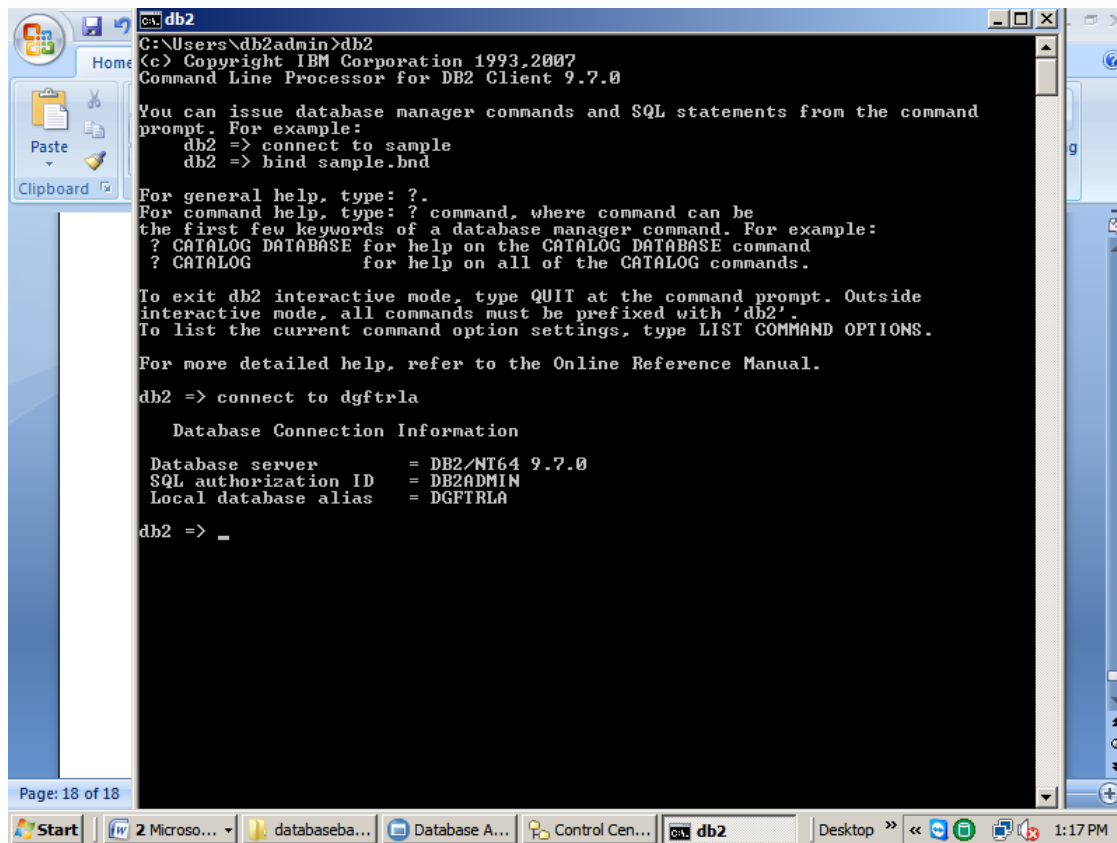
For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => _
```

3. Run following command – update dbm cfg using federated yes;
And then connect to dgftrl and press enter; to connect the database (DGFTRLA) from where you want to connect analyze tables of another database (DGFTMAIN, EBRC).



```
C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
    ? CATALOG DATABASE for help on the CATALOG DATABASE command
    ? CATALOG           for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftrla

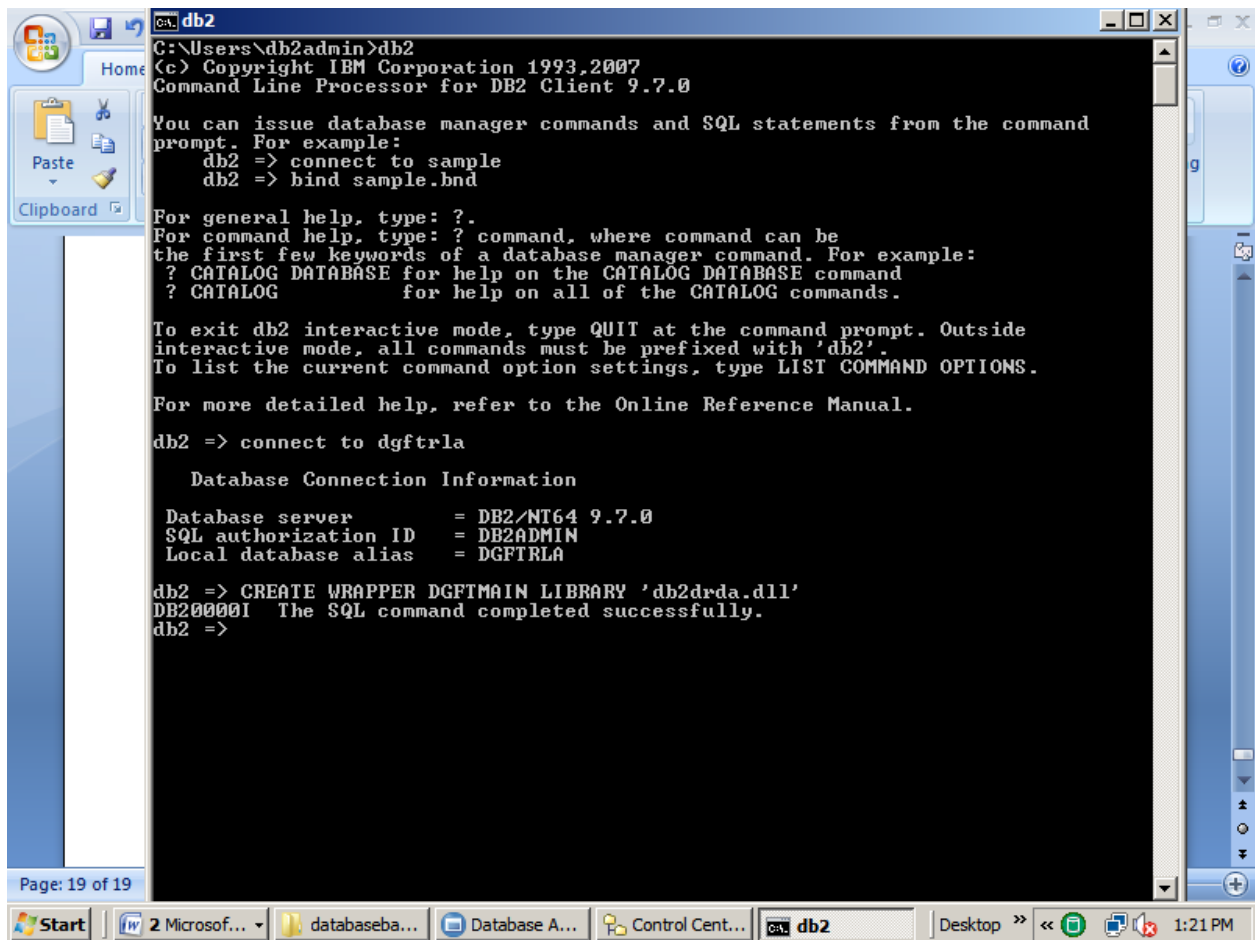
    Database Connection Information

    Database server      = DB2/NT64 9.7.0
    SQL authorization ID = DB2ADMIN
    Local database alias = DGFTRLA

db2 => _
```

4. Run command to create a wrapper -- CREATE WRAPPER <wrapper name> LIBRARY 'db2drda.dll'.

Eg. CREATE WRAPPER DGFTMAIN LIBRARY 'db2drda.dll'



```

C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
  db2 => connect to sample
  db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
  ? CATALOG DATABASE for help on the CATALOG DATABASE command
  ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftrla

      Database Connection Information

Database server      = DB2/NT64 9.7.0
SQL authorization ID = DB2ADMIN
Local database alias = DGFTRLA

db2 => CREATE WRAPPER DGFTMAIN LIBRARY 'db2drda.dll'
DB20000I The SQL command completed successfully.
db2 =>

```

5. Run command for server mapping -- CREATE SERVER <server name> TYPE DB2/UDB VERSION '<db2 version>' WRAPPER <wrapper name> AUTHID "<user id of the database to be connected>" PASSWORD "<password of the database to be connected>" OPTIONS(ADD DBNAME '<database name>');

Eg. -- CREATE SERVER DGFTMAIN TYPE DB2/UDB VERSION '9.7' WRAPPER DGFTMAIN AUTHID "db2admin" PASSWORD "db2admin" OPTIONS(ADD DBNAME 'DGFTMAIN');

```

C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
  db2 => connect to sample
  db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
  ? CATALOG DATABASE for help on the CATALOG DATABASE command
  ? CATALOG           for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.
For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftrla

      Database Connection Information

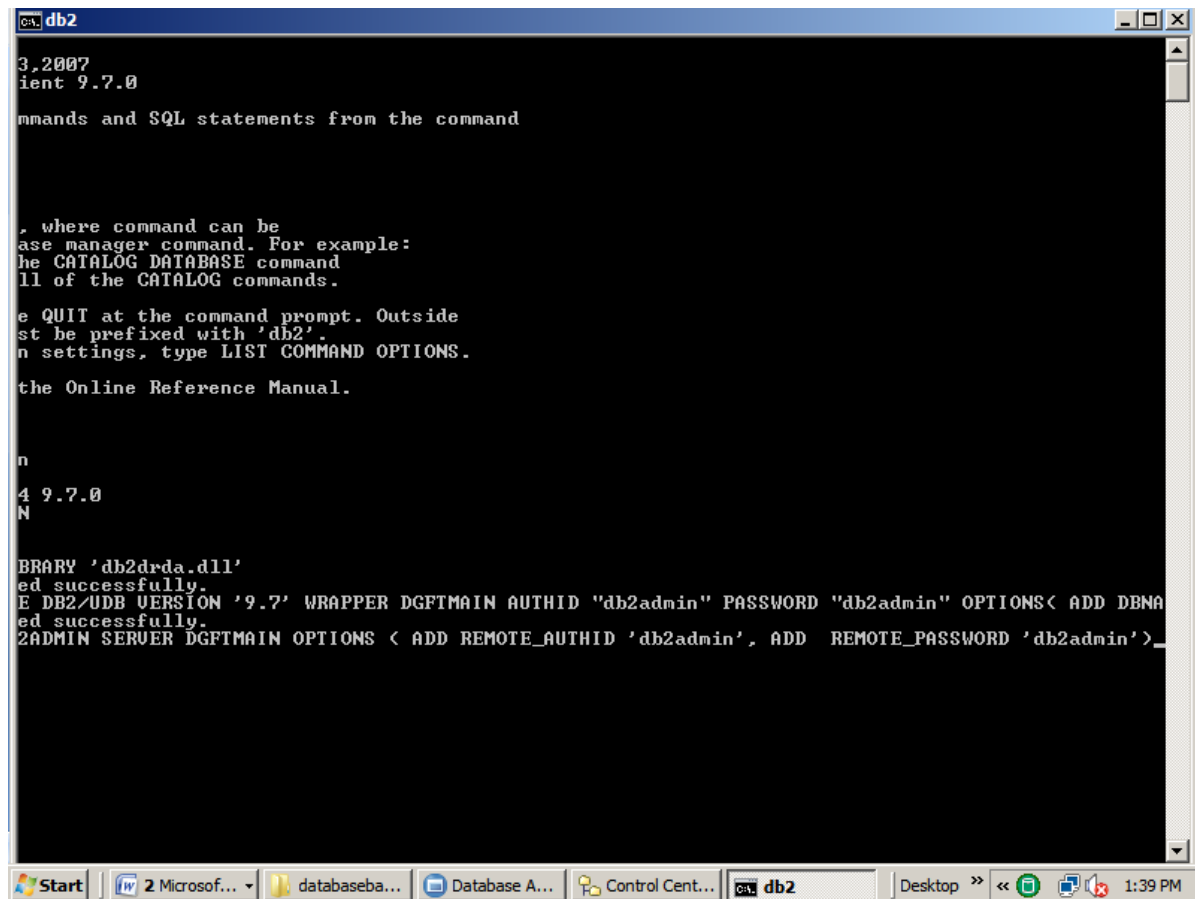
Database server      = DB2/NT64 9.7.0
SQL authorization ID = DB2ADMIN
Local database alias = DGFTRLA

db2 => CREATE WRAPPER DGFTMAIN LIBRARY 'db2drda.dll'
DB20000I The SQL command completed successfully.
db2 => CREATE SERVER DGFTMAIN TYPE DB2/UDB VERSION '9.7' WRAPPER DGFTMAIN AUTHID "db2admin" PASS
DB20000I The SQL command completed successfully.
db2 =>

```

6. Run command for user mapping -- CREATE USER MAPPING FOR <user name of the database from where your are connecting and will use to connect other database. For eg. in this case dgftrla is the database and we use db2admin user to connect dgftmain database> SERVER <server name> OPTIONS (ADD REMOTE_AUTHID '< user id of the database to be connected >', ADD REMOTE_PASSWORD '<password of the database to be connected >');

Eg. -- CREATE USER MAPPING FOR DB2ADMIN SERVER DGFTMAIN OPTIONS (ADD REMOTE_AUTHID 'db2admin', ADD REMOTE_PASSWORD 'db2admin')



```
db2
3.2007
level 9.7.0

Commands and SQL statements from the command

, where command can be
ase manager command. For example:
he CATALOG DATABASE command
ll of the CATALOG commands.

e QUIT at the command prompt. Outside
st be prefixed with 'db2'.
n settings, type LIST COMMAND OPTIONS.
the Online Reference Manual.

n
4 9.7.0
N

BRARY 'db2drda.dll'
ed successfully.
E DB2/UDB VERSION '9.7' WRAPPER DGFTMAIN AUTHID "db2admin" PASSWORD "db2admin" OPTIONS< ADD DBNA
ed successfully.
2ADMIN SERVER DGFTMAIN OPTIONS < ADD REMOTE_AUTHID 'db2admin', ADD REMOTE_PASSWORD 'db2admin'>_
```

```

C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
  db2 => connect to sample
  db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftrl

    Database Connection Information

Database server      = DB2/NT64 9.7.0
SQL authorization ID = DB2ADMIN
Local database alias = DGFTRLA

db2 => CREATE WRAPPER DGFTMAIN LIBRARY 'db2drda.dll'
DB20000I The SQL command completed successfully.
db2 => CREATE SERVER DGFTMAIN TYPE DB2/UDB VERSION '9.7' WRAPPER DGFTMAIN AUTHID "db2admin" PASS
DB20000I The SQL command completed successfully.
db2 => CREATE USER MAPPING FOR DB2ADMIN SERVER DGFTMAIN OPTIONS ( ADD REMOTE_AUTHID 'db2admin',
DB20000I The SQL command completed successfully.
db2 => _

```

7. Run command to create nickname -- CREATE NICKNAME <schema name in which you want to create nickname>.<nick name which we want to use in this case dgftrl> FOR <database name which we want to connect in this case dgftmain>.<schema name where table exist in dgftmain database>.<table name in dgftmain which we want to analyze in dgftrl>;

Eg. -- CREATE NICKNAME DB2ADMIN.CAT_MAST_109 FOR DGFTMAIN.COMMON.CAT_MAST_109;

```

C:\Users\db2admin>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
db2 => connect to sample
db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to dgftrla

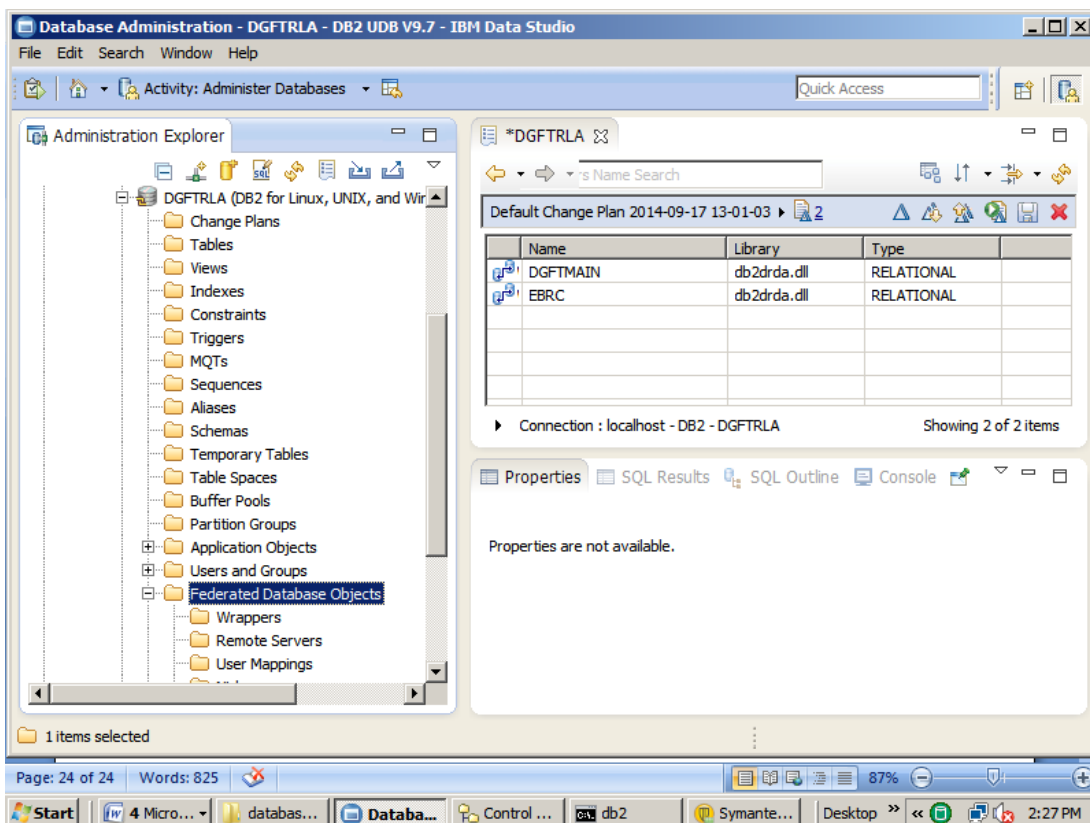
Database Connection Information

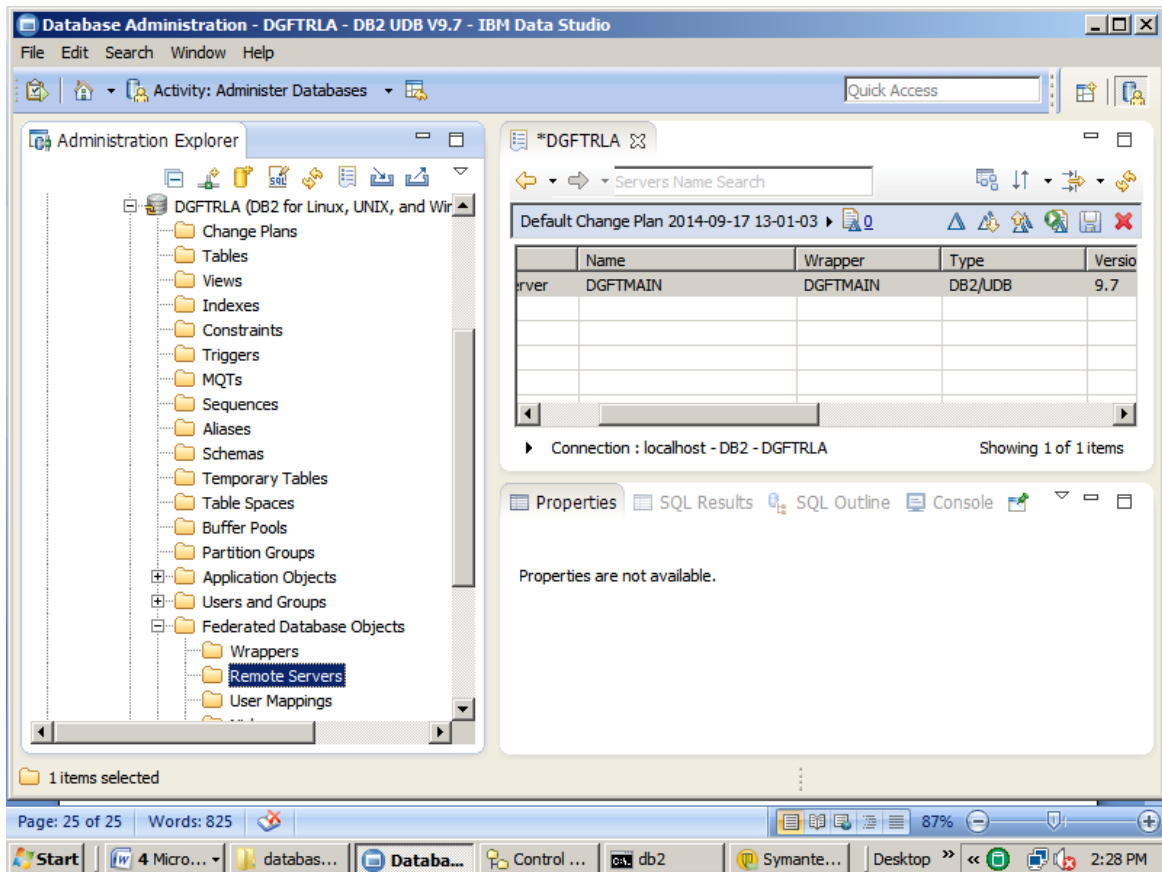
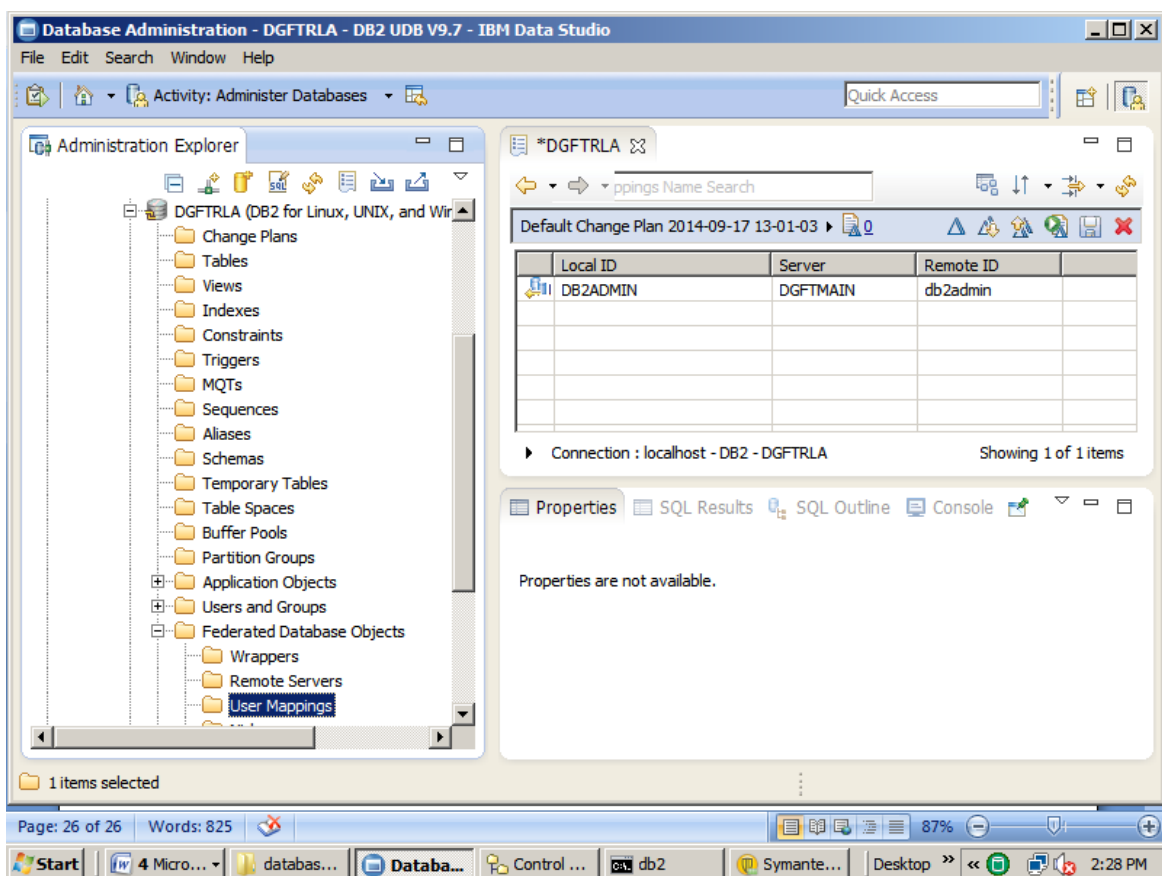
Database server      = DB2/NT64 9.7.0
SQL authorization ID = DB2ADMIN
Local database alias = DGFTRLA

db2 => CREATE WRAPPER DGFMAIN LIBRARY 'db2drda.dll'
DB20000I The SQL command completed successfully.
db2 => CREATE SERVER DGFMAIN TYPE DB2/UDF VERSION '9.7' WRAPPER DGFMAIN AUTHID 'db2admin' PASS
DB20000I The SQL command completed successfully.
db2 => CREATE USER MAPPING FOR DB2ADMIN SERVER DGFMAIN OPTIONS ( ADD REMOTE_AUTHID 'db2admin',
DB20000I The SQL command completed successfully.
db2 => CREATE NICKNAME DB2ADMIN.CAT_MAST_109 FOR DGFMAIN.COMMON.CAT_MAST_109
DB20000I The SQL command completed successfully.
db2 =>

```

We can verify above steps from data studio client by clicking **wrappers, remote servers, user mappings and nicknames** under **Federated database objects** of a database (DGFTRLA in this case).



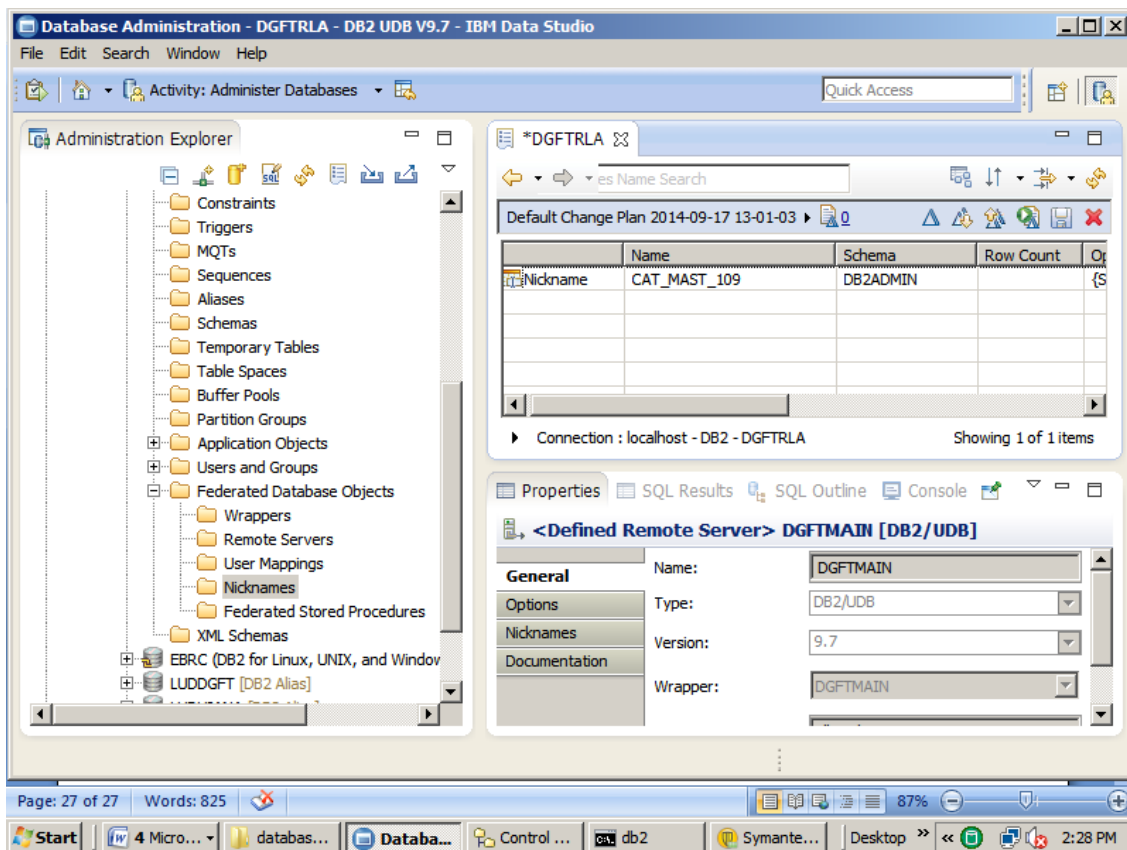


Note: There is no need to create nick names if DB2 version 10.5 is installed. We can directly analyze the table by giving the *<database name>.<schema name>.<table name>*

Eg. -- if we want to select/analyze a table COMMON.CAT_MAST_109 in DGFTMAIN database from DGFTRLA database, we can run command like --

```
select * from DGFTMAIN.COMMON.CAT_MAST_109
```

```
SELECT *
FROM DGFTMAIN.DB2ADMIN.CAT_MAST_109 AS A, LICM.LIC_MAST_1500 AS B
WHERE A.CAT_CODE_109 = B.LIC_CATG_1500
```



Analyzing nickname created in dgftrla database for a table in dgftmain database.

Database Administration - .sqlexecutor_project/CAT_MAST_109.sql - IBM Data Studio

Activity: Administer Databases

Administration Explorer

- DGFTMAIN [DB2 Alias]
 - DGFTRLA (DB2 for Linux, UNIX, and Windows)
 - Change Plans
 - Tables
 - Views
 - Indexes
 - Constraints
 - Triggers
 - MQTs
 - Sequences
 - Aliases
 - Schemas
 - Temporary Tables
 - Table Spaces
 - Buffer Pools
 - Partition Groups
 - Application Objects
 - Users and Groups
 - Federated Database Objects
 - XML Schemas
 - EBRC [DB2 Alias]

SQL Editor

```
SELECT * FROM "DB2ADMIN"."CAT_MAST_109"
```

Properties | SQL Results | SQL Outline | Console

Type query expression here

Status	Operation	Result1
✓	Succeed select * from depb.dept	
✓	Succeed select * from LICM.lic_m	
✓	Succeed select * from common.LI	
✓	Succeed select * from depb.dept	
✓	Succeed select * from depb.dept	
✓	Succeed SELECT FILE_NUM_2102	
✓	Succeed SELECT * FROM "DB2ADM	

CAT_CODE_109	CAT_NAME_10
1 0	None
2 131	Profile Updati..
3 190	Corr. Against..
4 127	Green Card ..
5 163	Amendment (..
6 164	Amendment (..

Total 86 records shown

Page: 28 of 28 Words: 848 87%

Start 4 Microsoft Office Word databasestructures Database Administra... Desktop 2:35 PM

Analyzing nickname by joining it with a table in dgftrla database.

Database Administration - .sqlexecutor_project/CAT_MAST_109.sql - IBM Data Studio

Activity: Administer Databases

Administration Explorer

- DGFTMAIN [DB2 Alias]
 - DGFTRLA (DB2 for Linux, UNIX, and Windows)
 - Change Plans
 - Tables
 - Views
 - Indexes
 - Constraints
 - Triggers
 - MQTs
 - Sequences
 - Aliases
 - Schemas
 - Temporary Tables
 - Table Spaces
 - Buffer Pools
 - Partition Groups
 - Application Objects
 - Users and Groups
 - Federated Database Objects
 - XML Schemas
 - EBRC [DB2 Alias]

SQL Editor

```
SELECT *
FROM DB2ADMIN.CAT_MAST_109 AS A, LICM.LIC_MAST_1500 AS B
WHERE A.CAT_CODE_109 = B.LIC_CATG_1500
FETCH FIRST 10 ROWS ONLY
```

Properties | SQL Results | SQL Outline | Console

Type query expression here

Status	Operation	Date
✓	Succeed select * from depb.depb_...	9/16
✓	Succeed select * from LICM.lic_mas...	9/16
✓	Succeed select * from common.LIC...	9/16
✓	Succeed select * from depb.depb_...	9/16
✓	Succeed select * from depb.depb_i...	9/16
✓	Succeed SELECT FILE_NUM_2102, ...	9/17
✓	Succeed SELECT * FROM "DB2ADM...	9/17
✓	Succeed SELECT * FROM "DB2ADM...	9/17

CAT_CODE_109	CAT_NAME_109	TIM_LIN
1 3	Hotels	...
2 3	Hotels	...
3 3	Hotels	...
4 3	Hotels	...
5 3	Hotels	...
6 3	Hotels	...
7 3	Hotels	...

Total 10 records shown

Displayed 467 of 467 results: 320 succeeded, 138 failed, 8 terminated, 1 warning, 0 critical error

Page: 1 of 18 Words: 1/3,384 87%

Start 4 Microsoft Office Word databasestructures Database Administra... Desktop 2:39 PM

Connect Oracle database with DB2 database for data analysis



9. *Connect Oracle database with Db2 database for data analysis*

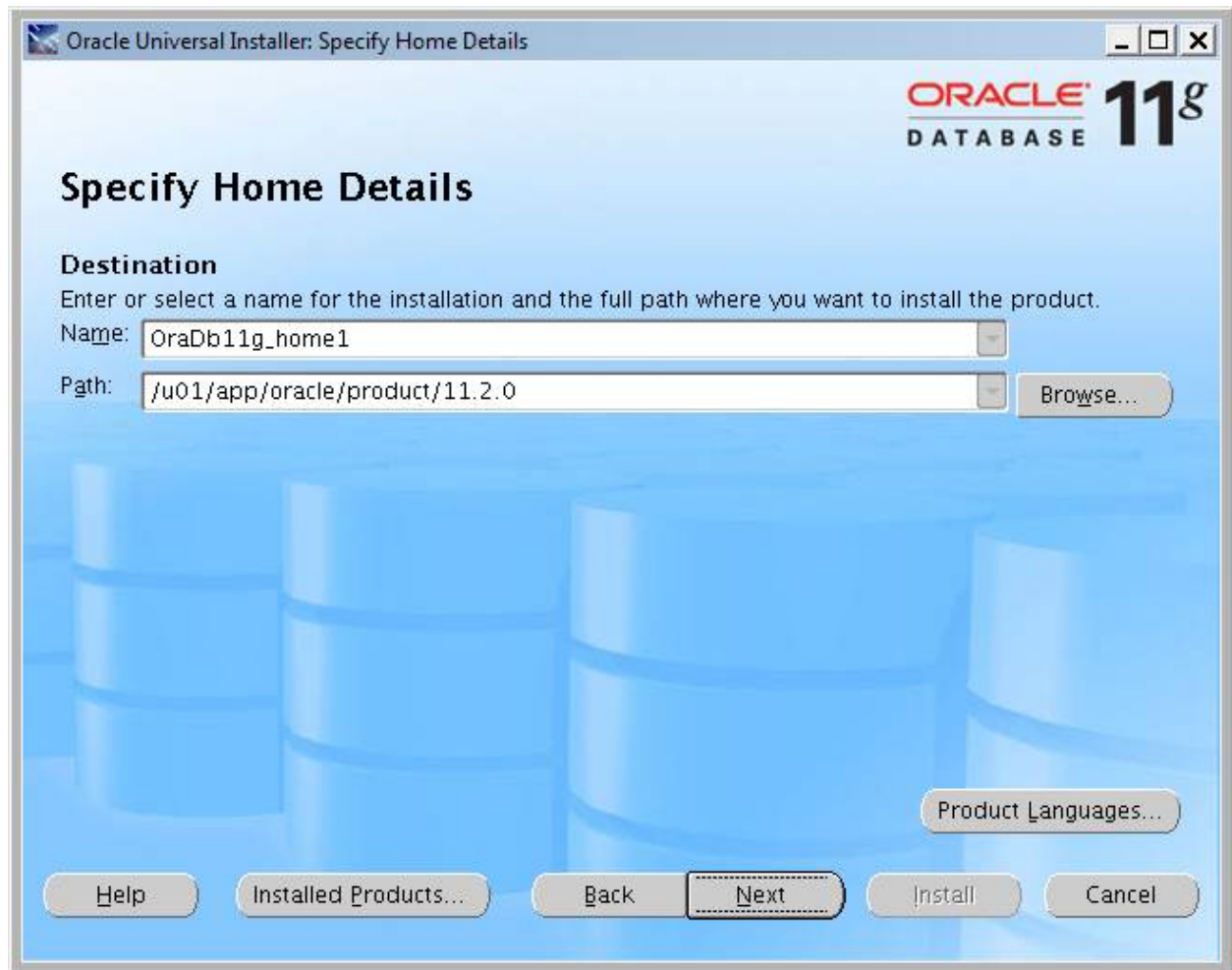
Following are the steps to connect oracle database with DB2 database:

Prerequisites:

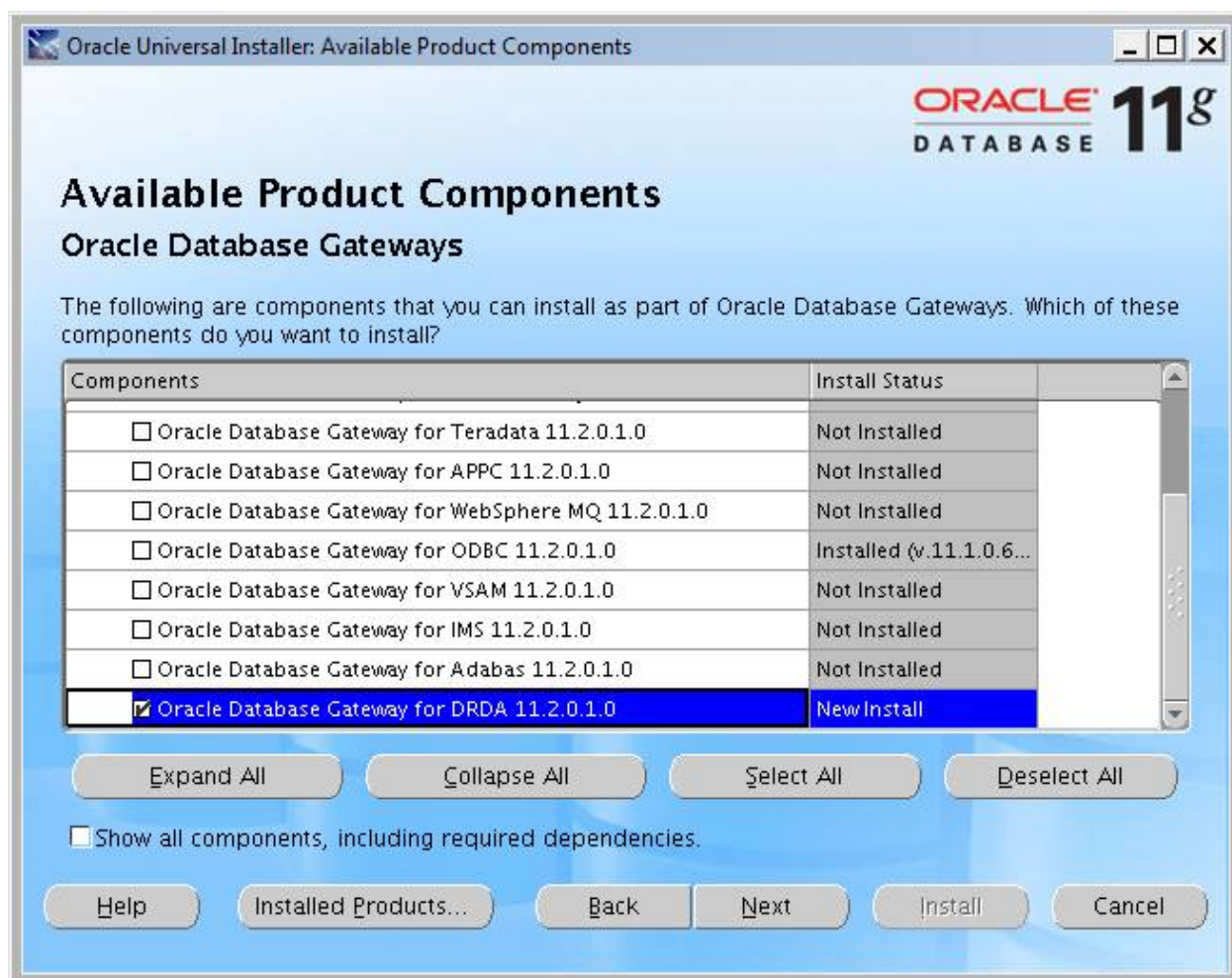
- Oracle 11g on windows
- Db2 9.7 or 10.5C on windows
- Oracle gateway software -- “win64_11gR2_gateways.zip” download it from (<http://www.oracle.com/technetwork/database/enterprise-edition/downloads/112010-win64soft-094461.html>) based on the oracle data base version and windows O/S (32 bit/64 bit).
- Both the servers should be on same network

a) Step One : Unzip the file and Install Oracle gateway driver by clicking the setup.exe





Here provide path as path of your oracle home -- C:\ORACLE\product\11.2.0\



Here checked the Oracle database gateway for DRDA.

Oracle Universal Installer: Oracle Database Gateway for DRDA

ORACLE 11^g
DATABASE

Oracle Database Gateway for DRDA

Please provide the following installation details for the DB2 UDB database.

DB2 UDB Server Host Name or IP Address:

DB2 UDB DRDA Server Listener Port Number:

DB2 UDB Database Name:

DB2 UDB Target Type:

Help Installed Products... Back Next Install Cancel

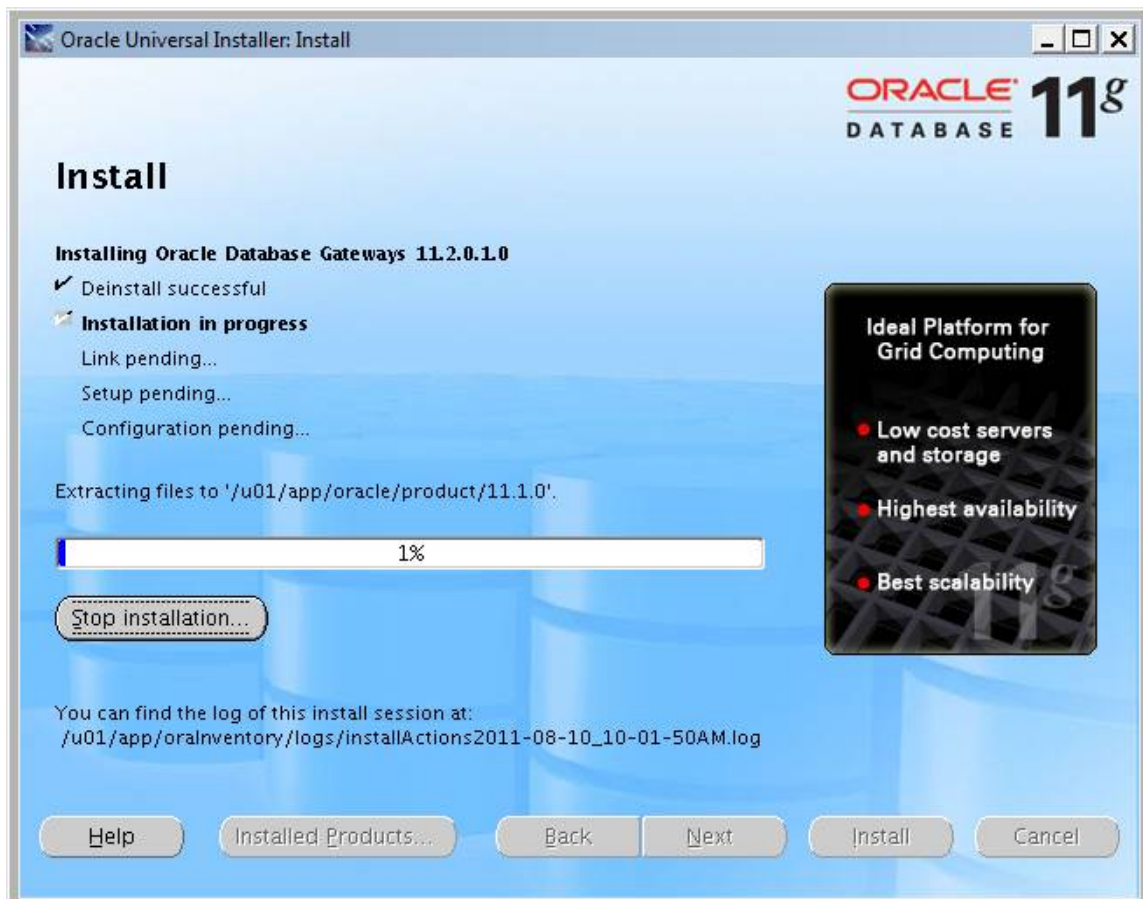
Here Provide DB2 database credentials, which you want to connect and select target as LUW for db2 database on windows.

Host: hostname or IP on which DB2 database is installed

Port: port on which db2 service is running, by default it is 50000

Database Name: DGFTMAIN

Target: LUW





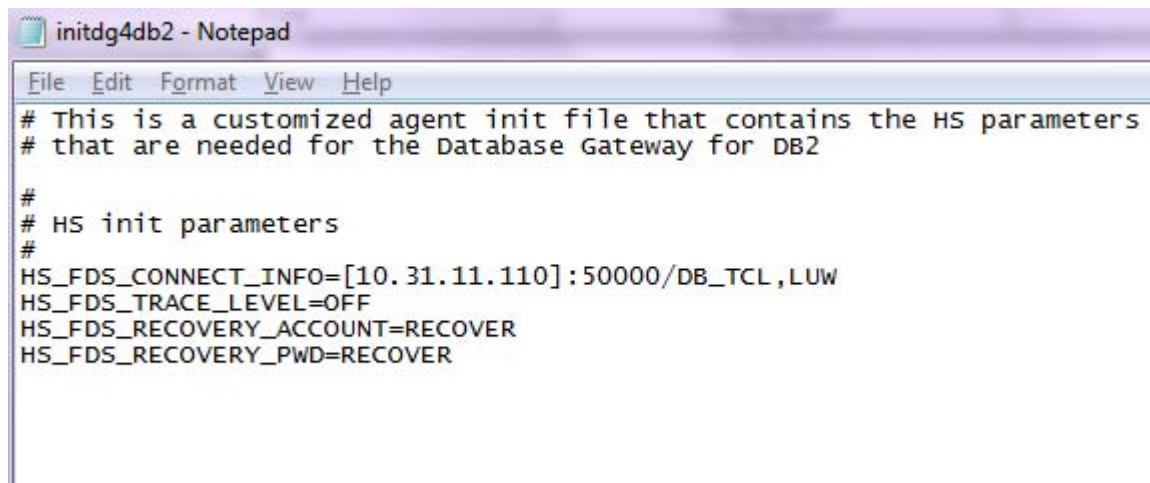
After completing the above step verify if the below entry created in **initdg4db2** file in the following directory `\dg4db2\admin\` in oracle home directory.

product ► 11.2.0 ► tg_2 ► dg4db2 ► admin			
Name	Date modified	Type	Size
initdg4db2	9/16/2014 12:07 PM	ORA File	1 KB
listener.ora.sample	9/15/2014 6:20 PM	SAMPLE File	1 KB
tnsnames.ora.sample	9/15/2014 6:20 PM	SAMPLE File	1 KB
DDOC510A.bnd	3/3/2009 3:09 PM	BND File	10 KB
DDOC510B.bnd	3/3/2009 3:09 PM	BND File	12 KB

```

HS_FDS_CONNECT_INFO=[10.31.11.110]:50000/DGFTMAIN,LUW
HS_FDS_TRACE_LEVEL=OFF
HS_FDS_RECOVERY_ACCOUNT=RECOVER
HS_FDS_RECOVERY_PWD=RECOVER

```



```

initdg4db2 - Notepad
File Edit Format View Help
# This is a customized agent init file that contains the HS parameters
# that are needed for the Database Gateway for DB2

#
# HS init parameters
#
HS_FDS_CONNECT_INFO=[10.31.11.110]:50000/DB_TCL,LUW
HS_FDS_TRACE_LEVEL=OFF
HS_FDS_RECOVERY_ACCOUNT=RECOVER
HS_FDS_RECOVERY_PWD=RECOVER

```

b) Step two : Configure Listener

Now go to the oracle home and then **NETWORK\ADMIN**

So the full path in this case was C:\ORACLE\product\11.2.0\tg_2\NETWORK\ADMIN

	SAMPLE	7/28/2014 12:19 PM	File folder	
	listener	9/16/2014 1:20 PM	ORA File	1 KB
	sqlnet	9/16/2014 1:29 PM	ORA File	1 KB
	tnsnames	9/16/2014 1:20 PM	ORA File	3 KB

Open the listener.ora file.

Then add the following :

SID_LIST_LISTENER =

(SID_LIST =

(SID_DESC =

(SID_NAME=dg4db2)

(ORACLE_HOME=C:\ORACLE\product\11.2.0\)

(PROGRAM=dg4db2)

)

)

LISTENER =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 1RK8102) (PORT = 1521))

)

SID_LIST_LISTENER =

(SID_LIST =

(SID_DESC =

(SID_NAME=dg4db2)

(ORACLE_HOME=C:\ORACLE\product\11.2.0\)

(PROGRAM=dg4db2)

)

)

ADR_BASE_LISTENER = C:\ORACLE

After configuring the listener check listener status to verify its working fine.

```

C:\Users>lsnrctl

LSNRCTL for 64-bit Windows: Version 11.2.0.1.0 - Production on 17-SEP-2014 13:40:34

Copyright (c) 1991, 2010, Oracle. All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL>

C:\windows\system32\cmd.exe - lsnrctl

LSNRCTL> start LISTENER
TNS-01106: Listener using listener name LISTENER has already been started
LSNRCTL> status LISTENER
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=1RK8102)(PORT=1521)))
STATUS of the LISTENER
-----
Alias                     LISTENER
Version                   TNSLSNR for 64-bit Windows: Version 11.2.0.1.0 - Production
Start Date                17-SEP-2014 13:39:23
Uptime                    0 days 0 hr. 2 min. 6 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   C:\ORACLE\product\11.2.0\dbhome_1\network\admin\listener.ora
Listener Log File         c:\oracle\diag\tnslsnr\1RK8102\listener>alert\log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=1RK8102.IN.nam.ad.pwcinternal.com)(PORT=1521)))
Services Summary...
Service "dg4db2" has 1 instance(s).
  Instance "dg4db2", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
LSNRCTL>

```

c) Step Three: Configure tnsnames.ora

Now go to the oracle home and then **NETWORK\ADMIN**

In my case home was C:\ORACLE\product\11.2.0\

So the full path for my case was C:\ORACLE\product\11.2.0\tg_2\NETWORK\ADMIN

 SAMPLE	7/28/2014 12:19 PM	File folder	
 listener	9/16/2014 1:20 PM	ORA File	1 KB
 sqlnet	9/16/2014 1:29 PM	ORA File	1 KB
 tnsnames	9/16/2014 1:20 PM	ORA File	3 KB

Open the tnsnames.ora file.

Then add the following :

dg4db2 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST=Localhost)(PORT = 1521))

(CONNECT_DATA = (SID = dg4db2))

(HS = OK)

)

```

File Edit Format View Help
# tnsnames.ora Network Configuration File: C:\ORACLE\product\11.2.0\dbhome_1\network\admin\tnsnames.ora
# Generated by oracle configuration tools.

dg4db2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST=localhost)(PORT = 1521))
    (CONNECT_DATA = (SID = dg4db2))
    (HS = OK)
  )

```

Here the host name will be oracle host name, in my case it was localhost
After configuring the tnsnames do tnsping to check if its working fine.

```

C:\Users>tnsping dg4db2

TNS Ping Utility for 64-bit Windows: Version 11.2.0.1.0 - Production on 17-SEP-2014 13:47:09

Copyright (c) 1997, 2010, Oracle. All rights reserved.

Used parameter files:
C:\ORACLE\product\11.2.0\tg_2\network\admin\sqlnet.ora

Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST=1RK8102)(PORT = 1521)) (CONNECT_DATA = (SID = dg4db2)) (HS = OK))
OK (0 msec)

C:\Users>

```

- d) Step Four : Create Oracle database link by executing following command:

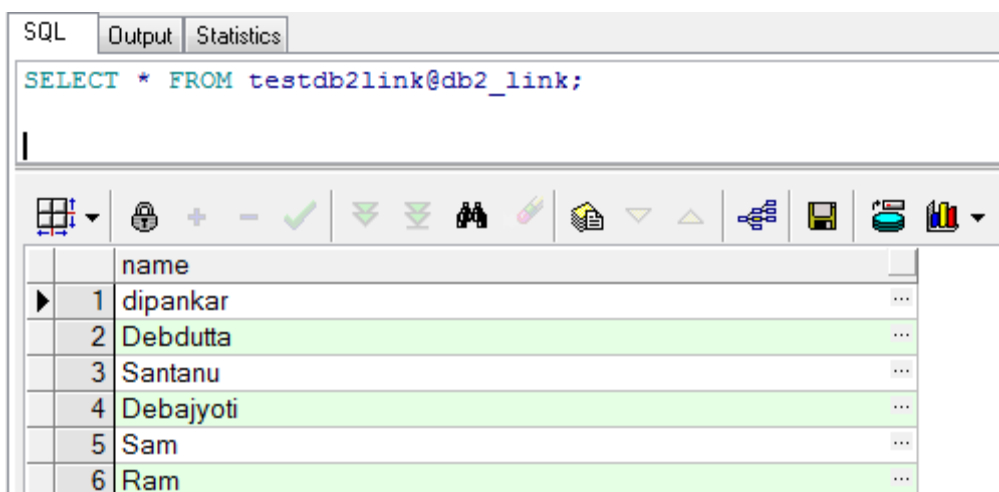
Create public database link db2_link3 connect to *ashish* identified by "*password*" using 'dg4db2';

where ashish is DB2 database server userID and password is DB2 server password and 'dg4db2' is tns entry created in the previous step.

- e) Step Five : Test Connection

Select from some DB2 table from oracle to verify if it is working fine,

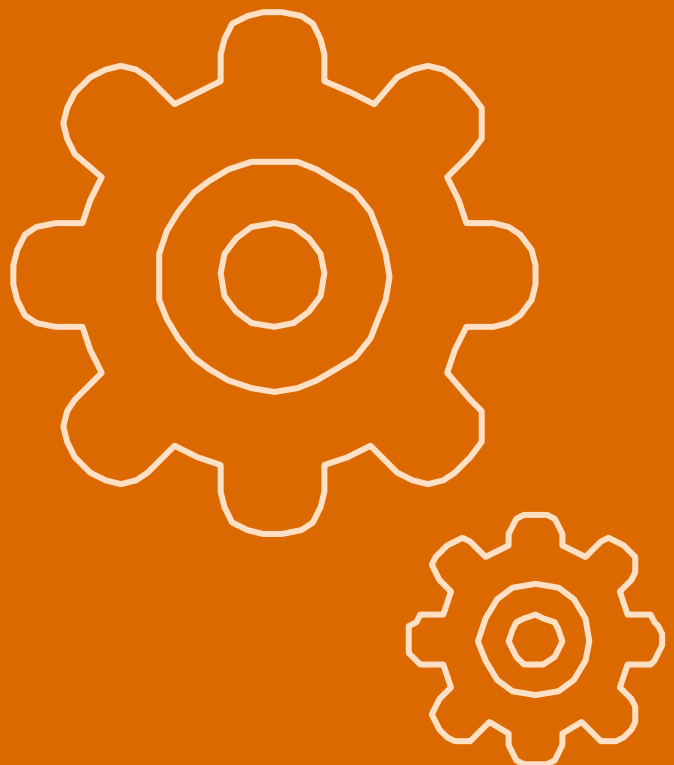
eg: select * from testdb2link@db2_link3;



	name
1	dipankar
2	Debdutta
3	Santanu
4	Debajyoti
5	Sam
6	Ram

Here testdb2link is the test table in DB2 database.

Troubleshooting in DB2



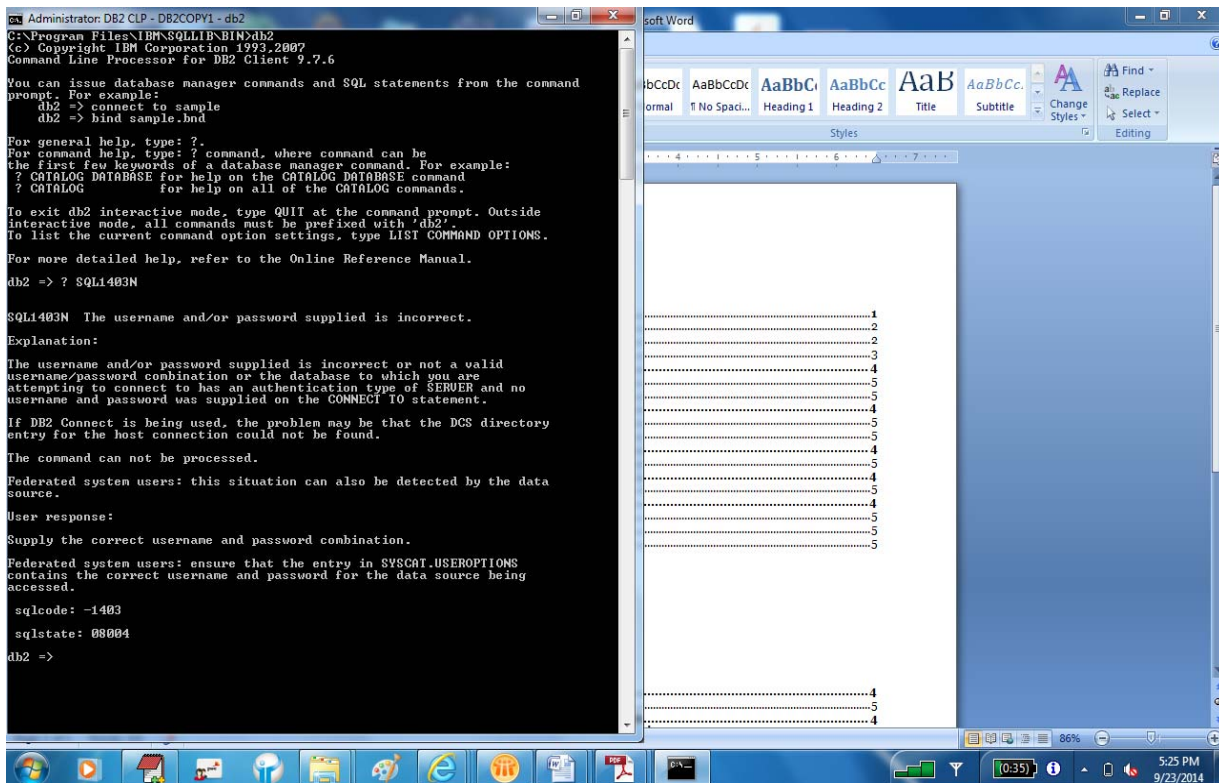
10. Troubleshooting in DB2

Whenever you make a mistake typing in any of the examples, or if an error occurs during execution of an SQL statement, the database manager returns an error message. The error message consists of a message identifier, a brief explanation, and an SQLSTATE.

SQLSTATE errors are error codes common to the DB2 family of products. For example, if the user ID or password had been incorrect in the CONNECT statement, the database manager will return a message identifier of SQL1403N and an SQLSTATE of 08004. The message is as follows: SQL1403N The username and/or password supplied is incorrect. SQLSTATE=08004

We can get information about resolution of the error message by typing a question mark (?) then the message identifier or the SQLSTATE in db2cmd.

```
? SQL1403N
OR
? SQL1403
OR
? 08004
```



General issues on information security in RDBMS



11. General issues on information security in RDBMS

There are three main objectives when designing a secure database application:

Secrecy: Information should not be disclosed to unauthorized users. For Example, a student should not be allowed to examine other students' grades.

Integrity: Only authorized users should be allowed to modify data. For Example, students may be allowed to see their grades, yet not allowed to modify them.

Availability: Authorized users should not be denied access. For example, An instructor who wishes to change a grade should be allowed to do so.

Security risks to database systems include, for example:

- Unauthorized or unintended activity or misuse by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers (e.g. inappropriate access to sensitive data, metadata or functions within databases, or inappropriate changes to the database programs, structures or security configurations);
- Unavailability of a sophisticated logging and a monitoring system resulting to inability of the user to detect unauthorized access and analyze historical records of user access which will be able to identify system misuse by revealing the usage pattern of the database.
- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services;
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended;
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic breakdowns/equipment failures and obsolescence leading to loss of critical data;
- Unavailability of a properly configured and tested disaster recovery plan leading to an increased downtime and loss of data;
- Design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities (e.g. unauthorized privilege escalation), data loss/corruption, performance degradation etc.;
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc.

Appendix - Sample Queries



12. Appendix - Sample Queries

Sample queries to test following business conditions:

1. **Query:** To extract records with port of export used when export was made to China. Since export to china is banned from 3 inland ports, the o/p of the query will give us the records where export was done to China using any of the bannedland port.

Further we can check if any of the above shipping bill export was actually attached in a particular scheme and its licence was not cancelled.

SQL: create table shb_atrch_dtch as (select a.*,b.dt_count, b.act_code_9101_dt, c.*, d.* from (select count(*) at_count, act_code_9101 act_code_9101_at,shbi_num_9101,ecom_ref_9101,sch_name_9101 from edi.SHB_UTIL_9101 where act_code_9101='ATCH' group by act_code_9101 ,shbi_num_9101,ecom_ref_9101,sch_name_9101 having count(*)>1) a, (select count(*) dt_count, act_code_9101 act_code_9101_dt,shbi_num_9101,ecom_ref_9101,sch_name_9101 from edi.SHB_UTIL_9101 where act_code_9101='DTCH' group by act_code_9101,shbi_num_9101,ecom_ref_9101,sch_name_9101 having count(*)>1) b, ECOM.ECOM_REFM_2001 c, edi.SHB_ITEM_9102 d where a.at_count > b.dt_count and a.ecom_ref_9101=b.ecom_ref_9101 and a.sch_name_9101=b.sch_name_9101 and a.shbi_num_9101=b.shbi_num_9101 and b.sch_name_9101=c.SCH_NAME_2001 and b.ecom_ref_9101=c.ecom_ref_2001 and d. SHBI_NUM_9102=b. SHBI_NUM_9101 and c. SCH_NAME_2001=d. EXP_SCHM_9102) with no data

insert into shb_atrch_dtch (select a.*,b.dt_count, b.act_code_9101_dt, c.* from (select count(*) at_count, act_code_9101 act_code_9101_at,shbi_num_9101,ecom_ref_9101,sch_name_9101 from edi.SHB_UTIL_9101 where act_code_9101='ATCH' group by act_code_9101 ,shbi_num_9101,ecom_ref_9101,sch_name_9101 having count(*)>1) a, (select count(*) dt_count, act_code_9101 act_code_9101_dt,shbi_num_9101,ecom_ref_9101,sch_name_9101 from edi.SHB_UTIL_9101 where act_code_9101='DTCH' group by act_code_9101,shbi_num_9101,ecom_ref_9101,sch_name_9101 having count(*)>1) b, ECOM.ECOM_REFM_2001 c, edi.SHB_ITEM_9102 d where a.at_count > b.dt_count and a.ecom_ref_9101=b.ecom_ref_9101 and a.sch_name_9101=b.sch_name_9101 and a.shbi_num_9101=b.shbi_num_9101 and b.sch_name_9101=c.SCH_NAME_2001 and b.ecom_ref_9101=c.ecom_ref_2001 and d. SHBI_NUM_9102=b. SHBI_NUM_9101 and c. SCH_NAME_2001=d. EXP_SCHM_9102 and c. FILE_NUM_2001=d. ACT_FILE_9102)

select * from db2admin.shb_atrch_dtch a, (select LICE_NUM_9102 lic, file_num_2001 fil from db2admin.shb_atrch_dtch except select LIC_NUMB_1503, FILE_NUM_1503 from dgftrla.licm.lic_canc_1503) b, onln.shbi_mast_9001

c where EXP_CNT_9001='CN' and PORT_COD_9001 in (<inland port codes where export is not allowed>) and a.LICE_NUM_9102=b. lic and a. file_num_2001=b. Fil and c. FILE_NUMB_9001=b.fil and a. FILE_NUMB_9001=c. FILE_NUMB_9001 and c. SHBI_NUM_9001=a. SHBI_NUM_9101

2. **Query:** To extract records where license under Zero duty EPCG scheme and SHIS scheme was issued in same year.

```
SQL: SELECT *
FROM (SELECT *
FROM LICM.LIC_MAST_1500
WHERE LIC_CATG_1500 = '31') AS A, (
SELECT *
FROM LICM.LIC_MAST_1500
WHERE LIC_CATG_1500 = '34') AS B
WHERE A.IEC_NUMB_1500 = B.IEC_NUMB_1500 AND YEAR(A.LIC_DATE_1500) =
YEAR(B.LIC_DATE_1500)
```

3. **Query:** To extract records where currency exchange rate used in the shipping bill is not as per the defined currency rate master table COMMON.CUR_EXPT_181 (all the currency code and respective exchange rate notification is updated in this table for future reference). Since some of the currency code used by exporters are not standard, hence we will make to separate query to find records where currency exchange rate is not as per notification.

For standard currency codes:

```
SQL: select * from (SELECT FOB_FCCD_9102, MONTH(SHBI_DAT_9102), YEAR(SHBI_DAT_9102),
EXC_INRS_9102
FROM EDI.SHB_ITEM_9102
WHERE EXC_INRS_9102 <> 0
GROUP BY FOB_FCCD_9102, MONTH(SHBI_DAT_9102), YEAR(SHBI_DAT_9102),
EXC_INRS_9102
EXCEPT
SELECT CUST_COD_181, MONTH(EFF_DATE_181), YEAR(EFF_DATE_181),
EQV_INRS_181
FROM COMMON.CUR_EXPT_181) where FOB_FCCD_9102 in (select distinct(CUST_COD_181) from
COMMON.CUR_EXPT_181
```

For non-standard currency codes:

```
SQL: select * from (SELECT FOB_FCCD_9102, MONTH(SHBI_DAT_9102), YEAR(SHBI_DAT_9102),
EXC_INRS_9102
FROM EDI.SHB_ITEM_9102
WHERE EXC_INRS_9102 <> 0
GROUP BY FOB_FCCD_9102, MONTH(SHBI_DAT_9102), YEAR(SHBI_DAT_9102),
EXC_INRS_9102
EXCEPT
SELECT CUST_COD_181, MONTH(EFF_DATE_181), YEAR(EFF_DATE_181),
EQV_INRS_181
FROM COMMON.CUR_EXPT_181) where FOB_FCCD_9102 not in (select distinct(CUST_COD_181)
from COMMON.CUR_EXPT_181
```

Same query can be replicated in all the tables where currency exchange rate is used by replacing the EDI.SHB_ITEM_9102 table and respective columns with table and columns where you want test the condition. Eg: We can replace EDI.SHB_ITEM_9102 table with depb. depb_pep_403 and FOB_FCCD_9102, SHBI_DAT_9102, EXC_INRS_9102 with FOB_FCCD_403, CON_DATE_403, CONV_RAT_403 respectively.

4. **Query:** To extract records with summary of day wise different currency exchange rate used for a currency code for further analysis in excel to check if any abnormal exchange rate is used (Eg < 1 or > 300)

```
SQL: SELECT COUNT(*), FOB_FCCD_9102, SHBI_DAT_9102, EXC_INRS_9102
FROM EDI.SHB_ITEM_9102 where FOB_FCCD_9102 in (select distinct(CUST_COD_181) from
COMMON.CUR_EXPT_181
GROUP BY FOB_FCCD_9102, SHBI_DAT_9102, EXC_INRS_9102
```

```
SELECT COUNT(*), REAL_CUR_9105, SHBI_DAT_9105, EXCH_RAT_9105
FROM EDI.BRC_UTIL_9105 where REAL_CUR_9105 in (select distinct(CUST_COD_181) from
COMMON.CUR_EXPT_181
GROUP BY REAL_CUR_9105, SHBI_DAT_9105, EXCH_RAT_9105
```

```
SELECT COUNT(*), FOB_FCCD_9002, SHBI_DAT_9002, EXC_INRS_9002
FROM ONLN.SHBI_ITEM_9002 where FOB_FCCD_9002 in (select distinct(CUST_COD_181) from
COMMON.CUR_EXPT_181
GROUP BY FOB_FCCD_9002, SHBI_DAT_9002, EXC_INRS_9002
```

5. **Query:** To extract records with details when different currency exchange rate is used for same currency code for different shipping bills in same day.

```
create table diff_curr_exch_rate (select a.*,b.EXC_inrs_9102 from
(select count(*),FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 exc from edi.SHB_ITEM_9102
where EXC_inrs_9102>1
GROUP BY FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 with ur) a,
(select count(*),FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 exc_1 from edi.SHB_ITEM_9102
where EXC_inrs_9102>1
GROUP BY FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 ) b
where a.FOB_fccd_9102=b.FOB_fccd_9102 and
a.SHBI_DAT_9102=b.SHBI_DAT_9102
and a.EXC<>b.EXC_1) with no data
```

```
insert into table diff_curr_exch_rate (select a.*,b.EXC_inrs_9102 from
(select count(*),FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 exc from edi.SHB_ITEM_9102
where EXC_inrs_9102>1
GROUP BY FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 with ur) a,
(select count(*),FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 exc_1 from edi.SHB_ITEM_9102
where EXC_inrs_9102>1
GROUP BY FOB_fccd_9102, SHBI_DAT_9102, EXC_inrs_9102 ) b
where a.FOB_fccd_9102=b.FOB_fccd_9102 and
a.SHBI_DAT_9102=b.SHBI_DAT_9102
and a.EXC<>b.EXC_1 with ur)
```

```
select * from db2admin.shb_atrch_dtch a, (select LICE_NUM_9102 lic, file_num_2001 fil from
db2admin.shb_atrch_dtch
except
select LIC_NUMB_1503, FILE_NUM_1503 from dgftrla.licm.lic_canc_1503) b, diff_curr_exch_rate
c where a.LICE_NUM_9102=b. lic and a. file_num_2001=b. Fil and c. LICE_NUM_9102 = b.lic and c.
SHBI_NUM_9102=a. SHBI_NUM_9101
```

6. **Query:** To extract records from currency exchange rate master on a particular date (20120411) to validate the accuracy and timely updation of currency exchange rate in master as per the notification passed on same date.

```
SQL: select * from common.CUR_EXPT_181 where eff_date_181='2012-04-11'
```

7. **Query:** To extract records with shipping bill date between 20120208 and 20120301 and currency code is CHF and exchange rate used is 57.55. As per the currency rate change notification dated 20120201 no rate was notified for CHF and 57.55 was notified on 20120301. Since currency code CHF was made effective in currency master from 20120208 so query will give us the transaction details where currency exchange rate mentioned in master (ie 57.55 for CHF) is used even though it was not notified.

```
SQL: SELECT *
FROM EDI.BRC_UTIL_9105
WHERE REAL_CUR_9105 = 'CHF' AND SHBI_DAT_9105 >= '2012-02-08'
AND SHBI_DAT_9105 < '2012-03-01'
```

```
SELECT *
FROM EDI.SHB_ITEM_9102
WHERE FOB_FCCD_9102 = 'CHF' AND SHBI_DAT_9102 >= '2012-02-08'
AND SHBI_DAT_9102 < '2012-03-01' AND EXC_INRS_9102 = '57.55'
```

```
SELECT A.*, B.SHBI_DAT_2502
FROM VFFM.VFFM_PEP_2503 AS A, VFFM.VFFM_PES_2502 AS B
WHERE FOB_FCCD_2503 = 'CHF' AND B.SHBI_DAT_2502 >= '2012-02-08'
AND B.SHBI_DAT_2502 < '2012-03-01' AND CONV_RAT_2503 = '57.55'
AND A.SHBI_NUM_2503 = B.SHBI_NUM_2502
```

8. **Query:** To extract records where incorrect currency exchange rate is used ie less than 1 and currency code is not Japanese Yen.

```
SQL: Select * from VFFM.VFFM_PEP_2503 where CONV_RAT_2503<'1' and FOB_FCCD_2503
<>'JPY'
```

9. **Query:** To extract records where '23069027', '210111' and '21011200' ITCH codes were used under VKUY export scheme prior to their addition as per notice dated 15feb11.

```
SQL: SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B
WHERE ITCH_COD_2103 IN ('23069027', '210111', '21011200') --,'15162039','120740')
AND A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102 and b.
EXP_DATE_2102<='2011-02-15'
WITH UR
```

10. **Query:** To extract records where product description is not changed for SION – 532,533,535,534 as mentioned in notification dated 14oct11 and 13jul11

```
SQL: SELECT A.*, B.LIC_DATE_1500
FROM LICM.EXP_ITEM_1501 AS A, LICM.LIC_MAST_1500 AS B
WHERE A.SION_NUM_1501 IN ('532', '533', '535', '534')
AND B.LIC_NUMB_1500 = A.LIC_NUMB_1501 AND B.LIC_DATE_1500 >= '2011-10-14'
```

```
SELECT A.*, B.LIC_DATE_1500
FROM LICM.EXP_ITEM_1501 AS A, LICM.LIC_MAST_1500 AS B
WHERE A.SION_NUM_1501 IN ('2913', '263') AND B.LIC_NUMB_1500 = A.LIC_NUMB_1501
AND B.LIC_DATE_1500 >= '2011-07-13'
```

11. **Query:** To extract the SION wise summary of records where license is issued with SION which was suspended as per notification dated 7aug14. The o/p can further be analysed in excel to see which of the SIONs have been used and compared with the SIONs mentioned in notification.

```
SQL: SELECT COUNT(B.LIC_NUMB_1501), B.SION_NUM_1501
FROM LICM.LIC_MAST_1500 AS A, LICM.EXP_ITEM_1501 AS B
WHERE A.LIC_DATE_1500 BETWEEN '2013-10-04' AND '2014-07-10'
AND A.LIC_NUMB_1500 = B.LIC_NUMB_1501
GROUP BY B.SION_NUM_1501
```

12. **Query:** To extract records where imports done by BCCI for further analysis of the scheme and notification used by BCCI Imports for which they are not entitled.

```
SQL: SELECT *
FROM LICM.LIC_MAST_1500
WHERE UPPER(APP_NAME_1500) LIKE '%CRICKET%' AND UPPER(APP_NAME_1500) LIKE
'%BOARD%'
```

```
SELECT *
FROM BE_MAIN AS A, BE_ITEM_DET AS B
WHERE IEC = '0300009895' AND A.BE_NO = B.BE_NO
```

13. **Query:** To extract the records where export was done under VKUY scheme for ITCHS code - '13023210', '1302320', '13023230', '15162039', '120740' which were deleted as per notification dated 5jun2012.

```
SQL: SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B
WHERE ITCH_COD_2103 IN ('13023210', '1302320', '13023230', '15162039',
'120740') AND A.FILE_NUM_2103 = B.FILE_NUM_2102
AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102 AND B.EXP_DATE_2102 >= '2012-06-05'
WITH UR
```

```
SELECT *
FROM ONLN.SHBI_ITEM_9002 AS A, VKUY.VKUY_PES_2102 AS B
WHERE A.ITCH_COD_9002 IN ('13023210', '1302320', '13023230', '15162039',
'120740') AND A.SHBI_DAT_9002 >= '2012-06-05'
AND A.SHBI_NUM_9002 = B.SHBI_NUM_2102 AND A.FILE_NUMB_9002 = B.FILE_NUM_2102
WITH UR
```

14. **Query:** To extract records from VKUY rate master table to validate the accuracy and timely updation of VKUY rates for ITCH codes 15162039, 120740.

```
SQL: select * from VKUY.VKUY_RAT_2105 where ITCH_CODE_2105 in ('15162039', '120740')
```

15. **Query:** To extract records where export was done for ITCH codes '13021930', '21012010' under VKUY scheme prior to addition of these ITCHS code as per the notification dated 23aug10

```
SQL: SELECT *
FROM VKUY.VKUY_PES_2102 AS A, VKUY.VKUY_PEP_2103 AS B
WHERE B.ITCH_COD_2103 IN ('13021930', '21012010') AND A.SHBI_DAT_2102 < '2010-04-01'
AND A.SHBI_NUM_2102 = B.SHBI_NUM_2103 AND A.FILE_NUM_2102 = B.FILE_NUM_2103
WITH UR
```

16. **Query:** To extract records with Shipping bill and IEC details where benefit given to exporter under DEPB scheme is with depb rate more than 1 and without the reduced benefit under VKUY scheme.

```
SQL: SELECT *
FROM (SELECT A.FILE_NUM_401, A.FILE_DAT_401, A.DEPB_TYP_401, A.IEC_NUMB_401,
A.LIC_NUMB_401, B.*
FROM DEPB.DEPB_POE_401 AS A, DEPB.DEPB_PEP_403 AS B
WHERE A.FILE_NUM_401 = B.FILE_NUM_403 AND A.FILE_DAT_401 > '2013-03-31'
AND A.FILE_NUM_401 <> '' AND B.RATE_AWD_403 > '1') AS X, (
SELECT A.FILE_NUM_2101, A.FILE_DAT_2101,
A.IEC_NUMB_2101, A.LIC_NUMB_2101,
B.*
FROM VKUY.VKUY_POE_2101 AS A,
VKUY.VKUY_PEP_2103 AS B
WHERE A.FILE_NUM_2101 = B.FILE_NUM_2103 AND B.RATE_AWD_2103 >= '5'
AND A.FILE_DAT_2101 > '2013-03-31') AS Y
WHERE X.IEC_NUMB_401 = Y.IEC_NUMB_2101 AND X.SHBI_NUM_403 = Y.SHBI_NUM_2103
WITH UR
```

17. **Query:** To extract records where currency exchange rate used in shipping bill is < 1 and currency code not of Japanese yen or shipping bill with blank currency code
- SQL:** select * from depb.depb_pep_403 where CONV_RAT_403<'1' and FOB_FCCD_403 <> 'JPY'
 select * from depb.depb_pep_403 where FOB_FCCD_403 is null or FOB_FCCD_403=' '
 select * from onln.shbi_item_9002 where EXC_INRS_9002<'1' and FOB_FCCD_9002 <> 'JPY'
 select * from depb.depb_pep_403 where FOB_FCCD_9002 is null or FOB_FCCD_9002=' '
18. **Query:** To extract records where RCMC issued by an old name RCMC issuing authority instead of new name RCMS issuing authority which was changed as per notification dated 25may11
- SQL:** select * from iec.IEC_STAX_217 where upper(ISSUD_BY_217) like '%JUTE%'
 select * from iec.IEC_RCMC_216 where upper(ISSUD_BY_216) like '%JUTE%' and
 RGN_DATE_216>='2011-05-25'
19. **Query:** To extract records with shipping details where DEPB benefit was given on a product even though benefit of the product under DEPB scheme was withdrawn as per notification date 25jan11
- SQL:** SELECT *
 FROM DEPB.DEPB_PEP_403 AS A, DEPB.DEPB_PES_402 AS B
 WHERE PRD_CODE_403 IN ('90') AND DEPB_SNO_403 IN ('22C', '22D')
 AND A.FILE_NUM_403 = B.FILE_NUM_402 AND A.SHBI_NUM_403 = B.SHBI_NUM_402
 AND EXP_DATE_402 >= '2011-01-25' AND UPPER(ITEM_DES_403) LIKE '%MILK%'
 AND DEPB_CVA_403 > '0'
20. **Query:** To extract records from DEPB rate master where PRD_CODE_413 IN ('90') AND DEPB_SNO_413 IN ('22C', '22D') to validate the accuracy and timely updation of rate master as per notification dated 25jan11.
- SQL:** SELECT *
 FROM DEPB.DEPB_RAT_413 WHERE PRD_CODE_413 IN ('90') AND DEPB_SNO_413 IN
 ('22C', '22D') and EFF_DATE_413>='20111001'
21. **Query:** To extract records where Shipping bill was used with let export order date after of before valid RCMC, to validate if any bill was filled without having any valid RCMC
- SQL:** SELECT *
 FROM IEC.IEC_RCMC_221 AS A, ONLN.SHBI_MAST_9001 AS B
 WHERE A.IEC_NUMB_221 = B.IEC_NUMB_9001
 AND (LEO_DATE_9001 < RCMC_ISDT_221 OR LEO_DATE_9001 > RCMC_EXDT_221)
22. **Query:** To extract records where EPCG license was issued after of before valid RCMC, to validate if any license was issued without having any valid RCMC
- SQL:** SELECT *
 FROM IEC.IEC_RCMC_215 AS A, EPCG.EPCG_MAST_801 AS B
 WHERE A.IEC_NUMB_215 = B.IEC_NUMB_801 AND A.RCMC_NUM_215 = B.RCMC_NUM_801
 AND (FLNO_DAT_801 < RCMC_ISDT_215 OR FLNO_DAT_801 > RCMC_EXDT_215)
- SQL:** SELECT *
 FROM IEC.IEC_RCMC_221 AS A, EPCG.EPCG_MAST_801 AS B
 WHERE A.IEC_NUMB_221 = B.IEC_NUMB_801 AND A.RCMC_NUMB_221 = B.RCMC_NUM_801
 AND (FLNO_DAT_801 < RCMC_ISDT_221 OR FLNO_DAT_801 > RCMC_EXDT_221)
23. **Query:** To extract records with shipping bill details where export was done on excluded items -- 0902,0903,0904 as per appendix 37a

DGFT DEL Database --

SQL: SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B, VKUY.VKUY_POE_2101 AS C
WHERE A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102
AND SUBSTR(A.ITCH_COD_2103, 1, 4) IN ('0902', '0903', '0904')
AND A.FILE_NUM_2103 = C.FILE_NUM_2101 AND B.FILE_NUM_2102 = C.FILE_NUM_2101

DGFTMAIN Database --

SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B
WHERE A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102
AND SUBSTR(ITCH_COD_2103, 1, 4) IN ('0902', '0903', '0904')

DGFTRLA Database --

SELECT *
FROM LICM.VKUY_PEP_2103 AS A, LICM.VKUY_PES_2102 AS B
WHERE A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102
AND SUBSTR(A.ITCH_COD_2103, 1, 4) IN ('0902', '0903', '0904')

24. **Query:** To extract records with shipping bill details where export was done on excluded items -- 0902,0903,0904 as per appendix 37a and respective file number was not found in license master of all license issued by Delhi RLA(Lic_mast_1500) in Delhi RLA database.

SQL: SELECT *
FROM
(SELECT *
FROM
(SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B, VKUY.VKUY_POE_2101 AS C
--upper(ITEM_DES_2103) like '%PEPPER%'
WHERE A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102
--AND SHBI_DAT_2102 > '2010-08-23'
AND SUBSTR(A.ITCH_COD_2103, 1, 4) IN ('0902', '0903', '0904')
AND A.FILE_NUM_2103 = C.FILE_NUM_2101 AND B.FILE_NUM_2102 = C.FILE_NUM_2101) AS X
LEFT OUTER JOIN LICM.LIC_MAST_1500 AS Y ON X.FILE_NUM_2103 = Y.FILE_NUM_1500) AS Z
WHERE FILE_NUM_1500 IS NULL

25. **Query:** To extract records with shipping bill details where export was done on excluded items -- 0902,0903,0904 as per appendix 37a and respective file number was not found in license master of all license issued (Lic_mast_1500) by HQ RLA in DGFTRLA database

SQL: SELECT *
FROM
(SELECT *
FROM
(SELECT *
FROM VKUY.VKUY_PEP_2103 AS A, VKUY.VKUY_PES_2102 AS B
WHERE A.FILE_NUM_2103 = B.FILE_NUM_2102 AND A.SHBI_NUM_2103 = B.SHBI_NUM_2102
AND SUBSTR(A.ITCH_COD_2103, 1, 4) IN ('0902', '0903', '0904')) AS X LEFT OUTER JOIN
DB2ADMIN.LIC_MAST_1500 AS Y ON X.FILE_NUM_2103 = Y.FILE_NUM_1500) AS Z
WHERE FILE_NUM_1500 IS NULL

26. **Query:** To extract records where multiple IECs are active on same pan after 20110331 and mapped with IEC cancel table to analyse the reason for cancellation

SQL: select * from IEC.IEC_MAST_211 a
(select COUNT (*), EXPO_PAN_211 FROM IEC.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31')

```

and EXPO_PAN_211 not like ' %'
GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1
) b
on a.EXPO_PAN_211=b.EXPO_PAN_211
left outer join iec.can_iecm_217 d
on d.IEC_NUMB_217 = a.IEC_NUMB_211
where IEC_STAT_211=0

```

27. **Query:** To extract records where IECs with status equal 3 and found in IEC cancel table, to validate if IEC with status 3 corresponds to cancel IEC or not.

```

SQL: SELECT *
FROM IEC.IEC_MAST_211 AS A, IEC.CAN_IECM_217 AS B
WHERE IEC_STAT_211 = 3 AND UPPER(CAN_REAS_217) NOT LIKE '%RESTOR%'
AND B.IEC_NUMB_217 = A.IEC_NUMB_211

```

28. **Query:** To extract records where IECs with status equal 3 and found in IEC cancel table and were not sent to custom despite being cancelled.

```

SQL: select * from iec.IEC_STAT_224 x right outer join (select * from iec.IEC_MAST_211 a,
iec.can_iecm_217 b where IEC_STAT_211=3 and upper(can_reas_217) not like '%RESTOR%'
and b.iec_numb_217=a.iec_numb_211) y on x.iec_numb_224=y.iec_numb_211

```

29. **Query:** To extract records where IECs with status equal 3 and found in IEC cancel table mapped with IES status table

```

SQL: select * from iec.IEC_STAT_224 x, (select * from iec.IEC_MAST_211 a, iec.can_iecm_217 b
where IEC_STAT_211=3 and upper(can_reas_217) not like '%RESTOR%'
and b.iec_numb_217=a.iec_numb_211) y where x.iec_numb_224=y.iec_numb_211

```

30. **Query:** To extract records where import was done with cancelled IEC

```

SQL: select a.iec,a.be_no, a.be_dt, a.imp_name,a.TOT_VAL, b.* from be_main a, cancelled_iec b
where a.iec = b.IEC_NUMB_211 and a.BE_DT>b.CAN_DATE_217

```

31. **Query:** To extract records where import was done with multiple active IECs having same PAN number, and import was done using second active IEC

```

SQL: select * from doubl_iec_same_pan x,
(select count(*),iec,second_alt,max(be_dt),min(be_dt),sum(TOT_ASS_VAL) from be_main a,
(select b.SERIAL,b.EXPO_PAN_211,c.IEC_NUMB_211,b.IEC_ALDT_211 second_alt,c.IEC_ALDT_211
first_alt from doubl_iec_same_pan b,doubl_iec_same_pan c
where b.SERIAL=2 and
b.EXPO_PAN_211=c.EXPO_PAN_211 ) d where
d.IEC_NUMB_211=a.iec and
be_dt>d.second_alt
group by iec,second_alt) y
where x.IEC_NUMB_211=y.iec order by 1,3

```

The cursor is used to update the data of multiple IECs with same PAN in table DOUBL_IEC_SAME_PAN by giving serial number to multiple IECs.

```

declare
cursor cc is select EXPO_PAN_211, IEC_ALDT_211
from DOUBL_IEC_SAME_PAN order by 1,2;
aa varchar2(15):='a';
a1 varchar2(15);
cnt number:=1;
bb date;

```

```

begin
open cc;
loop
fetch cc into aa,bb;
exit when cc%notfound;
if a1=aa then
cnt:=cnt+1;
else
cnt:=1;
a1:=aa;
end if;
update DOUBL_IEC_SAME_PAN set SERIAL=cnt
where EXPO_PAN_211=aa and IEC_ALDT_211=bb;
end loop;
close cc;
end;

```

32. **Query:** To extract records where multiple IEC were issued with same PAN number

SQL: select COUNT (*), EXPO_PAN_211 FROM IEC.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31' GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1;

33. **Query:** To extract records where licenses were issued with validity prior to the issue date

SQL: Select * from LICM.LIC_MAST_1500 where LIC_DATE_1500 > LIC_VLDT_1500 fetch first 200 rows only

34. **Query:** To extract records where multiple IEC were issued with same PAN number mapped with cancel IEC reason

SQL: select * from IEC.IEC_MAST_211 a join
(select COUNT (*), EXPO_PAN_211 FROM IEC.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31'
and EXPO_PAN_211 not like ' %'
GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1
) b
on a.EXPO_PAN_211=b.EXPO_PAN_211
left outer join iec.can_iecm_217 d
on d.IEC_NUMB_217 = a.IEC_NUMB_211
where IEC_STAT_211<>0

35. **Query:** To extract records where multiple licence along with IEC were issued with new pan even though previous licence was active

SQL: select a.IEC_NUMB_211 can_IEC, a.EXPO_PAN_211,LIC_NUMB_1500,
LIC_DATE_1500,c.IEC_ALDT_211, c.IEC_NUMB_211 act_IEC, a.DAT_MODI_211
from LICM.lic_mast_1500 d,db2admin.IEC_MAST_211 a,
(select COUNT (*), EXPO_PAN_211 FROM db2admin.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31' and EXPO_PAN_211 not like ' %'
GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1) b,
db2admin.IEC_MAST_211 c
where a.EXPO_PAN_211=b.EXPO_PAN_211
and d.IEC_NUMB_1500=c.IEC_NUMB_211
and a.IEC_STAT_211 =3
and c.IEC_STAT_211 =0
and b.EXPO_PAN_211=c.EXPO_PAN_211
and LIC_DATE_1500<a.DAT_MODI_211

36. **Query:** To extract records where multiple IEC were issued with same PAN number mapped with status details table to analyse which IEC are active and which are cancel.

```

SQL: select * from (select a.* from IEC.IEC_MAST_211 a,
(select COUNT (*), EXPO_PAN_211 FROM IEC.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31'
GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1) b
where a.EXPO_PAN_211=b.EXPO_PAN_211 and a.EXPO_PAN_211 not like ' %') c left outer join
iec.IEC_STAT_224 d on c.IEC_NUMB_211 = d.IEC_NUMB_224

```

```

select * from IEC.IEC_MAST_211 a join
(select COUNT (*), EXPO_PAN_211 FROM IEC.IEC_MAST_211
WHERE IEC_ALDT_211 > '2011-03-31' and EXPO_PAN_211 not like ' %'
GROUP BY EXPO_PAN_211 HAVING COUNT(*)>1) b
on a.EXPO_PAN_211=b.EXPO_PAN_211
left outer join
iec.IEC_STAT_224 d on a.IEC_NUMB_211 = d.IEC_NUMB_224

```

37. **Query:** To extract records where all multiple IEC issued are active with same PAN number

```

SQL: SELECT *
FROM
IEC.IEC_MAST_211 AS A JOIN (
SELECT COUNT(*), EXPO_PAN_211, IEC_STAT_211
FROM IEC.IEC_MAST_211
WHERE EXPO_PAN_211 NOT LIKE ' %' AND IEC_STAT_211 = 0
GROUP BY EXPO_PAN_211, IEC_STAT_211
HAVING COUNT(*) > 1) AS B ON A.EXPO_PAN_211 = B.EXPO_PAN_211 LEFT OUTER JOIN
IEC.CAN_IECM_217 AS D ON D.IEC_NUMB_217 = A.IEC_NUMB_211
WHERE A.IEC_STAT_211 = 0

```

```

Declare
cursor cc is select EXPO_PAN_211, IEC_ALDT_211
from DOUBL_IEC_SAME_PAN order by 1,2;
aa varchar2(15):='a';
a1 varchar2(15);
cnt number:=1;
bb date;
begin
open cc;
loop
fetch cc into aa,bb;
exit when cc%notfound;
if a1=aa then
cnt:=cnt+1;
else
cnt:=1;
a1:=aa;
end if;

```

```

UPDATE DOUBL_IEC_SAME_PAN
SET SERIAL = CNT
WHERE EXPO_PAN_211 = AA AND IEC_ALDT_211 = BB;

```

```

end loop;
close cc;
end;

```

38. **Query:** To extract records where a shipping bill can be used in single application for VFFM schemes (VKUY, FMS, FPS and MLFPS) (ref: Sl. 5 & 6 of the Ayat Niryat Form No.).

```

SQL: select distinct a.USR_CODE_9101, a.SHBI_NUM_9101,
        a.SHBI_DAT_9101, a.SCH_NAME_9101,atch, dtch,
        s.ECOM_REF_9101, FILE_NUM_1500, FOB, c.DUTY_CIF , totentt, entt, shbi_sts_2502,
        shbi_fob_2502,
        c.IEC_NUMB_1500, c.LIC_COUNT, c.LIC_CATG_1500, c.first_lic ,
        c.LIC_DATE,c.SCH_NAME_2001
from UTIL_DTCH a, UTIL_ATCH d, shb_util s, vffm_pes vs, vffm_pep_sb vp, vffm_pep_file vp2,
lic_sum c
where a.USR_CODE_9101=d.USR_CODE_9101
and a.SHBI_NUM_9101=d.SHBI_NUM_9101
and a.SHBI_DAT_9101=d.SHBI_DAT_9101
and a.SCH_NAME_9101=d.SCH_NAME_9101
and atch-dtch>1
and a.USR_CODE_9101=s.USR_CODE_9101
and a.SHBI_NUM_9101=s.SHBI_NUM_9101
and a.SHBI_DAT_9101=s.SHBI_DAT_9101
and a.SCH_NAME_9101=s.SCH_NAME_9101
and ecom_ref_9101=ECOM_REF_2001
and FILE_NUM_1500=file_num_2502
and shbi_sts_2502='U'
and shbi_num_2502=s.shbi_num_9101
and SHBI_DAT_2502=s.SHBI_DAT_9101
and vp.shbi_num_2503= s.shbi_num_9101
and vp.file_num_2503=FILE_NUM_1500
and vp2.file_num_2503=FILE_NUM_1500

```

```

create table ecom_refm as select * from ecom_refm_2001 where SBMT_DAT_2001>='01-apr-2011';
create table shb_util as select a.* from shb_util_9101 a,ecom_refm_2001 where
ecom_ref_9101=ECOM_REF_2001 and SBMT_DAT_2001>='01-apr-2011';

```

```

create table util_atc as select count(*) atch, USR_CODE_9101, SHBI_NUM_9101, SHBI_DAT_9101,
SCH_NAME_9101,act_code_9101
from shb_util_9101 ,ecom_refm_2001
where ECOM_REF_9101 is not null
and act_code_9101='ATCH'
and ECOM_REF_9101=ECOM_REF_2001 and SBMT_DAT_2001>='01-apr-2011'
group by USR_CODE_9101, SHBI_NUM_9101, SHBI_DAT_9101, SCH_NAME_9101, act_code_9101;

```

```

create table util_dtch as select count(*) dtch, USR_CODE_9101, SHBI_NUM_9101, SHBI_DAT_9101,
SCH_NAME_9101,act_code_9101
from shb_util_9101 ,ecom_refm_2001
where ECOM_REF_9101 is not null
and act_code_9101='DTCH'
and ECOM_REF_9101=ECOM_REF_2001 and SBMT_DAT_2001>='01-apr-2011'
group by USR_CODE_9101, SHBI_NUM_9101, SHBI_DAT_9101, SCH_NAME_9101, act_code_9101;

```

```

create table lic as select sum(FOB_INRS_1500) fob,sum(cif_inrs_1500) duty , lic_catg_1500,
FILE_NUM_1500 from lic_mast_1500, ecom_refm_2001 where file_num_1500=file_num_2001 and
SBMT_DAT_2001>='01-apr-2011'
group by FILE_NUM_1500, lic_catg_1500;

```

```

create table vffm_pep_file as select file_num_2503, sum(act_entl_2503) totentt
from vffm_pep_2503 ,ecom_refm_2001
where file_num_2001=file_num_2503 and SBMT_DAT_2001>='01-apr-2011'
group by file_num_2503;

```

```

create table vffm_pep_sb as select shbi_num_2503, file_num_2503, sum(act_entl_2503) entt
from vffm_pep_2503 , ecom_refm_2001
where file_num_2001=file_num_2503 and SBMT_DAT_2001>='01-apr-2011'
group by shbi_num_2503, file_num_2503;

```

```
create table vffm_pes as select a.* from vffm_pes_2502 a, ecom_refm_2001
where file_num_2502=file_num_2001 and file_num_2001 not like '%' and SBMT_DAT_2001>='01-
apr-2011';
```

39. **Query:** To extract records where SHIS can be applied for in the year subsequent to year of export. Zero duty EPCG cannot be granted to a license holder who availed SHIS in that particular year. The SHIS cannot be claimed in such cases after the due date of claim, either (even with application of provisions of Late Cut), if Zero Duty EPCG has already been claimed for the relevant year. (ref: Para 3.10.3 and 5.1A of HBP)

```
SQL: select * from dgftrla.licm.lic_mast_1500 e, SHIS.SHIS_MAST_2601, ecom.ecom_refm_2001,
dgftrla.licm.lic_mast_1500 s
where e.lic_catg_1500=12
and e.LIC_DATE_1500 between '2010-04-01' and '2011-03-31'
and e.iec_numb_1500=IEC_NUMB_2601
and FIN_YEAR_2601 ='2009-2010'
and file_num_2601=ecom_ref_2001
and file_num_2001=s.file_num_1500
and s.lic_catg_1500=31
```

```
select * from dgftrla.licm.lic_mast_1500 e, SHIS.SHIS_MAST_2601, ecom.ecom_refm_2001,
dgftrla.licm.lic_mast_1500 s
where e.lic_catg_1500=12
and e.LIC_DATE_1500 between '2011-04-01' and '2012-03-31'
and e.iec_numb_1500=IEC_NUMB_2601
and FIN_YEAR_2601 ='2010-2011'
and file_num_2601=ecom_ref_2001
and file_num_2001=s.file_num_1500
and s.lic_catg_1500=31
```

```
select * from dgftrla.licm.lic_mast_1500 e, SHIS.SHIS_MAST_2601, ecom.ecom_refm_2001,
dgftrla.licm.lic_mast_1500 s
where e.lic_catg_1500=12
and e.LIC_DATE_1500 between '2012-04-01' and '2013-03-31'
and e.iec_numb_1500=IEC_NUMB_2601
and FIN_YEAR_2601 ='2011-2012'
and file_num_2601=ecom_ref_2001
and file_num_2001=s.file_num_1500
and s.lic_catg_1500=31
```

```
select * from dgftrla.licm.lic_mast_1500 e, SHIS.SHIS_MAST_2601, ecom.ecom_refm_2001,
dgftrla.licm.lic_mast_1500 s
where e.lic_catg_1500=12
and e.LIC_DATE_1500 between '2013-04-01' and '2014-03-31'
and e.iec_numb_1500=IEC_NUMB_2601
and FIN_YEAR_2601 ='2012-2013'
and file_num_2601=ecom_ref_2001
and file_num_2001=s.file_num_1500
and s.lic_catg_1500=31
```

40. **Query:** To extract records where a incense holder cannot be issued a licence if black listed under DEL. a Denied Entity List (DEL) is maintained as per provisions of Enforcement Division of DGFT Circular vide F.No. 18/24//HQ/99-2000/ECA II dated December 31, 2003 read with Rule 7 of Foreign Trade (Regulation) Rules 1993.

```
SQL: select * from lic_mast_1500 , bls_den_1005 ,
(select count(*) cnt,iecnumb_1005 iec
from bls_den_1005
```

```

group by iec_numb_1005 having count(*)=1)
where lic_date_1500>='01-apr-2011'
  and iec_numb_1005=iec
  and ord_stat_1005=1
  and iec_numb_1500=iec
  and lic_date_1500>ord_date_1005

select * from lic_mast_1500 , bls_den_1005 , LIC_CANC_1503,
(select count(*) cnt,iec_numb_1005 iec
 from bls_den_1005
group by iec_numb_1005 having count(*)=1)
where lic_date_1500 >= '01-apr-2011'
  and iec_numb_1005=iec
  and ord_stat_1005=1
  and iec_numb_1500=iec
  and lic_date_1500 >= ord_date_1005
  and LIC_NUMB_1503=LIC_NUMB_1500

```

41. **Query:** 1.To compare the number of Licences issued in RLA Chennai data with that of number of licences issued in DGFT RLA data

SQL: select count(*), lic_catg_1500 from licm.lic_mast_1500 WHERE
SUBSTR(FILE_NUM_1500,13,4) IN ('AM12','AM13','AM14','AM12S','AM13S','AM14S') group by
lic_catg_1500;(RLA Chennai)

select count(*), lic_catg_1500 from licm.lic_mast_1500 where lic_numb_1500 like '04%' and
SUBSTR(FILE_NUM_1500,13,4) IN ('AM12','AM13','AM14','AM12S','AM13S','AM14S') group by
lic_catg_1500;(DGFT RLA)

42. **Query:** To extract records where calculation of FOB value realised in rupee terms is not correct.

SQL: select FILE_NUM_2503,round(fob_onbc_2503,0), round((conv_rat_2503 *
fob_infc_2503),0),round(fob_onbc_2503,0)-round((conv_rat_2503 * fob_infc_2503),0) from
vffm.vffm_pep_2503 where round(fob_onbc_2503,0) - round((conv_rat_2503 * fob_infc_2503),0) > 1
AND SUBSTR(FILE_NUM_2503,15,2) IN ('12%','13%','14%')

43. **Query:** To extract records where the rate awarded under VKGUY scheme was at 7% instead of reduced rate of 5% where the shipping bill was utilised in other schemes

SQL: select * from vffm.vffm_pep_2503 WHERE SUBSTR(FILE_NUM_2503,13,4) IN
('AM12','AM13','AM14','AM12S','AM13S','AM14S') AND VFFM_CODE_2503 LIKE '%37A%'AND
CHK_UTIL_2503 = 'Y' AND CHK_RATE_2503='Y' AND RATE_AWD_2503 = '7'

44. **Query:** To extract records where the rate awarded under VKGUY scheme was at 5% instead of reduced rate of 3 % where the shipping bill was utilised in other schemes.

SQL: select lic_numb_1500,lic_catg_1500,file_num_2503,vffm_code_2503, shbi_num_2503,
prd_code_2503, itch_cod_2503, vffm_rat_2503, item_des_2503, rate_awd_2503,FOB_ONBC_2503,
act_entl_2503, shbi_dat_2502, leo_date_2502 from vffm.vffm_pep_2503,
vffm.vffm_pes_2502,licm.lic_mast_1500 WHERE (shbi_num_2503=shbi_num_2502) and
(file_num_2503=file_num_1500) and (file_num_2502=file_num_1500) and
SUBSTR(FILE_NUM_2503,13,5) IN
('AM12','AM13','AM14','AM12S','AM13S','AM14S','AM12SS','AM13SS','AM14SS') and VFFM_CODE_2503
LIKE '%37A%' AND CHK_UTIL_2503 = 'Y' AND RATE_AWD_2503 = '5' AND VFFM_RAT_2503='5';

45. **Query:** To extract records where updation of exchange rate was effected in the Shipping Bill. The check was exercised in respect of US Dollar applicable for the period 30/07/2013 till 02/08/2013. The rate applicable on 30/07/2013 was Rs. 58.65 and on 02/08/2013 was Rs.60.55.

SQL: SELECT

```
LEO_DATE_2502,FILE_NUM_2503,FILE_NUM_2502,CONV_RAT_2503,SHBI_NUM_2503,FOB_F
CCD_2503,SHBI_NUM_2502,SHBI_DAT_2502 FROM
VFFM.VFFM_PEP_2503,VFFM.VFFM_PES_2502 WHERE SHBI_NUM_2503=SHBI_NUM_2502
AND FOB_FCCD_2503='USD' AND FILE_NUM_2503=FILE_NUM_2502 AND SHBI_DAT_2502
BETWEEN '2013-07-19' AND '2013-08-01' AND CONV_RAT_2503<>58.65 ORDER BY
LEO_DATE_2502,SHBI_DAT_2502;
```

46. **Query:** To extract records where FOB value allowed in foreign currency terms is greater than FOB realised in foreign currency terms

SQL: Select * from vffm.vffm_pep_2503 where fob_infc_2503 < vffm_alow_2503

47. **Query:** To extract records where duty credit rate applied in respect of VKGUY Scheme(37A) is correct. Rates applicable (3%, 5%,7% = Regular : 2%=Supplementary)

SQL: SELECT COUNT(*),ENTL_PER_2503 FROM VFFM.VFFM_PEP_2503 WHERE SUBSTR(FILE_NUM_2503,13,4) IN ('AM12','AM13','AM14','AM12S','AM13S','AM14S') AND VFFM_CODE_2503 LIKE '%37A%' GROUP BY ENTL_PER_2503

48. **Query:** To extract records where RATE AWARDED IS GREATER THAN ENTITLED percentage of duty credit under VKGUY 37A

SQL: SELECT VFFM_RAT_2503,ENTL_PER_2503,RATE_AWD_2503 FROM VFFM.VFFM_PEP_2503 WHERE SUBSTR(FILE_NUM_2503,13,4) IN ('AM12','AM13','AM14','AM12S','AM13S','AM14S') AND VFFM_CODE_2503 LIKE '%37A%' AND RATE_AWD_2503 > ENTL_PER_2503;

49. **Query:** To extract records where duty credit was granted at 7% of FOB , to check whether the items in the field item_des_2503 was eligible for duty credit at 7% of FOB or not with appendix 37A. The query was run and all items with rate of 7% was called out and imported into excel and manually checked for eligibility

SQL: select lic_numb_1500,lic_catg_1500,file_num_2503,vffm_code_2503, shbi_num_2503, prd_code_2503, itch_cod_2503, vffm_rat_2503, item_des_2503, rate_awd_2503,FOB_ONBC_2503, act_entl_2503, shbi_dat_2502, leo_date_2502 from vffm.vffm_pep_2503, vffm.vffm_pes_2502,licm.lic_mast_1500 WHERE (shbi_num_2503=shbi_num_2502) and (file_num_2503=file_num_1500) and (file_num_2502=file_num_1500) and SUBSTR(FILE_NUM_2503,13,5) IN ('AM12','AM13','AM14','AM12S','AM13S','AM14S','AM12SS','AM13SS','AM14SS') and VFFM_CODE_2503 LIKE '%37A%' AND VFFM_RAT_2503='7'

50. **Query:** To extract records where BRC OR BG/LUT is AVAILABLE FOR ISSUE OF LICENCE FOR THE YEAR 2011-12,2012-13,2013-14 in vffm schema

SQL: SELECT BGL_TYPE_2501, CHE_BACE_2501 FROM VFFM.VFFM_POE_2501 WHERE BGL_TYPE_2501=CHE_BACE_2501 AND CHE_BACE_2501='N' AND SUBSTR(FILE_NUM_2501,15,2) IN ('12','13','14')

51. **Query:** To extract records where duplicates shipping bill exists in same date in VFFM_PES_2502 Table

SQL: select COUNT(*),SHBI_NUM_2502,SHBI_DAT_2502,SHBI_PRT_2502 from (SELECT * FROM VFFM_PES_2502 WHERE SUBSTR(FILE_NUM_2502,15,2)='13' AND (VKUY_FOB_2502+FPS_FOB_2502+FMS_FOB_2502+MLFP_FOB_2502)>0) GROUP BY SHBI_NUM_2502,SHBI_DAT_2502,SHBI_PRT_2502 HAVING COUNT(*)>=2

52. **Query:** To extract records where multiple IEC issued for same PAN

SQL: Select COUNT(*),EXPO_PAN_211 from IEC.IEC_MAST_211 WHERE SUBSTR(FILE_NUM_211,13,4) IN ('AM12','AM13','AM14') GROUP BY EXPO_PAN_211 HAVING COUNT(*)>=2

53. **Query:** To extract records where Duplicate IEC issued

SQL: select COUNT(*),IEC_NUMB_211 from IEC.IEC_MAST_211 GROUP BY IEC_NUMB_211 HAVING COUNT(*)>=2

54. **Query:** To extract records where duplicate entries of PAN exists

SQL: select COUNT(*),EXPO_PAN_211 from IEC.IEC_MAST_211 GROUP BY EXPO_PAN_211 HAVING COUNT(*)>=2

Sample Logical Queries

1. E-Com Application:

- To check the correctness of linking of SBs done automatically, eBRC uploaded by banks in cases of Chapter -3 Schemes [VKGUY, FMS, FPS, MLFPS and DEPB].

2. Checks common to DEPB, VKUY, FPS, FMS, MLFPS schemes:

- To check if products prohibited by a notification or public notice for exports at the time of exports were barred by the systems or not
- To check if the same SB is not utilized again in another claim under the same scheme or different scheme under chapter III.
- Correctness of correlation between BRC & SBs
- Correctness of correlation of product code with RITC(Revised Indian trade classification)& item description
- Correctness of correlation of DEPB schedule serial no. and application of valid rate with reference to LEO date
- Correctness of calculations of applicable rates for DEPB / VKUY/ FPS
- Whether value cap applied correctly for DEPB
- In respect of products where rate of credit entitlement under DEPB Scheme comes to 10% or more, amount of credit against each such export product restricted to 50% of Present Market Value (PMV) of export product.
- Correctness of calculations Application of late cut
- Correctness of applicable exchange rate
- treatment where currency of actual realization and invoice currency were different
- Whether commission limited to 12.5%
- item claimed under VKUY eligible as per Appendix-37A
- In case of VKGUY
 - (i) Duty credit allowed correctly i.e.@ of 5 % for products listed in Appendix 37A(additional 2 % for items listed in Table 1, Appendix 37A)
 - (ii) @ of 3 % where benefits also availed on(additional 2 % for items listed in Table 1, Appendix 37A) drawback (at higher than1%), specific DEPB rate and Adv Auth or DFIA
- item claimed under FPS/MLFPS eligible as per Appendix-37C
- table 1 & 2 of Appendix – 37C are eligible for Duty credit at 3% and Exporter of products under table 3 of the Appendix at 1% under FMS scheme
- Only eligible country of export were under FMS/ MLFPS

3. Checks in respect of DES- Advance Authorisation, DFIA

- Input-output quantity correlation is correct wrt SION

- To check the correctness of details of licenses which were not discharged given at the time of application
- If Invalidation of license and issue of ARO, amendment sheets are transmitted to Customs/ jurisdictional central excise authority in time.
- if Co-licensee details are communicated to customs
- To see if supporting manufacturer data are valid
- To check if valid Deemed export details are entered
- All figure shown in ANF 4F are same as shown in SBs & BRCs
- Whether the date of any SB is prior to date of submission of Auth. Application

4. Verification of the data for in respect of EPCG

- Applicant is required to give details of licenses which were not discharged at the time of application. To check if correctness of same is verifiable from data and comment accordingly.
- Validity period of licenses is correct
- Check the correctness of AEP of previous three year for the same and similar products for all the applications filed in a year.
- Monitoring of application for redemption after expiry of EO period.

5. SFIS Scheme

- Credit allowed only for services listed in Appendix 41of HBP.
- Duty credit restricted to 10 % of FE.
- Free Foreign Exchange Earned $\geq$ 10 lakh/5 lakh
- Exports pertain to the relevant year only for which claims are being made (exports of two different F.Y.s not allowed in single application).
- Late cut applied where application submitted after 12 months from the end of Qr/Half Year/Year.
- Credit calculated on Net Foreign Exchange(NFE) for exports after 18TH April, 2013

6. General checks

- A status certificate & RCMC remain valid for 5 years from 1st April of the year of issuance. (3.2.2 of HBP). Check that RCMC/ Status Certificate were valid and not suspended/ cancelled when applied for relevant scheme e.g. SHIS.
- If exporter was allowed to apply for a license after put on Denied Entity List (DEL) without getting removed from the list.
- Whether license holders are put on alert before/ after due date of submission of any statutory documents such as import details, BRC, application for redemption etc.
- Applications were processed within due date as prescribed in Policy for different schemes.

7. Audit checks on e-BRC Application:

- test data for existence of various validations being claimed as per error codes generated while accepting e-BRC from banks
- To verify that if any original xml file transmitted by the bank was tempered, with the help of public key and hash code generated and stored
- *Whether invalid records are also inserted in database. Can these records be utilized to verify the correctness of records? e.g. There is requirement of existence of SB data before upload of e-BRC as the SB no & currency in both is matched before accepting Where SB no. is entered manually exporter may purposely enter one extra alphabet along with SB no. In such cases system may not show that SB*

was already utilized in any other license

- a BRC record is rejected for want of SB no, which was inserted later, to see in audit whether bank send fresh e-BRC for the SB
- whether there is mismatch between SB value / currency code stored in database and the similar details in the e-BRC
- To check mismatch between SB no and its IEC stored in database with BRC's SB no & IEC.
- In e-BRC data exporter are required to enter freight, insurance and commission portion of the realized FC, whereas similar data already exist in SBs. To verify if any discrepancies.
- Cases where term of invoice in SB is CIF but exporter did not record freight charges/ insurance charges, or recorded unreasonable figure or figures grossly mismatched with SB data
- Check that the all the amounts viz invoice value realised, freight, commission and insurance all should be in same currency.

IEC application

- To verify whether more than one IEC issued / allowed against single PAN (Para 2.9 of HBP).
- PAN holder name and IEC holder name are same.
- Whether PAN holder status and IEC holder status are same.
- Necessary documents are being uploaded through system and same is verifiable in audit
- Check from ICES data if IEC is in use after the date of cancellation as found from DGFT data.
- Check from ICES data for use of an IEC by more than one exporter

RCMC Application

- RCMC is valid for 5 years (Para of FTP) from 1st April of issuing year, to check data for the validation.
- Check for validity of data according to various error code generated (listed in Annexure-6 of guidelines).

EFT (Electronic Fund Transfer)

- There is provision (2.70.3 of HBP) for discount on Application fee to be collected e.g. Digitally Signed Appln (25%), Payment through EFT (25%), for both 50% discount. To verify data for the appropriate discount applied.
- Proper fee have been collected and System exists for re-conciliation of all the fee collected