SQL DATA CONTROL LANGUAGE COMMANDS AND TRANSACTION CONTROL COMMANDS TO THE SAMPLE EXERCISES

AIM :- To create Database to perform DATA CONTROL LANGUAGE and Transaction control commands.

Transaction Control Statements

Transaction control Language (TCL) commands are used to manage transactions in the database.

Examples of TCL :-

(i) Commit
(ii) Roll back.
(iii) Save Point.

1) Commit :- Commit command saves all the work done
Syntax :- Commit;

2) Roll back :- Rollback command restores database to original since the last Commit
Syntax :- ROLLBACK TO SAVEPOINT <savepoint_name>;

3) Savepoint :-
Syntax :- SAVEPOINT <savepoint_name>;

RESULT :-

Hence, the above transaction control commands has been implemented successfully.

INBUILT FUNCTIONS IN SQL

AIM :-

To write a program on Inbuilt functions in SQL

SYNTAX :-

MIN() : Returns the smallest value of the selected column.

Syntax :- SELECT MIN (column_name).

MAX() : Returns the largest value of the selected column.

Syntax : SELECT MAX (column_name)

COUNT() : Returns the number of rows that matches a specified criterion.

Syntax :- SELECT COUNT (column_name).

AVG() : Returns the average value of a numeric column.

Syntax :- SELECT AVG (column_name)

SUM() : Returns the total sum of a numeric column.

Syntax :- SELECT SUM (column_name).

RESULT :-

Hence the above SQL Pubnit Queries are implemented successfully.

ER MODEL FOR THE APPLICATION TO BE CONSTRUCTED TO A DATABASE

AIM:- To construct a ER Model for the Application to be constructed to a Database.

STEPS FOR DRAWING ER DIAGRAM:

1. First, identify the entities in your database. In this case, we have three entities
2. The second step involves identifying the relationship between the selected entities
3. The third step involves identifying cardinalities
4. Entity:- Entities are represented by rectangle. All table of database are treating as entity.
5. Attributes: Attributes are represented by ellipses. Attributes are properties of entities.

RESULT :

Hence we have successfully drawn the E-R Diagram.

EXPERIMENT-6 NESTED QUERIES

AIM:
To implement nested queries commands on sample exercise.

NESTED Query (Sub Query):
Sub query can have more than one level of nesting in one single Query.

Syntax for nested Query:-

SQL > select column-name 1" from "Table-name 1" where "Column-name 2" [comparison operator]
(select "column-name 2" from "Table-name 2" where condition);

→ SQL > select <column-nam> from <table-1> where <column-name> <relational-operation> 'Values' (select (aggregate function) from <table-1> where <column name> = "value" (select <column-name> from <table-2> where <column-name> = 'value'));

EXPERIMENT - I. JOIN OPERATORS

AIM:- To know the use of join operator in SQL.

JOIN:-

A join is used to combine rows from multiple A join is performed whenever two or more tables is listed in the form clause of an SQL commands.

TABLE STRUCTURE:

Syntax: SQL > desc supplies;
SQL > desc order;
SQL > select * from suppliers;

1) **Natural joins:**

Syntax: SQL > select * from supplier, ord where supplier.supid = ord.supid;

2) **Outer join:**

→ **left outer join**

Syntax:- select supplier.supid, supplier.SUPNAME, ord.odate where supplier.supid (+) = ord.supid

→ **right outer join**

Syntax:- SQL > supplier.supid, supplier.supname, ord.odate from supplier.ord where supplier.supid = ord.supid (+) supid (+) = ord.supid (+)

EXPERIMENT - 8 SET OPERATORS AND VIEWS

AIM:

To implement set operators and views using SQL

→ **set operators:**

They are special type of operators which are used to combine the result of 2 queries operator covered under SET operators are:

1) **union:** It will be used to combine the result of two select statements. Duplicate rows will be eliminated from the results obtained after performing the union operation.

Syntax:

select * from (table_name1) union select * from (table_nam

2) **union All:**

Syntax: select * from (table_name1) union All select * from (table_name2)

3) **Intersect:**

Syntax: select * from (table_name1) intersect select * from (table name2);

4) **Minus (Except):**

Syntax: Select * from (table_name1) minus select * from (table_name2);

→ **views:**

1) **Create view:**

Syntax: Create view (view_name) as select (column1, column2....) from (table name) where (condition);

2) **Insert:**

Syntax: Insert into (table_name) values (value1, value2, value3, value n);

4) update:

Syntax : update (tablename) set (column1 = value1, column2 = value2, ----) where condition;

5) DROP:

Syntax : drop view (view_name);