EXP. NO :- 1
Date: 5/1/22

Data Definition Commands

Aim :- To create database using Data Definition commands of SQL queries in RDBMS.

Data Definition Commands :-

DDL actually consists of SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

Examples of DDL commands :-

* CREATE : It is used to create the database (or) its objects
* DROP : It is used to delete objects from the database.
* ALTER :- It is used to alter the structure of the database.
* TRUNCATE :- It is used to remove all record from a table, including all spaces allocated for records are removed.
* COMMENT :- It is used to add elements to the data dictionary
* RENAME :- It is used to rename an object existing in database.

CREATE TABLE:-

It is used to create a table.

Rules:-

1. Oracle reserved words cannot be used.
2. Underscore, numericals, letters are allowed but not blank space.
3. Maximum length for the table name is 30 characters.
4. Different tables should not have same name.
5. We should specify a unique column name.
6. We should specify proper data type along with width.

Syntax:-

SQL> create table tablename (column_name1 data_type constraints,
column_name2 data_type constraints ---)

DESC:-

This is used to view the structure of the table.

Creating new table from existing table:-

Syntax:-

Create table new_table_name ASSELECT column1, column2, ----
FROM existing_table_name WHERE ----;

DROP TABLE:- It will delete the table.

Syntax:- SQL> DROP TABLE <TABLE NAME>;

TRUNCATE TABLE:-

If there is no further use of records stored in a table and the structure has to be retained then the records alone can be deleted.

Syntax:- TRUNCATE TABLE <TABLE NAME>;

ALTER COMMAND:-

It is used to

1. Add a new column.
2. Modify the existing column definition.
3. To include (or) drop integrity constraint.

ADD COMMAND:- Add the new column to the existing table.

Syntax:- SQL>Alter table tablename add (attribute datatype(size));

Modify command:- It is used to modify existing column definition.

Syntax:- SQL>alter table <table name>modify (columnname constraint);

SQL>alter table <tablename> modify (column name datatype);

COMMENT:- It can be written in the following three formats.

1. single line comments.
2. multi line comments
3. In line comments.

1. Single Line Comments:- Comments starting and ending in a single line are considered as single line comments.

Line starting with '--' is a comment and will not be executed.

Syntax:-

```
-- single line comment.
-- another comment.

SELECT * FROM Customers;
```

2. Multi Line Comments:- Comments starting in one line and ending in different line are considered as multi line comments. Line starting with '/*' is considered as starting point of comment and are terminated when '*/' is encountered.

Syntax:-

```
/* multi line comment
another comment */
SELECT * FROM customers;
```

3. In-line comments:- In-line comments are an extension of multi-line comments, comments can be stated in b/w the statements and are enclosed in b/w '/*' & '*/'.

Syntax:-

```
SELECT * FROM /* customers; */
```

RENAME SYNTAX:-

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

Syntax to rename a table:-

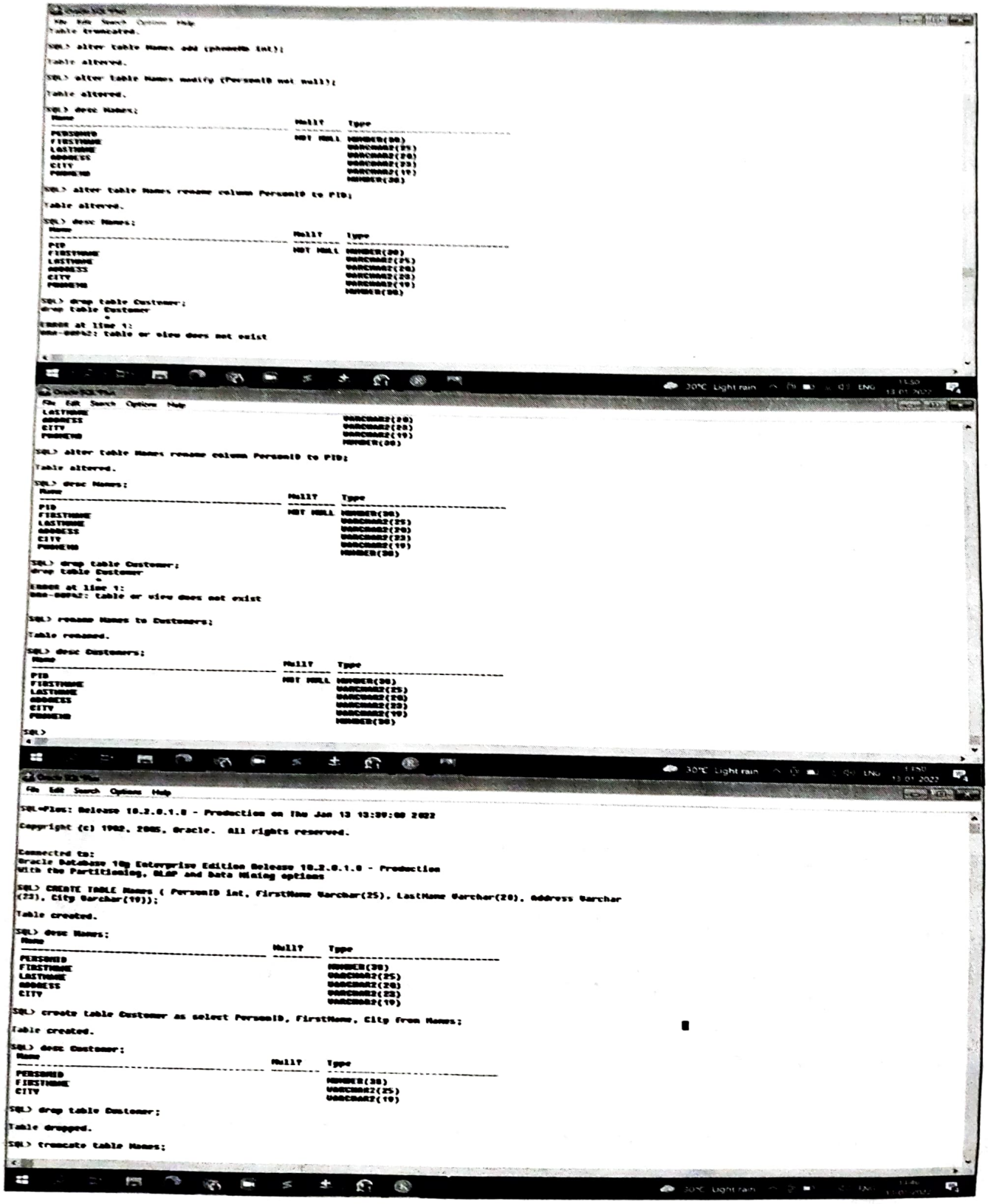
```
RENAME old_table_new TO new_table_name;
```

Program 1:

```
SQL> connect
Enter user-name: system
Enter password: admin
Connected.

SQL> create table emp(id number(10),name varchar(10));

Table created.

SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)

SQL> alter table emp add(dept varchar(10));

Table altered.

SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)
 DEPT                                               VARCHAR2(10)

SQL> alter table emp modify dept varchar(20);

Table altered.
SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)
 DEPT                                               VARCHAR2(20)


SQL> alter table emp drop column dept;

Table altered.

SQL> desc emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
```

```
 NAME          VARCHAR2(10)

SQL> alter table emp rename to emp1;

Table altered.

SQL> desc emp1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)

SQL> select * from emp1;

no rows selected

SQL> alter table emp1 add(dept varchar(10));

Table altered.

SQL> desc emp1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)
 DEPT                                               VARCHAR2(10)
```

Experiment 1:



Result :- Thus, the database is created using data definition commands of SQL queries in RDBMS.

Exp. No: 2

Date:-12/1/22

SQL Data Manipulation Language Commands

Aim :- To create database using Data manipulations commands for inserting, deleting, updating, retrieving tables, transaction control st.

Data Manipulation Commands :- The statement access and manipulate data in existing tables.

Examples :-

INSERT

SELECT

UPDATE

DELETE

INSERT COMMAND :- It is used to insert values into table.

INSERT a single record into table :

Syntax :- SQL> insert into <table name> values (value list).

INSERT more than a record into person table using a single insert command :-

SQL> insert into persons values (&pid, '&firstname', '&lastname', '&address', '&city');

Skipping the fields while inserting :-

SQL > insert into persons (pid, firstname) values (500, 'prabhu');

SELECT Commands :- It is used to retrieve inf. from the table. It is generally referred to as querying the table. we can either display all columns in a table (or) only specify column from the table.

Syntax :-

SQL> select * from table name;

The retrieval of specific columns from a table :-

It retrieves the specified columns from a table.

Syntax :- SQL> select column_name 1, ... , column_name from table name;

Elimination of duplicates from the select clause:-

It prevents retrieving the duplicated values. Distinct keyword is to be used.

Syntax:- SQL>Select DESTINCT col1, col2 from table name;

Select command with where clause:-

To select specific rows from a table we include 'where' clause in the select command. It can appear only after the 'from' clause.

Syntax:- SQL> select column_name 1, ---- column_name n from table name where condition;

Select command with order by clause:-

syntax:- SQL> select column_name 1, ---- column_name n from table name where Gnditim orderby column name;

Select command to create a table:-

syntax:- SQL> create table table name as select * from existing table name;

Select command to insert records :-

Syntax :- SQL> insert into table name (select columns from existing_tablename);

Select command using IN keyword :-

Syntax :- SQL> select column_name 1, ---, column_name n from table name where column name IN (value 1, value 2);

Select command using BETWEEN keyword :-

Syntax :- SQL> select column_name 1, ----, column_name n from table name where columnname BETWEEN value 1 AND value 2;

Select command using pattern :-

Syntax :- SQL> select column_name 1, ---, column_name n from table name where column name LIKE '%. or _';

Renaming the fieldname at the time of display using select statement :-

syntax :- SQL> select old_column_name new_column_name from table name where condition ;

SELECT command to retreive null values:-

Syntax:- SQL> Select column_name from table name where column_name is NULL;

UPDATE COMMAND:-

Syntax:- UPDATE table_name SET column_name = value [, column_name = value] ... [WHERE condition];

DELETE COMMAND:-

SYNTAX:- SQL> Delete from table where condition;

Program 2:

```
SQL> insert into emp1(id,name,dept) values (1,'aaa','cse');

1 row created.

SQL> select * from emp1;

        ID NAME     DEPT
---------- -------- --------
         1 aaa      cse

SQL> insert into emp1 values (2,'bbb','cse');

1 row created.

SQL> select * from emp1;

        ID NAME     DEPT
---------- -------- --------
         1 aaa      cse
         2 bbb      cse

SQL> insert into emp1 values(&id,'&name','&dept');
Enter value for id: 3
Enter value for name: ccc
Enter value for dept: cse
old   1: insert into emp1 values(&id,'&name','&dept')
new   1: insert into emp1 values(3,'ccc','cse')

1 row created.

SQL> select * from emp1;

        ID NAME     DEPT
---------- -------- --------
         1 aaa      cse
         2 bbb      cse
```

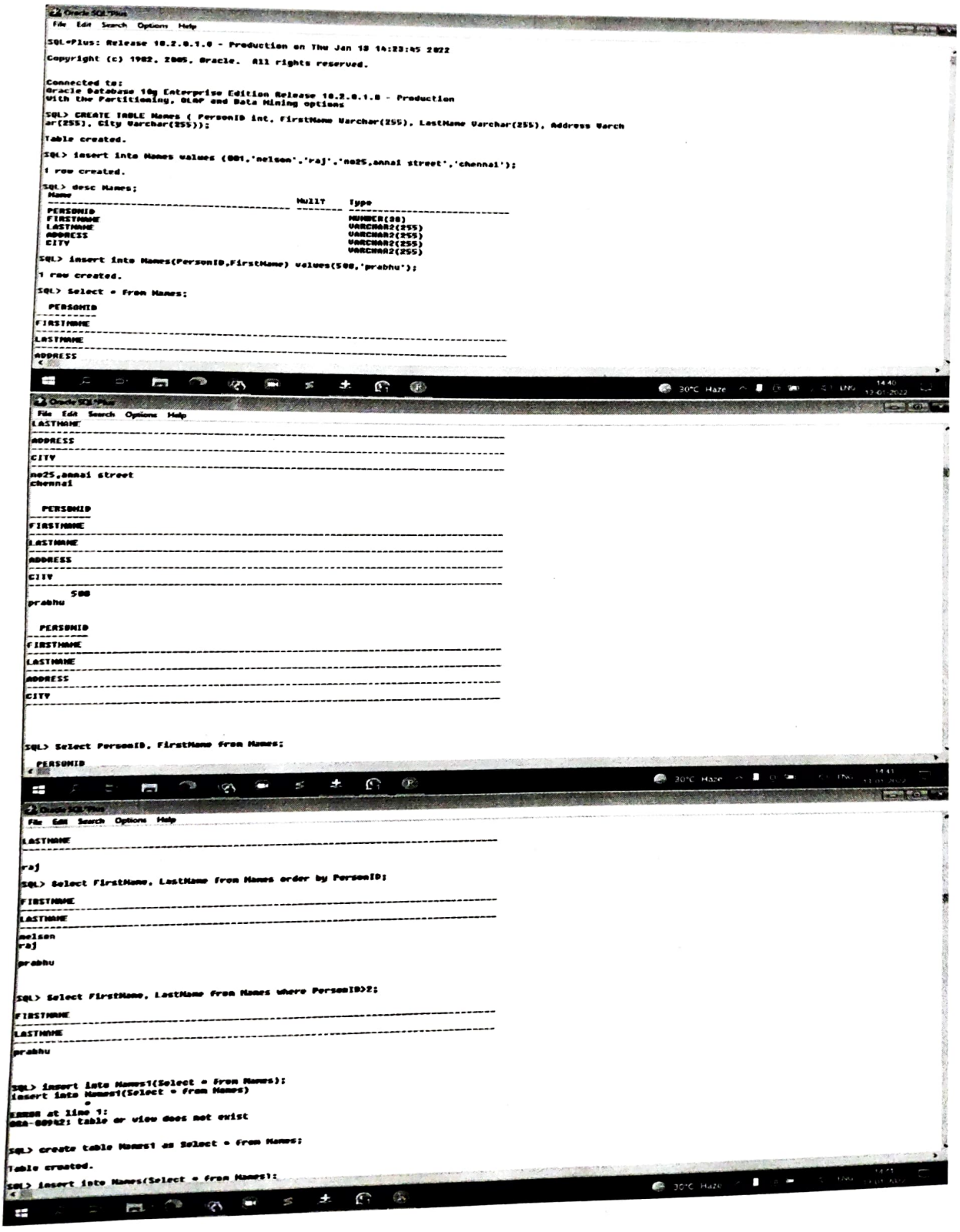
```
         3 ccc       cse

SQL> insert into emp1 values(&id,'&name','&dept');
Enter value for id: 4
Enter value for name: ddd
Enter value for dept: cse
old   1: insert into emp1 values(&id,'&name','&dept')
new   1: insert into emp1 values(4,'ddd','cse')

1 row created.

SQL> /
Enter value for id: 5
Enter value for name: eee
Enter value for dept: cse
old   1: insert into emp1 values(&id,'&name','&dept')
new   1: insert into emp1 values(5,'eee','cse')

1 row created.

SQL> select * from emp1;

        ID NAME       DEPT
---------- ---------- ----------
         1 aaa        cse
         2 bbb        cse
         3 ccc        cse
         4 ddd        cse
         5 eee        cse
SQL> update emp1 set name='BBB' where id=2;

1 row updated.

SQL> select * from emp1;

        ID NAME       DEPT
---------- ---------- ----------
         1 aaa        cse
         2 BBB        cse
         3 ccc        cse
         4 ddd        cse
         5 eee        cse
SQL> update emp1 set name='CCC',dept='ece' where id=3;

1 row updated.

SQL> select * from emp1;
```

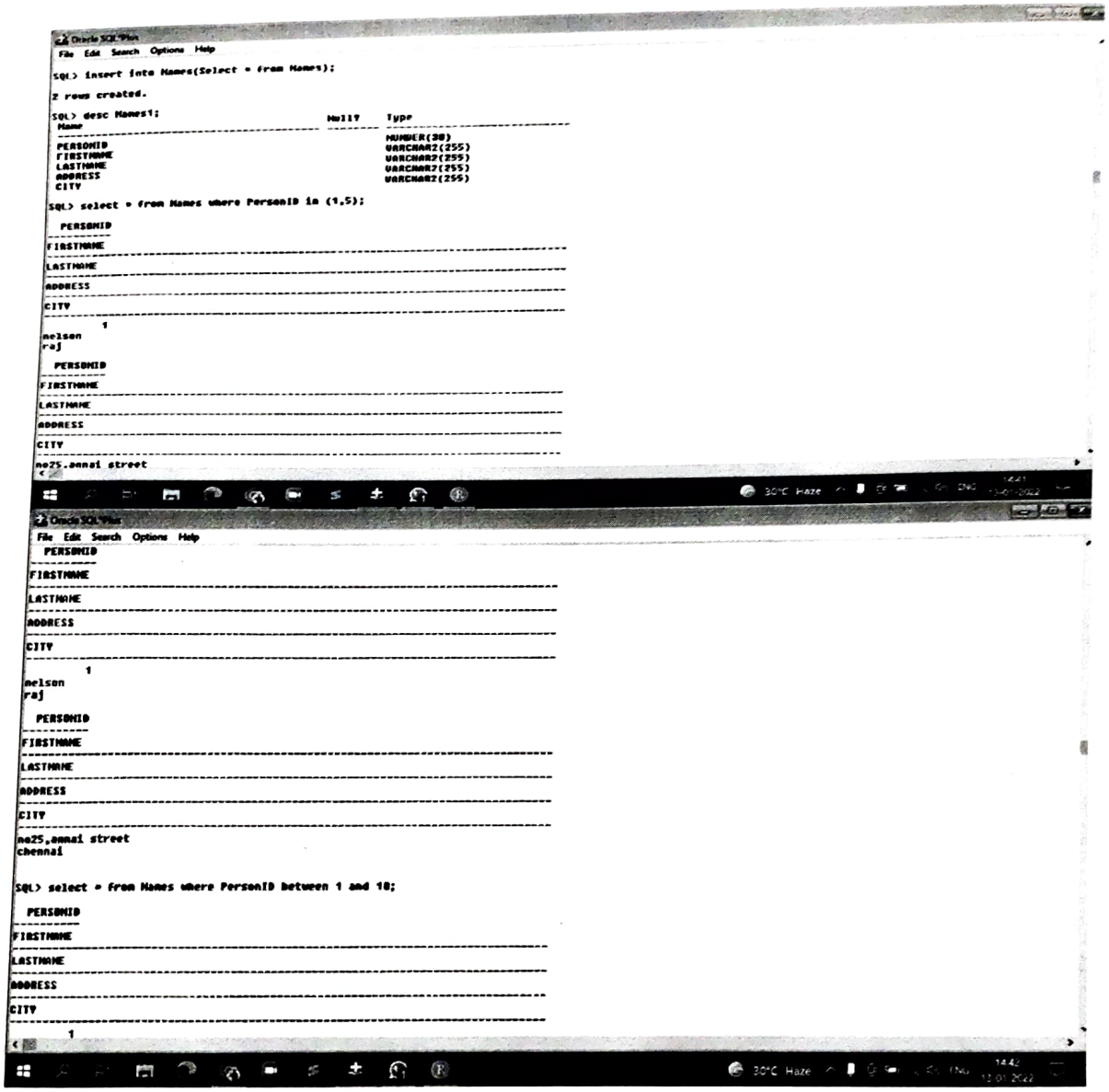
```
        ID NAME      DEPT
---------- ---------- ----------
         1 aaa       cse
         2 BBB       cse
         3 CCC       ece
         4 ddd       cse
         5 eee       cse
SQL> update emp1 set name='aaa';

5 rows updated.

SQL> select * from emp1;

        ID NAME      DEPT
---------- ---------- ----------
         1 aaa       cse
         2 aaa       cse
         3 aaa       ece
         4 aaa       cse
         5 aaa       cse
SQL> create table emp2(id number(10),name varchar(10),dept varchar(10));

Table created.

SQL> insert into emp2 select * from emp1;

5 rows created.

SQL> select * from emp2;

        ID NAME      DEPT
---------- ---------- ----------
         1 aaa       cse
         2 aaa       cse
         3 aaa       ece
         4 aaa       cse
         5 aaa       cse
SQL> delete from emp2 where id=1;

1 row deleted.

SQL> select * from emp2;

        ID NAME      DEPT
---------- ---------- ----------
         2 aaa       cse
```

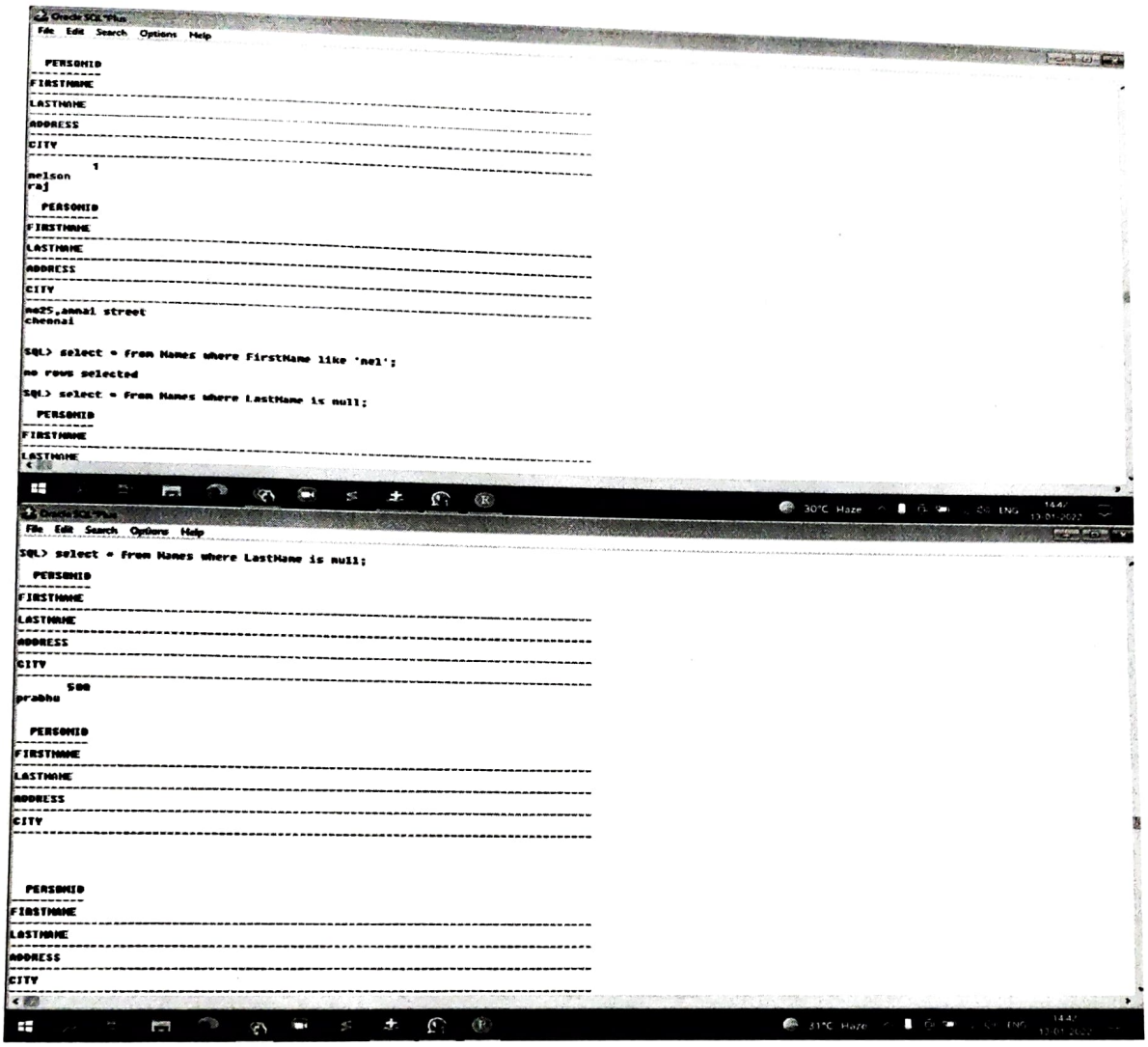
```
         3 aaa      ece
         4 aaa      cse
         5 aaa      cse
SQL> delete from emp2 where dept='cse';

3 rows deleted.

SQL> select * from emp2;

        ID NAME      DEPT
---------- ---------- ----------
         3 aaa      ece
SQL> delete from emp2;

1 row deleted.

SQL> select * from emp2;

no rows selected
SQL> select * from emp2;

no rows selected

SQL> desc emp2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)
 DEPT                                               VARCHAR2(10)
SQL> drop table emp2;

Table dropped.

SQL> select * from emp2;
select * from emp2
         *
ERROR at line 1:
ORA-00942: table or view does not exist
SQL> select * from emp1;

        ID NAME      DEPT
---------- ---------- ----------
         1 aaa      cse
         2 aaa      cse
         3 aaa      ece
         4 aaa      cse
         5 aaa      cse
```

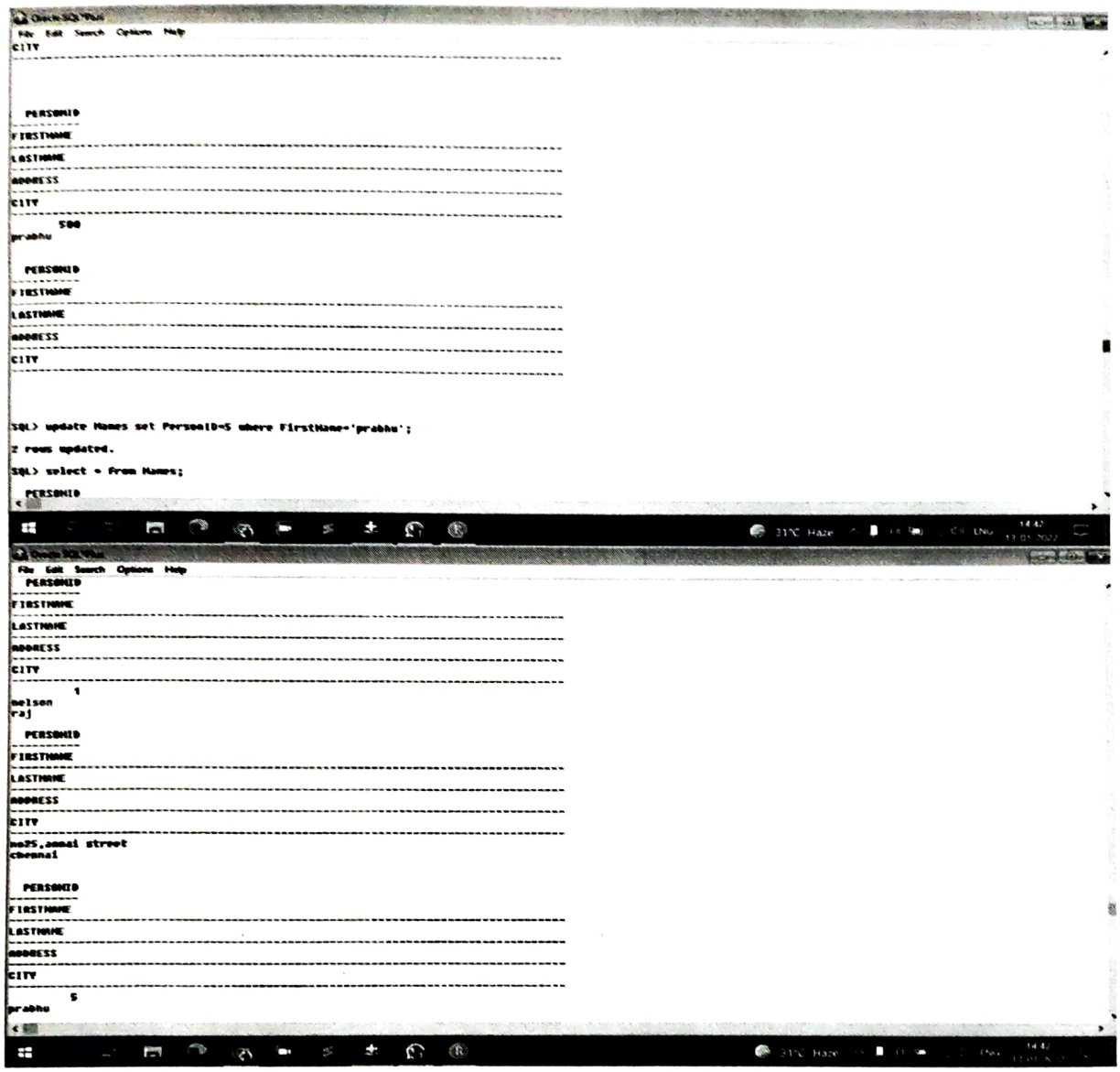
```
SQL> truncate table emp1;

Table truncated.

SQL> select * from emp1;

no rows selected

SQL> desc emp1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER(10)
 NAME                                               VARCHAR2(10)
 DEPT                                               VARCHAR2(10)

SQL> drop table emp1;

Table dropped.

SQL> select * from emp1;
select * from emp1
       *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> desc emp1;
ERROR:
ORA-04043: object emp1 does not exist
```

Experiment 2:




Oracle SQL*Plus
File Edit Search Options Help
SQL> insert into Names(Select * from Names);
2 rows created.
SQL> desc Names1;
Name Null? Type
PERSONID NUMBER(38)
FIRSTNAME VARCHAR2(255)
LASTNAME VARCHAR2(255)
ADDRESS VARCHAR2(255)
CITY VARCHAR2(255)
SQL> select * from Names where PersonID in (1,5);
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
1
nelson
raj
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
no25,annai street
Oracle SQL*Plus
File Edit Search Options Help
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
1
nelson
raj
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
no25,annai street
chennai
SQL> select * from Names where PersonID between 1 and 10;
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
1
30°C Haze
14:41
12-01-2022
30°C Haze
14:42
12-01-2022


Oracle SQL*Plus
File Edit Search Options Help
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
1
nelson
raj
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
no25,annai street
cheenai
SQL> select * from Names where FirstName like 'nel';
no rows selected
SQL> select * from Names where LastName is null;
PERSONID
FIRSTNAME
LASTNAME
30°C Haze
Oracle SQL*Plus
File Edit Search Options Help
SQL> select * from Names where LastName is null;
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
500
prabhu
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
31°C Haze


File Edit Search Options Help
CITY
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
500
prabhu
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
SQL> update Names set PersonID=5 where FirstName='prabhu';
2 rows updated.
SQL> select * from Names;
PERSONID
31°C Haze
File Edit Search Options Help
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
1
nelson
raj
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
no25,annai street
chennai
PERSONID
FIRSTNAME
LASTNAME
ADDRESS
CITY
5
prabhu
31°C Haze

Result :- Thus, the database has been created ~~and~~ using Data manipulations commands for inserting, deleting, updating, retrieving tables.

EXP. NO :- 3

Date :- 19/1/22

SQL Data Control Language Commands & Transaction control Language

Aim :- To create database using SQL DCL commands & TCL commands.

Data control Language :-

It is a syntax similar to a computer programming language used to control access to data stored in a database. It is a component of structured query Language.

Example of DCL commands include :-

* GRANT
* REVOKE

* GRANT :-

syntax : GRANT privilege_name on objectname to user;

* REVOKE :-

syntax : REVOKE privilege_name on objectname from user;

Transaction control command (TCL):-

TCL commands are used to manage transactions in the database.

Ex. of TCL:-

(i). commit.

(ii). Rollback

(iii). Save point.

(i). commit:- commit command saves all the work done.

Syntax:- commit;

(ii). Rollback:- Rollback command restores database to original since the last commit.

Syntax:- Roll Back TO SAVE POINT <save point_name>;

(iii). Save point:-

Syntax:- Save point <save point_name>;

Result:- Thus, the database has been created using DCL commands & TCL commands.

EXP. NO:-4

Date:- 9/2/22.

Inbuilt Functions in SQL on sample exercise

Aim :- To execute inbuilt functions in SQL. on sample exercise.

SQL Function :-

* Set the instructions which will do some specific task.
* It may (or) maynot return the value.
* It's the module.

Types :-

Single row Function :-

1. Data Function.
2. Numeric Function (or) Arithematic (or) Mathematical Function.
3. character function.
4. General Function (or) Miscellaneous Function.

Group Function :-

1. Max ( )
2. Min ( )
3. Sum ( )
4. Avg ( )
5. Std dev ( )
6. varience ( )

Result:- Thus, all the inbuilt functions in SQL has been executed successfully.

EXP. NO :-5 Construct ER Diagram for application

Date : 16/2/22

Aim :- To construct an E-R Diagram for Bank Managment system to be created & constructed to a database.

Steps for drawing ER Diagram :-

* First, identify the entities in your database. In this case, we have three entities.
* The 2nd step involves identifying the r/w b/w the selected entities.
* The 3rd step involves identifying cardinalities.
* The 4th step involves identifying entity attributes. Make sure that every attribute is mapped to only one entity : assign modifiers for those that belong to more than one.
* Once you have identified the entities, relationships, cardinalities and attributes, you can now create your ER diagram.

Result :- Hence, we have successfully drawn the ER diagram.

EXP. NO:- 6

Date:- 23/2/22

Nested Queries on Sample Exercise

Aim :- To create a database using Nested Queries in DBMS.

Theory :-

* Nested Queries :- Nesting of Queries one within another is known as a nested queries.

Syntax :- SQL > select empid, emp name, job name from.

* Sub Query :- The Query within another is known as subquery. A statement containing subquery is called parent statement. The rows returned by subquery are used by parent statement.

Ex :- SQL > select * from EMP1 where empid in (select empid from EMP1)

* Corelated SubQuery :- A subquery is evaluated once for the entire parent, where as a correlated subquery is evaluated once per row processed by the parent statement.

Ex :- Select * from EMP1 where sal > (select avg(sal) from EMP1 where empid = empid);

Result :- Thus, the database table using Nested queries have been written and executed successfully.

EXP. NO:-7

Date: 2/3/22

Join Queries on sample Exercise

Aim :- To create a database using join queries in DBMS.

Theory :- Join concept is to combine data spread across tables. A join is actually performed by 'where' clause which combines specified rows of tables.

Syntax :- select columns from table 1, table 2 where logical exp;

Types of joins :-

1. Simple Join :- Most common type of join. It retrieves rows from two tables having column in common.

   Ex:- select * from emp1;

(a). EQUI JOIN: A Join which is based on equalities.

(b). NON EQUI JOIN:- It specifies r/w b/w columns belonging to different tables by making use of relational operators other than '='.

2. Self-Join :- Joining of table itself is known as self join. It joins one row in table to another. It can compare each row of table to itself & also with other rows of same table.

Ex:- select * from EMP1, DEPT1 where sal>=(select avg(sal))

3. Outer_Join :- (+) represents outer join. Returns all rows returned by simple join as well as rows from one table that don't match any row from table.

Ex:- EMP1, DEPT1 & their columns.

DIFFERENT JOINS :-

* Inner Join :- Returns all rows when their is atleast one match in both tables.

Ex:- SQL> select Emp1.empname, DEPT1.dname from EMP1. inner join DEPT1 on EMP1.deptid=DEPT1.deptid;

* Left Join :- Return all rows from left table & matched rows from right table.

Ex:- Sql> Select DEPT1.dname, DEPT1.deptid, EMP1.deptid, EMP1. empname from EMP1 left join DEPT1 on DEPT1.deptid=EMP1.deptid;

Right Join :- Returns all rows from right table, and matched rows from left table.

Ex :- SQL>select DEPT1.dname, DEPT1.deptid, EMP1.deptid, EMP1.empname. from EMP1 right join DEPT1 on DEPT1.deptid= EMP1.deptid;

NULL JOIN :- Returns all rows when their is a match in one of tables.

Ex :- SQL> select DEPT1.dname, DEPT1.deptid, EMP1 deptid, EMP1.empname from EMP1 full join DEPT1 on DEPT1.deptid= EMP1.deptid;

Result:- Thus, the database table using join query have been written and executed successfully.

EXP. NO: 8

Set operations and Views

Date: 2/3/22.

Aim :- To perform set operations and views using sql plus.

SET operations :-

Theory :-

1. UNION: It returns a table which consists of all rows either appearing in result of <query 1> (or) <query 2>.

   Ex :- Select regno from T1 Union select regno from T2.

2. Intersect :- It returns all rows that appears in both results <query 1> & <query 2>.

   Ex :- Select regno from T1 intersect select regno from T2;

3. MINUS :- It returns those rows which appear in result from <query 1> but not in result of <query 2>.

   Ex :- select regno from T1 minus select regno from T2;

4. UNION ALL:- It returns all rows selected by either query, including all duplicates.

5. INTERSECT ALL:- It is same as intersect all distinct rows selected by both queries.

VIEWS:-

A view is nothing more than a SQL statement that is stored in database with an associated name. A view is actually a composition of table in form of a predefined SQL query. A view can contain all rows of a table. A view can be created from one (or) many tables which depends on written SQL query to create a view.

Creating Views:- Database are created using create view statements.

Syntax:-

```
create view view_name as
select column1, column2, ...
FROM table_name
where [condition];
```

CHECK OPTION:-

Syntax:

```
create view customers_views as
select column1, column2, --
FROM table_name
where [condition];
with check = option;
```

Updating a View:-

Conditions:

* select clause may not contain keyword distinct.
* select clause may not contain set functions & operators.
* May not contain order by clause.
* FROM clause maynot contain an order by clause.
* WHERE clause maynot contain subqueries.
* Query may not contain group by (or) HAVING.
* Calculated columns may not be updated.
* ALL NOT NULL columns from base table must be included in view in order for insert query to function.

Inserting rows into a view :-

Rows of data can be inserted into view same Rules that apply to UPDATE & INSERT apply to delete command.

Dropping views :- Drop view if it is no longer needed.

Syntax :- Drop View view_name ;

Result :- Thus, the set operations and various views were executed successfully.