20EUCS010 – ANGELIN VARGHESE

Ex.No:1 CONCEPTUAL DATABASE DESIGN USING E-R DIAGRAM

Date: 28-02-2022

AIM:

To create an conceptual database design using E-R diagram for an application

DESCRIPTION:

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The main elements of an ERD are:

- Entities
- Relationships
- Attributes

ENTITY:An entity is a real-world item or concept that exists on its own. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

RELATIONSHIP:A relationship is the association that describes the interaction between entities.

TYPES:

1. Unary Relationship-Relationship between single entity
2. Binary Relationship-Relationship between two entity
3. Ternary Relationship-Relationship between three enntity

ATTRIBUTE:An attribute of an entity is a particular property that describes the entity.

TYPES:

1. **Simple-** An attribute cannot be further subdivided(Eg:Reg no)
2. **Composite**-An attribute can be further subdivided(Eg:Name---Firstname,Middlename,Lastname)
3. **Single valued**-An attribute has only one value(Eg:Reg No)
4. **Multi valued**-An attribute has more than one value(Eg:Phone Number)
5. **Stored -** An attribute's value cannot be determined from the values of other attributes(Eg:Date of Birth)
6. **Derived-** An attribute's value can be determined from the values of other attributes(Eg:Age which is derived from Date of Birth)
7. **Descriptive-** Attributes of the relationship is called descriptive attribute.(Eg: employee works for department. Here 'works for' is the relation between employee and department entities. The relation 'works for' can haveattribute DATE_OF_JOIN)

CONSTRAINTS:Contents of a database system must conform"

TYPES:

1. Mapping Cardinalities
2. Participation Cardinalities

MAPPING CARDINALITIES/CARDINALITY RATIO-Maximum no of relationship in which an entity can participate.

TYPES:

1. **One to One**- An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A
2. **One to Many**- An entity in A is associated with any number of entities in B.An entity in B,however can be associateed with atmost one entity in A.
3. **Many to One-** An entity in A is associated with at most one entity in B. An entity in B,however can be associated with any number of entities in A.
4. **Many to Many**- An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A

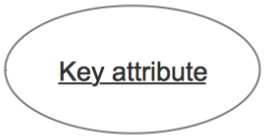
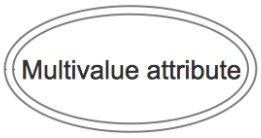
PARTICIPATION CONSTRAINTS/ EXISTENCE –Minimum no of relationship in which an entity can participate.

TYPES:

1. **Total Participation**-Every entity in E participates in at least one relationship in R
2. **Partial Participation**-Only some entities in E participate in relationships in R.

Entity Relationship Diagram Symbols — Chen notation

Symbol	Shape Name	Symbol Description
Entities		
	Entity	An entity is represented by a rectangle which contains the entity's name.
	Weak Entity	An entity that cannot be uniquely identified by its attributes alone. The existence of a weak entity is dependent upon another entity called the owner entity. The weak entity's identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.

	Associative Entity	An entity used in a many-to-many relationship (represents an extra table). All relationships for the associative entity should be many
Attributes		
	Attribute	In the Chen notation, each attribute is represented by an oval containing atributte's name
	Key attribute	An attribute that uniquely identifies a particular entity. The name of a key attribute is underscored.
	Multivalued attribute	An attribute that can have many values (there are many distinct values entered for it in the same column of the table). Multivalued attribute is depicted by a dual oval.
	Derived attribute	An attribute whose value is calculated (derived) from other attributes. The derived attribute may or may not be physically stored in the database. In the Chen notation, this attribute is represented by dashed oval.
Relationships		
	Strong relationship	A relationship where entity is existence-independent of other entities, and PK of Child doesn't contain PK component of Parent Entity. A strong relationship is represented by a single rhombus
	Weak (identifying) relationship	A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double rhombus.

STEPS TO DESIGN AN ER DIAGRAM:

Step 1: Identify the entities from the requirements gathered.

Various elements that are identified as elements are listed below under as Strong & Weak entities.

Strong Entities:

1. Student
2. Registration
3. Exam Result

Weak Entity:

1. Examination

Step 2: Finding the relationship between those entities.

Various relationships between the entities are characterized as Strong & Weak Relationships and are listed below.

Strong relationship:

1. **Performs registration:**
    a. Between student and registration
2. **Receives exam result:**
    a. Between student and exam result
3. **Attends examination:**
    a. Between student and exam

Weak Relationship:

1.Produces exam result:

a. Between examination and exam result

Step 3: Identify the key attributes for all entities.

Various Key Attributes for the entities are listed below

1. Student - **Reg No**
2. Registration - **Registration id**
3. Exam result - **Exam id**

Step 4: Identify the other relevant attributes.

Various Attributes for the entities are listed below

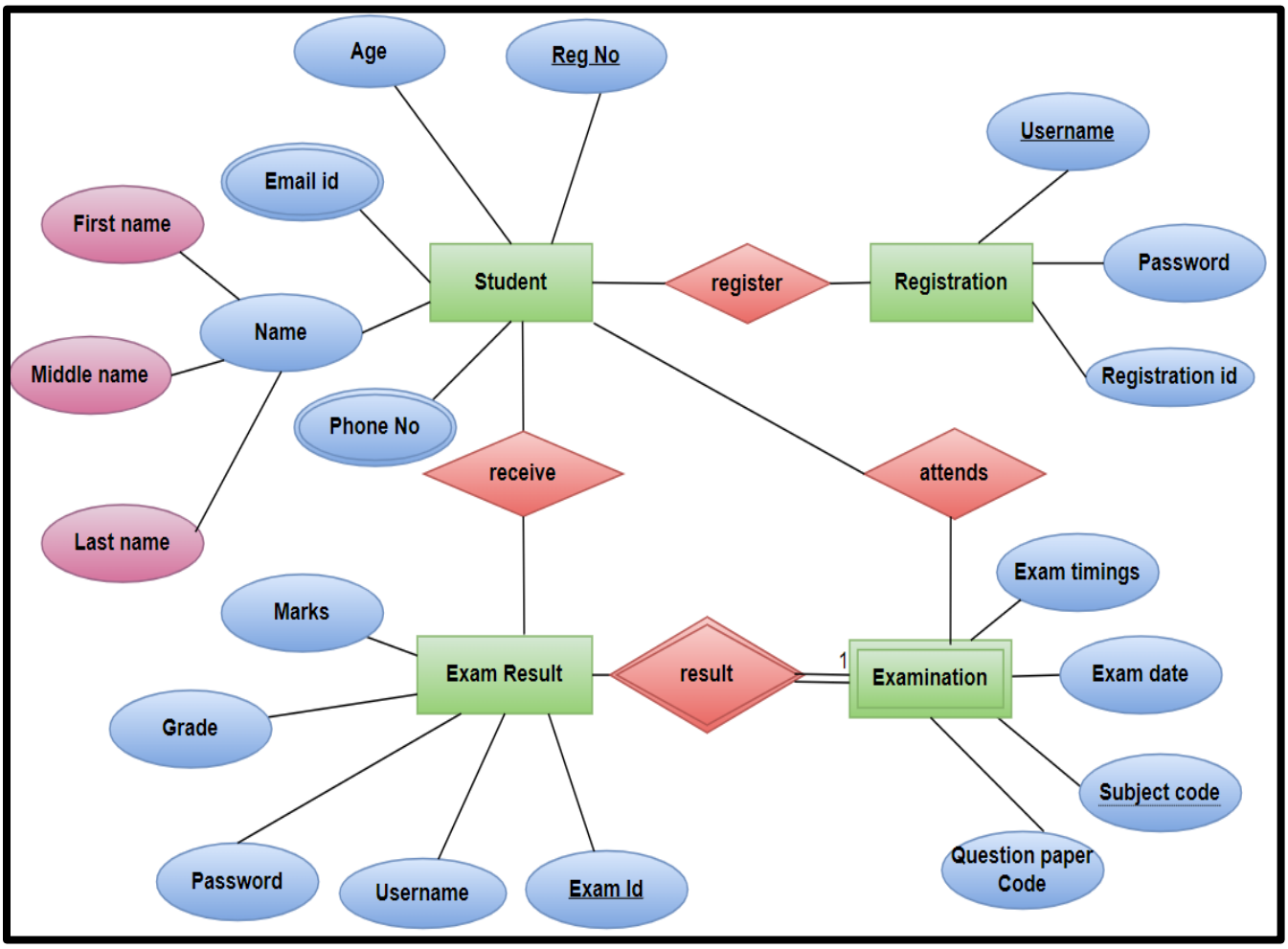
1. **Student:**
    a. Simple Attribute - Age, Reg No
    b. Multi-value Attribute - E-mail, Phone No
    c. Composite Attribute - Name

2. **Registration:**
    a. Simple Attribute - Username, Password, Registration id

3. **Examination:**
    a. Simple Attribute - Exam date, Question paper code, Subject name, Subject code
4. **Exam result:**
    a. Simple Attribute – Marks, Grade,Password, Username , Exam Id

Step 5: Draw an ER diagram.

EXAM MANAGEMENT SYSTEM:



Continuous Assessment

S.No.	Assessment Criteria	Max. Marks	Marks Obtained
1.	Objective & Description with sample data	20	
2.	Design	40	
3.	Execution and Testing	20	
4.	Viva Voce	10	
5.	Documentation	10	
	Total	100	
Faculty-In-Charge Signature			

Result:

Thus the implementation of E-R diagram for **Exam Management System** has been designed successfully.

EX.NO: 2a	IMPLEMENTATION OF SQL COMMANDS DDL, DML, DCL AND TCL
DATE:	

DATA DEFINITION LANGUAGE COMMANDS

OBJECTIVES

To implement the various Data Definition Language (DDL) commands using SQL.

LEARNING OUTCOMES

After the completion of this experiment, student will be able to

create a table and understand the table structure to perform any SQL operations alter a table to add columns or modify the table data types for storing various data. delete all contents of the table and drop the table and realize the importance of DDL.

PROBLEM STATEMENT

The queries to implement Data Definition Language (DDL) commands are

1. Create
2. Desc
3. Alter (add, modify)
4. Truncate
5. Drop

System and Software tools required

Software Required: SQL Plus

Operating System : WINDOWS 2000 / XP / 7

Computers Required : Minimum Requirement: Pentium III or Pentium IV RAM and 40 GB hard disk

DESCRIPTION

1. Create Table

This command is used to create a new table in RDBMS

SYNTAX:

Create table<table name> (column definition1, column definition2,…..);

2. Desc Table

This command is used to view the structure of the table.

Syntax

Desc<table name>;

3. Alter Table

This command is used to change the structure of a table. Alter command is used to:
a. Add a new column. b. Modify the existing column definition

Syntax

a. Alter table <table name> modify (column definition..);

b. Alter table<table name> add (column definition..);

4. Truncate Table

This command is used to delete all records from the table but the structure of the table will be retained.

Syntax

Truncate table <table name>;

5. Drop Table

This command is used to delete the table structure.

Syntax

Drop table <table name>;

SAMPLE CODING AND EXECUTION OF THE PROGRAM

1. SQL> create table employee7(employee_id varchar(20),first_name char(20),adress varchar(20),salary varchar(20));
2. SQL> desc employee7;
3. SQL> alter table employee7 add (dept_name varchar(20));

4. SQL> alter table employee7 modify (employee_id varchar(20));
5. SQL> truncate table employee7;
6. SQL> drop table employee7;

SAMPLE OUTPUT

1. Table created.
2. Name Null? Type

   EMPLOYEE_ID VARCHAR2(20)

   FIRST_NAME CHAR(20)

   ADDRESS

   VARCHAR2(20) SALARY

   VARCHAR2(20)
3. Table altered.
4. Table altered.
5. Table truncated.
6. Table dropped.

SCREEN OUTPUT:

DDL COMMANDS:

1. **CREATE:**

```
mysql> create database 20eucs010;
Query OK, 1 row affected (0.50 sec)
```

```
mysql> use 20eucs010;
Database changed
```

```
mysql> CREATE Table student_details1(
    -> roll_no int(3),
    -> first_name char(20),
    -> last__name char(20),
    -> phone_number int(10),
    -> );
```

```
mysql> CREATE Table waste_table(
    -> col1 int(1),
    -> col2 varchar(2),
    -> col3 bigint(3)
    -> );
Query OK, 0 rows affected (0.33 sec)
```

2. **DESC:**

```
mysql> desc student_details1;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| roll_no      | int(3)   | YES  |     | NULL    |       |
| first_name   | char(20) | YES  |     | NULL    |       |
| last__name   | char(20) | YES  |     | NULL    |       |
| phone_number | int(10)  | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

3. **ALTER:**

i) ADD:

```
mysql> alter table student_details1  add (dept_name varchar(20));
Query OK, 0 rows affected (0.59 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_details1;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| roll_no      | int(3)      | YES  |     | NULL    |       |
| first_name   | char(20)    | YES  |     | NULL    |       |
| last__name   | char(20)    | YES  |     | NULL    |       |
| phone_number | int(10)     | YES  |     | NULL    |       |
| dept_name    | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

ii) MODIFY:

```
mysql> ALTER Table student_details MODIFY phone_number bigint(10);
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student_details;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| roll_no      | int(3)      | YES  |     | NULL    |       |
| first_name   | char(20)    | YES  |     | NULL    |       |
| last__name   | char(20)    | YES  |     | NULL    |       |
| phone_number | bigint(10)  | YES  |     | NULL    |       |
| dept_name    | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4. TRUNCATE:

```
mysql> TRUNCATE Table waste_table;
Query OK, 0 rows affected (0.17 sec)

mysql> desc waste_table;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| col1  | int(1)     | YES  |     | NULL    |       |
| col2  | varchar(2) | YES  |     | NULL    |       |
| col3  | bigint(3)  | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

5. DROP:

```
mysql> DROP Table waste_table;
Query OK, 0 rows affected (0.48 sec)
```

```
mysql> desc waste_table;
ERROR 1146 (42S02): Table '20eucs010.waste_table' doesn't exist
```

TEST CASES:

1. Alter the table by adding more than two columns
2. Change the datatype from number to character for salary
3. Describe the table after each query is executed

CONCLUSION:

Thus the Data Definition Language (DDL) commands are executed successfully.

EX No: 2b
DATE:

DATA MANIPULATION LANGUAGE COMMANDS

OBJECTIVES

To implement the various Data Manipulation Language (DML) commands using SQL.

LEARNING OUTCOMES

After the completion of this experiment, student will be able to
query and manipulate the existing data base objects in table structure.
Insert into a table to add rows or modify the table to update values for storing data.
delete the unnecessary contents of the table, rename it and display the reformed table.

PROBLEM STATEMENT

The queries to implement the Data Manipulation Language (DML) commands are
1. insert
2. select
3. update
4. delete
5. rename

SYSTEM AND SOFTWARE TOOLS REQUIRED

Software Required: SQL Plus
Operating System : WINDOWS 2000 / XP / 7
Computers Required : Minimum Requirement: Pentium III or Pentium IV with 256

RAM and 40 GB hard disk

DESCRIPTION

1. Insert Command

This is used to add one or more rows to a table.

a. Inserting Values Into Table for All Fields

Here the number of values must correspond to number of column in table.

Syntax

Insert into <table name> values (value1, value2, value3…);

b. Inserting Values From User

Here the values of the columns in the table are obtained from user during run-time.

Syntax

Insert into <table name>values (<&column1&>, <&column2>…);

2. Select Command

This is used to retrieve columns from the table.

a. Selecting all rows from the table

Syntax

Select * from <table name>;

b. Selecting specific columns from the table

Syntax

Select column_name1, …..,column_nameN from <table name>;

c. Selecting rows using where clause from the table

Syntax

Select column_name1, …..,column_nameN from <table name>;

3. Update Command

This is used to modify one or a set of rows at a time.

Syntax

Update <table name>set <col1>= value1, <col2> = value2…where <condition>

4. Delete Command

This is used to remove the row from the table..

Syntax

Delete from <table name> where <condition>;

SAMPLE CODING AND EXECUTION OF THE PROGRAM

1. a) SQL> insert into employee7 values('14pc07','saranya','tnagar','5000');

1. b) SQL> insert into employee7 values('&employee_id','&first_name','&adress','&salary');

2. a) SQL> select *from employee7;

2. b) SQL> select employee_id, salary from employee7;

2. c) SQL> select employee_id, salary from employee7 where salary=5000;

3) SQL> update employee7 set adress='kknagar' where first_name='saranya';

4) SQL> delete from employee7 where first_name='saranya';

SAMPLE OUTPUT

1. a) 1 row created.

1. b) Enter value for employee_id: 14pc07

Enter value for first_name: saranya

Enter value for adress: tnagar

Enter value for salary: 5000

old 1: insert into employee7 values('&employee_id','&first_name','&adress'

new 1: insert into employee7 values('14pc07','saranya','tnagar','5000')

1 row created.

SQL> /

Enter value for employee_id: 14pc02

Enter value for first_name: chitra

Enter value for adress: amman nagar

Enter value for salary: 6000

old 1: insert into employee7 values('&employee_id','&first_name','&adress'

new 1: insert into employee7 values('14pc02','chitra','amman nagar','6000'

1 row created.

2. a) EMPLOYEE_ID FIRST_NAME ADRESS SALARY

---------- ---------- ---------- ----------

14pc07	saranya	tnagar	5000
14pc02	chitra	amman nagar	6000
14pc07	saranya	tnagar	5000
14pc02	chitra	amman nagar	6000

2. b)

EMPLOYEE_ID	SALARY
14pc07	5000
14pc02	6000
14pc07	5000
14pc02	6000

2. c)

EMPLOYEE_ID	SALARY
14pc07	5000
14pc07	5000

3) 2 rows updated.

4) 2 rows deleted.

SCREEN OUTPUT:

DML COMMANDS:

1. **INSERT:**

```
mysql> insert into student_details1 values(10,"Angelin","Varghese",9010074227,"CSE");
Query OK, 1 row affected (0.00 sec)

mysql> insert into student_details1 values(27,"Santina","Maria",9010066787,"ECE");
Query OK, 1 row affected (0.01 sec)
```

2. **SELECT:**

```
mysql> select * from student_details1;
+---------+------------+-----------+--------------+-----------+
| roll_no | first_name | last_name | phone_number | dept_name |
+---------+------------+-----------+--------------+-----------+
|      10 | Angelin    | Varghese  |   9010074227 | CSE       |
|      27 | Santina    | Maria     |   9010066787 | ECE       |
+---------+------------+-----------+--------------+-----------+
2 rows in set (0.00 sec)
```

```
mysql> select roll_no,first_name from student_details1;
+---------+------------+
| roll_no | first_name |
+---------+------------+
|      10 | Angelin    |
|      27 | Santina    |
+---------+------------+
2 rows in set (0.00 sec)
```

3. **UPDATE:**

```
mysql> update student_details1 set phone_number=9877456677 where first_name="Angelin";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update student_details1 set phone_number=7010074667 where last_name="Maria";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student_details1;
+---------+------------+-----------+--------------+-----------+
| roll_no | first_name | last_name | phone_number | dept_name |
+---------+------------+-----------+--------------+-----------+
|      10 | Angelin    | Varghese  |   9877456677 | CSE       |
|      27 | Santina    | Maria     |   7010074667 | ECE       |
+---------+------------+-----------+--------------+-----------+
2 rows in set (0.00 sec)
```

4. **DELETE:**

```
mysql> INSERT into waste_table values('2','Oi','88');
Query OK, 1 row affected (0.13 sec)

mysql> DELETE From waste_table where col2='Oi';
Query OK, 1 row affected (0.13 sec)

mysql> SELECT *From waste_table;
+------+------+------+
| col1 | col2 | col3 |
+------+------+------+
|    1 | Hi   |   95 |
+------+------+------+
1 row in set (0.01 sec)
```

```
mysql> alter table student_details1 drop column last_name;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from student_details1;
+---------+------------+--------------+-----------+
| roll_no | first_name | phone_number | dept_name |
+---------+------------+--------------+-----------+
|      10 | Angelin    |   9877456677 | CSE       |
|      27 | Santina    |   7010074667 | ECE       |
+---------+------------+--------------+-----------+
2 rows in set (0.00 sec)
```

5. **RENAME:**

```
mysql> ALTER Table student_details1 RENAME student_details;
Query OK, 0 rows affected (0.28 sec)
```

TEST CASES

1. Select the table after each DML query is executed
2. Select multiple columns to be displayed from a table
3. Update all rows with same salary for same name

CONCLUSION

Thus the Data Manipulation Language (DML) commands are executed successfully.

EX No: 2c
DATE:

DATA AND TRANSACTION CONTROL LANGUAGE COMMANDS

OBJECTIVES

To implement the various Data Control Language (DCL) and Transaction Control Language (TCL) commands using SQL.

LEARNING OUTCOMES

After the completion of this experiment, student will be able to

Perform transactions with the existing data base objects in table.

Undo, redo and save the operations done with the existing rows in the table.

Understand the access control of database, user privileges and securing the database.

PROBLEM STATEMENT

The queries to implement Data Control Language (DCL) and Transaction Control Language (TCL) commands are

1. grant
2. revoke
3. commit
4. save point
5. roll back

SYSTEM AND SOFTWARE TOOLS REQUIRED

Software Required: SQL Plus
Operating System : WINDOWS 2000 / XP / 7
Computers Required : Minimum Requirement: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk

DESCRIPTION

DCL Commands

1. Grant Command

This statement will grant permissions for any particular operation.

Syntax

Grant < database_priv [database_priv…..] > to <user_name> identified by <password>

[,<password…..];

2. Revoke Privilege Command

Using this statement the privilege that you specify to user are revoked cascade required to remove any referential integrity constraints.

Syntax

Revoke <database_priv> from <user / public>;

TCL COMMANDS

1. Commit Command

It completes the transactions done by making changes permanently to the database.

Syntax

Commit;

2. Save point Command

Save points are temporary markers that are used to divide a lengthy transaction and save the changes made as each point of reference to the table.

Syntax

Save point<save point id>;

3. Rollback Command

It reverts back the changes made and completes the transaction in the table.

SYNTAX:
Rollback; (or) rollback to <save point id>;

SAMPLE CODING AND EXECUTION OF THE PROGRAM

1. SQL> Grant select, update , insert on employees to departments with grant option;

2. SQL> Revoke select, update , insert on employees from departments;

3. SQL> commit;

4. SQL> savepoint ep;

5. SQL> rollback to savepoint ep1;

SAMPLE OUTPUT

1. user or role 'CHITRA' does not exist

2. user or role 'CHITRA' does not exist

3. Commit complete.

4. Save point created.

5. Rollback complete.

SCREEN OUTPUT:

```
mysql> create user newuser@localhost identified by '1234';
Query OK, 0 rows affected (0.01 sec)

mysql> select user from mysql.user;
+---------------+
| user          |
+---------------+
| mysql.session |
| mysql.sys     |
| newuser       |
| root          |
| test          |
+---------------+
5 rows in set (0.00 sec)

mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

1.GRANT:

```
mysql> show grants for newuser@localhost;
+-----------------------------------------------+
| Grants for newuser@localhost                  |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO 'newuser'@'localhost' |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> grant select on dbms.users to newuser@localhost;
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> show grants for newuser@localhost;
+-----------------------------------------------------------+
| Grants for newuser@localhost                              |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO `newuser`@`localhost`               |
```

```
| GRANT SELECT ON `dbms`.`users` TO `newuser`@`localhost` |
+-----------------------------------------------------------+
3 rows in set (0.00 sec)
```

Newuser:

```
C:\Program Files (x86)\MySQL\MySQL Server 5.7\bin>mysql -u newuser -p
Enter password:
```

```
mysql> use dbms;
Database changed
```

```
mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    2 | anwi |
|    0 | anu  |
+------+------+
3 rows in set (0.00 sec)
```

2. REVOKE:

```
mysql> revoke select on dbms.users from newuser@localhost;
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> show grants for newuser@localhost;
+-----------------------------------------------+
| Grants for newuser@localhost                  |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `newuser`@`localhost` |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

3. COMMIT

```
mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
+------+------+
1 row in set (0.03 sec)
```

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> set autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into users values(1,'anu');
Query OK, 1 row affected (0.00 sec)

mysql> insert into users values(2,'anwi');
Query OK, 1 row affected (0.00 sec)

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    1 | anu  |
|    2 | anwi |
+------+------+
3 rows in set (0.00 sec)
```

4.ROLLBACK

```
mysql> rollback;
Query OK, 0 rows affected (0.04 sec)

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
+------+------+
1 row in set (0.00 sec)
```

5. SAVEPOINT AND ROLLBACK:

```
mysql> set autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into users values(2,'anwi');
Query OK, 1 row affected (0.00 sec)

mysql> insert into users values(1,'anu');
Query OK, 1 row affected (0.00 sec)

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    2 | anwi |
|    1 | anu  |
+------+------+
3 rows in set (0.00 sec)
```

```
mysql> savepoint insertion;
Query OK, 0 rows affected (0.01 sec)

mysql> update users set id=0 where name='anu';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    2 | anwi |
|    0 | anu  |
+------+------+
3 rows in set (0.00 sec)
```

```
mysql> rollback to insertion;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    2 | anwi |
|    1 | anu  |
+------+------+
3 rows in set (0.00 sec)
```

```
mysql> update users set id=1 where name='anwi';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    1 | anwi |
|    0 | anu  |
+------+------+
3 rows in set (0.00 sec)

mysql> ROLLBACK TO UPDATION;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from users;
+------+------+
| id   | name |
+------+------+
|    3 | ABI  |
|    2 | anwi |
|    0 | anu  |
+------+------+
3 rows in set (0.00 sec)
```

CONCLUSION

Thus the Data Control Language (DCL) and Transaction Control Language (TCL) commands are executed successfully.

EX.NO: 3	QUERIES TO DEMONSTRATE IMPLEMENTATION OF INTEGRITY CONSTRAINTS
DATE:	

Aim

To implement various Integrity Constraints in SQL queries.

Description:

INTEGRITY CONSTRAINT

An integrity constraint is a mechanism used by oracle to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table.

The types of the integrity constraints are:

a) Domain Integrity

b) Entity Integrity

c) Referential Integrity

a) Domain Integrity:

This constraint sets a range and any violations that take place will prevent the user from performing the manipulation that caused the breach.

It includes:

Not Null constraint:

- While creating tables, by default the rows can have null value .the enforcement of not null constraint in a table ensure that the table contains values. Principle of null values:
- Setting null value is appropriate when the actual value is unknown, or when a value would not be meaningful.
- A null value is not equivalent to a value of zero. A null value will always evaluate to null in any expression.
- When a column name is defined as not null, that column becomes a mandatory i.e., the user has to enter data into it.
- Not null Integrity constraint cannot be defined using the alter table command when the table contain rows.

Syntax

- Column name data type[size] <not null>;

Example:

- Create table cust(custid number(6) not null, name char(10));
- Alter table cust modify (name not null);

Check Constraint:

- Check constraint can be defined to allow only a particular range of values .when the manipulation violates this constraint, the record will be rejected. Check condition cannot contain sub queries.

Syntax

Column name data type[size] <check> <logical expression>.

Example:

- Create table student (regno number (6), mark number (3) constraint b check (mark >=0 and mark <=100));
- Alter table student add constraint b2 check (length(regno<=4));

b) Entity Integrity:

Entity Integrity maintains uniqueness in a record.

An entity represents a table and each row of a table represents an instance of that entity. To identify each row in a table uniquely we need to use this constraint.

There are 2 entity constraints:

Unique key constraint:

It is used to ensure that information in the column for each record is unique, as with telephone or drivers license numbers. It prevents the duplication of value with rows of a specified column in a set of column. A column defined with the constraint can allow null value. If unique key constraint is defined in more than one column i.e., combination of column cannot be specified. Maximum combination of columns that a composite unique key can contain is 16.

Syntax

Column name data type[size] <unique>;

Example: Create table cust(custid number(6) constraint uni unique, name char(10));
Alter table cust add(constraint c unique(custid));

Primary Key Constraint:

A primary key avoids duplication of rows and does not allow null values. It can be defined on one or more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null. A table should have only one primary key. If a primary key constraint is assigned to more than one column or combination of column is said to be composite primary key, which can contain 16 columns.

Syntax

Column name data type [size] <primary key>;

Example: Create table stud(regno number(6) constraint primary key, name char(20));

c) Referential Integrity:

It enforces relationship between tables.

To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint.

To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry.

Foreign key: A column or combination of column included in the definition of referential integrity, which would refer to a referenced key.
Referenced key: It is a unique or primary key upon which is defined on a column belonging to the parent table.

Syntax

Column name data type[size] <references> TABLE1 NAME(column name of table 1);

Example:

PRIMARY KEY TABLE:

CREATE TABLE product(product_id number(5) CONSTRAINT pd_id_pk PRIMARY KEY, product_name char(20),supplier_name char(20),unit_price number(10));

FOREIGN KEY TABLE:

CREATE TABLE order_items
(order_id number(5) CONSTRAINT od_id_pk PRIMARY KEY,
product_id number(5) CONSTRAINT pd_id_fk REFERENCES,
product(product_id),
product_name char(20),
supplier_name char(20),
unit_price number(10));

Or

CREATE TABLE order_items
(order_id number(5) ,
product_id number(5),
product_name char(20),
supplier_name char(20),
unit_price number(10)
CONSTRAINT od_id_pk PRIMARY KEY(order_id),
CONSTRAINT pd_id_fk FOREIGN KEY(product_id) REFERENCES product(product_id));

TEST CASES:

1. For storing the supplier details, the Supplier table needs to be created. Table is created with the given constraints. Describe the table after creating the table

Column_name	Data Type	Constraints	Description
supplierid	VARCHAR2(6)	PRIMARY KEY	ID of the supplier
suppliername	Varchar2(30)	UNIQUE	Name of the supplier

suppliercontactno	Varchar2(15)		Contact details of Supplier
supplieremailid	Varchar2(30)		Mailid of the supplier

2. For describing about stock of items in warehouse, Item table needs to be created. Table is created with the given constraints. Describe the table after creating the table

Column_name	Data Type	Constraints	Description
itemcode	VARCHAR2(6)	PRIMARY KEY	Unique Code of an item
itemtype	Varchar2(30)		Type of an item
description	Varchar2(100)	NOT NULL	Description about an item
price	NUMBER(7,2)		Price of an item
reorderlevel	NUMBER		Minimum amount of an item which a company holds in stock,such that, when stock falls to this amount, the item must be reordered
quantityonhand	NUMBER		Stock availability for an item
category	CHAR(1)		Category of an item

3. To describe about details of quotations provided by supplier for items, Quotation table needs to be created. Table is created with the given constraints

Column_name	Data Type	Constraints	Description
quotationid	VARCHAR2(6)	PRIMARY KEY	Unique ID
supplierid	Varchar2(6)	References Supplier	ID of the supplier
itemcode	VARCHAR2(10)	References Item code	Unique Code of an item
Quotedprice	NUMBER		Estimate of price
quotationdate	DATE		Quotation expiration date
quotationstatus	VARCHAR2(10)	CHECK	Status of quotation. It can take the values as Open or Accepted or Rejected or Closed

4. SQL statement to insert Minimum 5 values inside Item, Supplier & Quotation table.
5. SQL statement to retrieve all the records from Item, Supplier & Quotation table.

SCREEN OUTPUT:

Table values insertion:

Table items:

```
mysql> insert into items values
    -> ('20cs1','biscuit','good',11,13,12,'a');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into items values ('cos22','cosmetics','good',14,11,12,'b');
Query OK, 1 row affected (0.00 sec)

mysql> insert into items values ('cos22','cosmetics','good',9,11,12,'b');
ERROR 3819 (HY000): Check constraint 'items_chk_1' is violated.
mysql> insert into items values ('fr22','fruits','good',19,13,14,'c');
Query OK, 1 row affected (0.00 sec)

mysql> insert into items values ('b022','books','good',35,15,18,'d');
Query OK, 1 row affected (0.00 sec)

mysql> insert into items values ('pe22','pencils','best',45,12,20,'c');
Query OK, 1 row affected (0.00 sec)
```

Table supplier:

```
mysql> insert into supplier values
    -> ('21','raj','8072996911','raj@gmail.com');
Query OK, 1 row affected (0.12 sec)

mysql> insert into supplier values ('22','ram','9072996911','ram@gmail.com');
Query OK, 1 row affected (0.10 sec)

mysql> insert into supplier values ('23','rajan','907298971','rajan@gmail.com');

Query OK, 1 row affected (0.09 sec)

mysql> insert into supplier values ('24','moorthy','907298872','moorthy@gmail.com');
Query OK, 1 row affected (0.08 sec)

mysql> insert into supplier values ('25','peter','907297792','peter@gmail.com');

Query OK, 1 row affected (0.08 sec)
```

Table quotation:

```
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> insert into quotation values ('11','21','b1','201','20.02.2021','closed'
);
Query OK, 1 row affected, 1 warning (0.09 sec)

mysql> insert into quotation values ('11','21','b1','201','20.02.2021','anu');
ERROR 3819 (HY000): Check constraint 'quotation_chk_1' is violated.
mysql> insert into quotation values ('12','22','f1','202','20.03.2021','rejected');
Query OK, 1 row affected, 1 warning (0.10 sec)

mysql> insert into quotation values ('13','23','y1','203','20.04.2021','rejected');
Query OK, 1 row affected, 1 warning (0.10 sec)

mysql> insert into quotation values ('14','24','x1','204','20.04.2022','open');
Query OK, 1 row affected, 1 warning (0.11 sec)

mysql> insert into quotation values ('15','25','z1','205','25.04.2022','open');

Query OK, 1 row affected, 1 warning (0.11 sec)
```

Describing Tables:

Table items:

OBJ

```
mysql> select * from items;
+----------+-----------+-------------+-------+---------------+--------------+----------+
| itemcode | itemtype  | description | price | recorderlevel | quantityhand | category |
+----------+-----------+-------------+-------+---------------+--------------+----------+
| 20cs1    | biscuit   | good        |    11 |            13 |           12 | a        |
| 20cs11   | dress     | good        |    20 |            11 |           12 | b        |
| b022     | books     | good        |    35 |            15 |           18 | d        |
| cos22    | cosmetics | good        |    14 |            11 |           12 | b        |
| fr22     | fruits    | good        |    19 |            13 |           14 | c        |
| pe22     | pencils   | best        |    45 |            12 |           20 | c        |
+----------+-----------+-------------+-------+---------------+--------------+----------+
6 rows in set (0.00 sec)
```

Table supplier:

```
mysql> select * from supplier;
+--------------+----------------+-------------------+--------------------+
| supplier_id | supplier_name | supplier_contact | supplier_emailid  |
+--------------+----------------+-------------------+--------------------+
| 21           | raj            | 8072996911        | raj@gmail.com      |
| 22           | ram            | 9072996911        | ram@gmail.com      |
| 23           | rajan          | 907298971         | rajan@gmail.com    |
| 24           | moorthy        | 907298872         | moorthy@gmail.com  |
| 25           | peter          | 907297792         | peter@gmail.com    |
+--------------+----------------+-------------------+--------------------+
5 rows in set (0.00 sec)
```

Table quotation:

```
mysql> select* from quotation;
+--------------+-------------+-----------+--------------+----------------+------------------+
| quotation_id | supplier_id | item_code | quoted_price | quotation_date | quotation_status |
+--------------+-------------+-----------+--------------+----------------+------------------+
| 11           | 21          | b1        |          201 | 2020-02-20     | closed           |
| 12           | 22          | f1        |          202 | 2020-03-20     | rejected         |
| 13           | 23          | y1        |          203 | 2020-04-20     | rejected         |
| 14           | 24          | x1        |          204 | 2020-04-20     | open             |
| 15           | 25          | z1        |          205 | 2025-04-20     | open             |
+--------------+-------------+-----------+--------------+----------------+------------------+
5 rows in set (0.00 sec)
```

```
mysql> use Anu
Database changed
mysql> create table supplier(supplier_id varchar(6) primary key,supplier_name varchar(30) unique,supplier_contact varchar(15),supplier_emailid varcha
(30));
Query OK, 0 rows affected (0.31 sec)

mysql> desc supplier;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| supplier_id     | varchar(6)  | NO   | PRI | NULL    |       |
| supplier_name   | varchar(30) | YES  | UNI | NULL    |       |
| supplier_contact | varchar(15) | YES  |     | NULL    |       |
| supplier_emailid | varchar(30) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

```
mysql> alter table supplier
    -> add supplier_phone bigint(15) NOT NULL;
Query OK, 0 rows affected (0.92 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> insert into supplier
    -> values('aa','anu','anukiru','nukirutyh',8072996911);
Query OK, 1 row affected (0.05 sec)

mysql> select * from supplier;
+-------------+---------------+------------------+------------------+----------------+
| supplier_id | supplier_name | supplier_contact | supplier_emailid | supplier_phone |
+-------------+---------------+------------------+------------------+----------------+
| aa          | anu           | anukiru          | nukirutyh        |     8072996911 |
+-------------+---------------+------------------+------------------+----------------+
1 row in set (0.00 sec)

mysql> desc supplier;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| supplier_id      | varchar(6)  | NO   | PRI | NULL    |       |
| supplier_name    | varchar(30) | YES  | UNI | NULL    |       |
| supplier_contact | varchar(15) | YES  |     | NULL    |       |
| supplier_emailid | varchar(30) | YES  |     | NULL    |       |
| supplier_phone   | bigint(15)  | NO   |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

NOT NULL:

```
mysql> insert into supplier
    -> values(null,'adsa','adfadfd','adafcdsf',7890887889);
ERROR 1048 (23000): Column 'supplier_id' cannot be null
mysql>
```

PRIMARY KEY:

```
mysql> insert into supplier
    -> values('aa','anu','hgjh','adafdf',8707878);
ERROR 1062 (23000): Duplicate entry 'aa' for key 'PRIMARY'
mysql>
```

UNIQUE KEY:

```
mysql> insert into supplier
    -> values('bb','anu','adsafa','adasfdaf',679689608);
ERROR 1062 (23000): Duplicate entry 'anu' for key 'supplier_name'
mysql>
```

```
mysql> alter table supplier modify column supplier_id varchar(6) NULL;
ERROR 1171 (42000): All parts of a PRIMARY KEY must be NOT NULL; if you need NULL in a key, use UNIQUE instead
mysql>
```

```
mysql> insert into supplier
    -> values('cc',null,'hojijpjp','hjjojopj',7960868);
Query OK, 1 row affected (0.03 sec)

mysql> select * from supplier;
+-------------+---------------+------------------+------------------+----------------+
| supplier_id | supplier_name | supplier_contact | supplier_emailid | supplier_phone |
+-------------+---------------+------------------+------------------+----------------+
|             | anukiugtr     | vuygbhjb         | acdavddc         |     8072996911 |
| aa          | anu           | anukiru          | nukirutyh        |     8072996911 |
| cc          | NULL          | hojijpjp         | hjjojopj         |        7960868 |
+-------------+---------------+------------------+------------------+----------------+
3 rows in set (0.00 sec)

mysql>
```

DEFAULT:

```
mysql> create table items(item_code int(10) primary key,item_type varchar(30),de
scription varchar(100) not null,price_no int(10) default 345);
Query OK, 0 rows affected (0.24 sec)
```

```
mysql> desc items;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| item_code   | int(10)      | NO   | PRI | NULL    |       |
| item_type   | varchar(30)  | YES  |     | NULL    |       |
| description | varchar(100) | NO   |     | NULL    |       |
| price_no    | int(10)      | YES  |     | 345     |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> insert into items
    -> values(132,'asdfdafdf','adqefrwgfbrwfb',445);
Query OK, 1 row affected (0.04 sec)

mysql> select * from items
    -> ;
+-----------+-----------+----------------+----------+
| item_code | item_type | description    | price_no |
+-----------+-----------+----------------+----------+
|       132 | asdfdafdf | adqefrwgfbrwfb |      445 |
+-----------+-----------+----------------+----------+
1 row in set (0.00 sec)
```

CHECK:

```
mysql> create table items(itemcode varchar(6) PRIMARY KEY,itemtype varchar(30),description varchar(100) NOT NULL,price int check(price > 10),rec
orderlevel int,quantityhand int,category char(1));
Query OK, 0 rows affected (0.54 sec)

mysql> desc items;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| itemcode      | varchar(6)   | NO   | PRI | NULL    |       |
| itemtype      | varchar(30)  | YES  |     | NULL    |       |
| description   | varchar(100) | NO   |     | NULL    |       |
| price         | int          | YES  |     | NULL    |       |
| recorderlevel | int          | YES  |     | NULL    |       |
| quantityhand  | int          | YES  |     | NULL    |       |
| category      | char(1)      | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> insert into items values
    -> ('20cs1','biscuit','good',11,13,12,'a');
Query OK, 1 row affected (0.00 sec)

mysql> select*from items;
+----------+----------+-------------+-------+--------------+--------------+----------+
| itemcode | itemtype | description | price | recorderlevel | quantityhand | category |
+----------+----------+-------------+-------+--------------+--------------+----------+
| 20cs1    | biscuit  | good        |    11 |            13 |           12 | a        |
+----------+----------+-------------+-------+--------------+--------------+----------+
1 row in set (0.00 sec)
```

```
mysql> insert into items values ('20cs11','dress','good',10,11,12,'b');
ERROR 3819 (HY000): Check constraint 'items_chk_1' is violated.
mysql>
```

FOREIGN KEY:

```
mysql> create table quotation(quotation_id varchar(6)primary key,supplier_id var
char(6),item_code varchar(10),quoted_price int,quotation_date date,quotation_sta
tus varchar(10)check(quotation_status in('open','accepted','rejected','closed'))
,foreign key(supplier_id)references supplier(supplier_id));
Query OK, 0 rows affected (1.64 sec)

mysql> alter table quotation add constraint const foreign key(supplier_id) refer
ences supplier(supplier_id);
Query OK, 0 rows affected (2.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc quotation;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| quotation_id     | varchar(6)  | NO   | PRI | NULL    |       |
| supplier_id      | varchar(6)  | YES  | MUL | NULL    |       |
| item_code        | varchar(10) | YES  |     | NULL    |       |
| quoted_price     | int         | YES  |     | NULL    |       |
| quotation_date   | date        | YES  |     | NULL    |       |
| quotation_status | varchar(10) | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE supplier  DROP COLUMN supplier_id;
ERROR 1829 (HY000): Cannot drop column 'supplier_id': needed in a foreign key constraint 'const' of table 'quotation'
mysql>
```

Result:

Thus the various integrity constraints queries are executed successfully in SQL.

Ex. No.

Continuous Assessment

S.No.	Assessment Criteria	Max. Marks	Marks Obtained
1.	Objective & Description with sample data	20	
2.	Design	40	
3.	Execution and Testing	20	
4.	Documentation	10	
5.	Viva	10	
6.	Total	100	
Faculty Signature			

20EUCS010 – ANGELIN VARGHESE

Ex.No: 4 **INBUILT FUNCTIONS**

Date: 13/04/2022

Aim:

To perform Inbuilt functions in SQL.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Perform mathematical calculations, character functions and other functions.
- Understand the need for pre defined functions in real time scenarios.

Problem Statement:

SQL has many built-in functions for performing calculations on data.

1. Mathematical functions
2. Character functions
3. Conversion Function
4. Group Function (Aggregate function)
5. Miscellaneous Function
6. Special Function

System and Software tools required:

Software Required: Oracle

Operating System : WINDOWS 2000 / XP / NT OR LINUX OR UBUNTU

Computers Required : Minimum Requirement: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk

Description:

1. Mathematical functions:

There are many mathematical functions. All mathematical functions return NULL in the event of an error. Some of the mathematical functions are:

a. ABS(X)-Returns the absolute value of X.

b. CEIL(X)-Returns the smallest integer value not less than X.

2. Character functions

A character function is a function that takes one or more character values as parameters and returns either a character value or a number value. Some of the character functions are:

a. RTRIM-Trims the right side of a string of all specified characters.

b. LPAD-Pads a string on the left with the specified characters.

3. Conversion Functions:

Conversion functions are used to must first convert the data to be the right datatype for the operation. Some of the conversion functions are:

a. ADD_MONTHS (date, n) - Returns a date value after adding 'n' months to the date 'x'.

b. LAST_DAY (x) -It is used to determine the number of days remaining in a month from the date 'x' specified.

C. NVL-The Oracle NVL function lets you substitute a value when a null value is encountered.

EG:NVL(string1, replace_with)

4. Group Function(Aggregate function):

a. MAX(): This function is used to get the maximum value from a column.
b. AVG(): This function is used to get the average value of a numeric column.

5. Miscelleneous Function :

a. UID function: It returns an integer that uniquely identifies the current user.

Sample Coding and Execution of the Program:

1. SQL> select abs(-34) from dual;
2. SQL> select ceil(50.8) from dual;
3. SQL> select rtrim('computer','r') from dual;
4. SQL> select lpad('oracle',9,'*') from dual;
5. SQL> select substr('indian',1,5) from dual;
6. SQL>select add_months('18-may-06') from dual;
7. SQL> select last_day('18-may-06') from dual;
8. SQL> select min(salary) from employee7;
9. SQL> select avg(salary) from employee7;
10. SQL> select uid from dual;

```
select rtrim('computer','r') from dual;
```

```
compute
```

1 rows returned in 0.01 seconds Download

```
select uid from dual;
```

```
73373
```

1 rows returned in 0.00 seconds Download

```
select avg(salary) from employee;
```

```
4442.857142857142857142857142857142857714
```

1 rows returned in 0.01 seconds Download

```
select min(salary) from employee;
```

```
2000
```

1 rows returned in 0.01 seconds Download

```
mysql> select abs(-34) from dual;
+----------+
| abs(-34) |
+----------+
|       34 |
+----------+
1 row in set (0.00 sec)

mysql> select ceil(50.8) from dual;
+------------+
| ceil(50.8) |
+------------+
|         51 |
+------------+
1 row in set (0.00 sec)

mysql> select lpad('oracle',9,'*') from dual;
+----------------------+
| lpad('oracle',9,'*') |
+----------------------+
| ***oracle            |
+----------------------+
1 row in set (0.00 sec)

mysql> select substr('indian',1,5) from dual;
+----------------------+
| substr('indian',1,5) |
+----------------------+
| india                |
+----------------------+
1 row in set (0.00 sec)

mysql> select last_day('18-may-06') from dual;
+-----------------------+
| last_day('18-may-06') |
+-----------------------+
| NULL                  |
+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

Sample Output:

1.ABS(-34)

34
2.CEIL(50.8)

51
3.LPAD('ORA

***oracle
4.RTRIM('

Compute
5.SUBST
----- India
6.LAST_DAY(

31-MAY-06
7.ADD_MONTH

18-OCT-06
8.AVG(SALARY)

6650
9.MIN(SALARY)

5400
10.UID

468

INBUILT FUNCTIONS

Test Cases:

1.For a discount of 25.5% being offered on all FMCG item's unit price, display item code existing unit price as "Old Price" and discounted price as "New Price". Round off the discounted price to two decimal values.

Select itemcode,price,round(price - price*0.255,2) as "New Price" from item where itemtype = 'FMCG'

```
1  Select itemcode,price,round(price - price*0.255,2) as "New Price" from item where itemtype = 'FMCG'
```

Results | Explain | Describe | Saved SQL | History

ITEMCODE	PRICE	New Price
I1001	20	14.9
I1002	60	44.7
I1003	120	89.4
I1004	15	11.18
I1005	30	22.35
I1020	25	18.63
I1021	23	17.14
I1022	21	15.65
I1023	17	12.67

9 rows returned in 0.08 seconds Download

2.Retrieve the employee id, employee name of billing staff and the retail outlet where they work. Perform a case insensitive search

Select empid, empname ,worksin from employee where lower(designation) = 'billing staff'

```
1  Select empid, empname ,worksin from employee where lower(designation) = 'billing staff'
```

Results | Explain | Describe | Saved SQL | History

EMPID	EMPNAME	WORKSIN
1003	Lisa	R1001
1007	Sam	R1002
1009	Henry	R1002
1010	Cris	R1001
1011	Donald	R1001
1012	Edwin	R1002

6 rows returned in 0.09 seconds Download

3.Retrieve the order id, order status and payment mode of all the orders. Display ‘Payment yet not done’ when payment mode has NULL value.

Select orderid, status, nvl(paymentmode,'Payment yet not done')from orderstatus

```
1  Select orderid, status, nvl(paymentmode,'Payment yet not done')from orderstatus
```

Results | Explain | Describe | Saved SQL | History

ORDERID	STATUS	NVL(PAYMENTMODE,'PAYMENTYETNOTDONE')
O1001	Delivered	Cash
O1002	Partial Delivery	Payment yet not done
O1003	Partial Delivery	Payment yet not done
O1004	Delivered	Cheque
O1005	Delivered	Cheque
O1006	Delivered	Cash
O1007	Partial Delivery	Payment yet not done
O1008	Ordered	Payment yet not done

8 rows returned in 0.08 seconds Download

4.Retrieve the description of items which have more than 15 characters

select description from item where length(description) > 15

```
1  select description from item where length(description) > 15
```

Results | Explain | Describe | Saved SQL | History

DESCRIPTION
Britannia Marie Gold Cookies
Intel C2D Processor
Intel Motherboard
Microsoft Keyboard
Britannia Choco Chip Cookies
Amfeast Choco Chip Cookies
Amfeast Marie Gold Cookies

7 rows returned in 0.08 seconds Download

5.Display numeric part of supplier id

Select substr(supplierid,2,5) from supplier

```
1  Select substr(supplierid,2,5) from supplier
```

Results | Explain | Describe | Saved SQL | History

SUBSTR(SUPPLIERID,2,5)
1001
1002
1003
1004
1005

5 rows returned in 0.08 seconds Download

6.Retrieve the order id and the number of days between order date and payment date for all orders

SELECT abs(to_date(orderdate,'mm/dd/yyyy')-to_date(paymentdate,'mm/dd/yyyy')) FRO M orderstatus

```
1  SELECT abs(to_date(orderdate,'mm/dd/yyyy')-to_date(paymentdate,'mm/dd/yyyy')) FROM orderstatus
```

Results | Explain | Describe | Saved SQL | History

ABS(TO_DATE(ORDERDATE,'MM/DD/YYYY')-TO_DATE(PAYMENTDATE,'MM/DD/YYYY'))
6
-
-
10
10
357
-
-

8 rows returned in 0.08 seconds Download

7. Retrieve the order id and the number of months between order date and payment date for all orders

SELECT ceil(MONTHS_BETWEEN(TO_DATE(orderdate,'MM-DD-YYYY'),TO_DATE(paymentdate,'MM-DD-YYYY'))) "Months" FROM orderstatus

```
1  SELECT ceil(MONTHS_BETWEEN(TO_DATE(orderdate,'MM-DD-YYYY'),TO_DATE(paymentdate,'MM-DD-YYYY') )) "Months" FROM orderstatus
```

Results | Explain | Describe | Saved SQL | History

Months
0
-
-
0
0
12
-
-

8 rows returned in 0.07 seconds Download

8.Display current date and current date as 'Mon/DD/YYYY Day'

select SYSDATE,to_char(SYSDATE,'mm/dd/yyyy,Day') from dual

```
1  select SYSDATE,to_char(SYSDATE,'mm/dd/yyyy,Day') from dual
```

Results | Explain | Describe | Saved SQL | History

SYSDATE	TO_CHAR(SYSDATE,'MM/DD/YYYY,DAY')
04/13/2022	04/13/2022,Wednesday

1 rows returned in 0.06 seconds Download

Result:

Thus the built in functions in SQL are executed successfully.

20EUCS010 - ANGELIN VARGHESE

EX.NO: 5	**IMPLEMENTATION OF SIMPLE QUERIES**
DATE:	

Aim:

To perform simple queries in SQL.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Perform simple SQL queries.
- Understand the use of simple queries.

System and Software tools required:

Software Required: Oracle

Operating System: WINDOWS 2000 / XP / NT OR LINUX OR UBUNTU

Computers Required: Minimum Requirement: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk.

Test Cases:

1. Find all departments located in location whose id is 1700

```
mysql> SELECT * FROM departments where location_id=1700;
+---------------+-----------------+-------------+
| department_id | department_name | location_id |
+---------------+-----------------+-------------+
|             1 | Administration  |        1700 |
|             3 | Purchasing      |        1700 |
|             9 | Executive       |        1700 |
|            10 | Finance         |        1700 |
|            11 | Accounting      |        1700 |
+---------------+-----------------+-------------+
5 rows in set (0.09 sec)
```

2. Find all employees whose department_id is not 8

```
mysql> SELECT * FROM employees WHERE department_id!=8;
+-------------+------------+------------+----------------------------------+--------------+------------+--------+----------+------------+---------------+
| employee_id | first_name | last_name  | email                            | phone_number | hire_date  | job_id | salary   | manager_id | department_id |
+-------------+------------+------------+----------------------------------+--------------+------------+--------+----------+------------+---------------+
|         100 | Steven     | King       | steven.king@sqltutorial.org      | 515.123.4567 | 1987-06-17 |      4 | 24000.00 |       NULL |             9 |
|         101 | Neena      | Kochhar    | neena.kochhar@sqltutorial.org    | 515.123.4568 | 1989-09-21 |      5 | 17000.00 |        100 |             9 |
|         102 | Lex        | De Haan    | lex.de haan@sqltutorial.org      | 515.123.4569 | 1993-01-13 |      5 | 17000.00 |        100 |             9 |
|         103 | Alexander  | Hunold     | alexander.hunold@sqltutorial.org | 590.423.4567 | 1990-01-03 |      9 |  9000.00 |        102 |             6 |
|         104 | Bruce      | Ernst      | bruce.ernst@sqltutorial.org      | 590.423.4568 | 1991-05-21 |      9 |  6000.00 |        103 |             6 |
|         105 | David      | Austin     | david.austin@sqltutorial.org     | 590.423.4569 | 1997-06-25 |      9 |  4800.00 |        103 |             6 |
|         106 | Valli      | Pataballa  | valli.pataballa@sqltutorial.org  | 590.423.4560 | 1998-02-05 |      9 |  4800.00 |        103 |             6 |
|         107 | Diana      | Lorentz    | diana.lorentz@sqltutorial.org    | 590.423.5567 | 1999-02-07 |      9 |  4200.00 |        103 |             6 |
|         108 | Nancy      | Greenberg  | nancy.greenberg@sqltutorial.org  | 515.124.4569 | 1994-08-17 |      7 | 12000.00 |        101 |            10 |
|         109 | Daniel     | Faviet     | daniel.faviet@sqltutorial.org    | 515.124.4169 | 1994-08-16 |      6 |  9000.00 |        108 |            10 |
|         110 | John       | Chen       | john.chen@sqltutorial.org        | 515.124.4269 | 1997-09-28 |      6 |  8200.00 |        108 |            10 |
|         111 | Ismael     | Sciarra    | ismael.sciarra@sqltutorial.org   | 515.124.4369 | 1997-09-30 |      6 |  7700.00 |        108 |            10 |
|         112 | Jose Manuel | Urman     | jose manuel.urman@sqltutorial.org | 515.124.4469 | 1998-03-07 |      6 |  7800.00 |        108 |            10 |
|         113 | Luis       | Popp       | luis.popp@sqltutorial.org        | 515.124.4567 | 1999-12-07 |      6 |  6900.00 |        108 |            10 |
|         114 | Den        | Raphaely   | den.raphaely@sqltutorial.org     | 515.127.4561 | 1994-12-07 |     14 | 11000.00 |        100 |             3 |
|         115 | Alexander  | Khoo       | alexander.khoo@sqltutorial.org   | 515.127.4562 | 1995-05-18 |     13 |  3100.00 |        114 |             3 |
|         116 | Shelli     | Baida      | shelli.baida@sqltutorial.org     | 515.127.4563 | 1997-12-24 |     13 |  2900.00 |        114 |             3 |
|         117 | Sigal      | Tobias     | sigal.tobias@sqltutorial.org     | 515.127.4564 | 1997-07-24 |     13 |  2800.00 |        114 |             3 |
|         118 | Guy        | Himuro     | guy.himuro@sqltutorial.org       | 515.127.4565 | 1998-11-15 |     13 |  2600.00 |        114 |             3 |
|         119 | Karen      | Colmenares | karen.colmenares@sqltutorial.org | 515.127.4566 | 1999-08-10 |     13 |  2500.00 |        114 |             3 |
|         120 | Matthew    | Weiss      | matthew.weiss@sqltutorial.org    | 650.123.1234 | 1996-07-18 |     19 |  8000.00 |        100 |             5 |
|         121 | Adam       | Fripp      | adam.fripp@sqltutorial.org       | 650.123.2234 | 1997-04-10 |     19 |  8200.00 |        100 |             5 |
|         122 | Payam      | Kaufling   | payam.kaufling@sqltutorial.org   | 650.123.3234 | 1995-05-01 |     19 |  7900.00 |        100 |             5 |
|         123 | Shanta     | Vollman    | shanta.vollman@sqltutorial.org   | 650.123.4234 | 1997-10-10 |     19 |  6500.00 |        100 |             5 |
|         126 | Irene      | Mikkilineni | irene.mikkilineni@sqltutorial.org | 650.124.1224 | 1998-09-28 |     18 |  2700.00 |        120 |             5 |
|         192 | Sarah      | Bell       | sarah.bell@sqltutorial.org       | 650.501.1876 | 1996-02-04 |     17 |  4000.00 |        123 |             5 |
|         193 | Britney    | Everett    | britney.everett@sqltutorial.org  | 650.501.2876 | 1997-03-03 |     17 |  3900.00 |        123 |             5 |
|         200 | Jennifer   | Whalen     | jennifer.whalen@sqltutorial.org  | 515.123.4444 | 1987-09-17 |      3 |  4400.00 |        101 |             1 |
|         201 | Michael    | Hartstein  | michael.hartstein@sqltutorial.org | 515.123.5555 | 1996-02-17 |     10 | 13000.00 |        100 |             2 |
|         202 | Pat        | Fay        | pat.fay@sqltutorial.org          | 603.123.6666 | 1997-08-17 |     11 |  6000.00 |        201 |             2 |
|         203 | Susan      | Mavris     | susan.mavris@sqltutorial.org     | 515.123.7777 | 1994-06-07 |      8 |  6500.00 |        101 |             4 |
|         204 | Hermann    | Baer       | hermann.baer@sqltutorial.org     | 515.123.8888 | 1994-06-07 |     12 | 10000.00 |        101 |             7 |
|         205 | Shelley    | Higgins    | shelley.higgins@sqltutorial.org  | 515.123.8080 | 1994-06-07 |      2 | 12000.00 |        101 |            11 |
|         206 | William    | Gietz      | william.gietz@sqltutorial.org    | 515.123.8181 | 1994-06-07 |      1 |  8300.00 |        205 |            11 |
+-------------+------------+------------+----------------------------------+--------------+------------+--------+----------+------------+---------------+
34 rows in set (0.00 sec)
```

3. Find employee_id, first_name and last_name and salary for those salary is greater than 10,000

```
mysql> SELECT employee_id, first_name, last_name, salary FROM employees WHERE salary > 10000;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         100 | Steven     | King      | 24000.00 |
|         101 | Neena      | Kochhar   | 17000.00 |
|         102 | Lex        | De Haan   | 17000.00 |
|         108 | Nancy      | Greenberg | 12000.00 |
|         114 | Den        | Raphaely  | 11000.00 |
|         145 | John       | Russell   | 14000.00 |
|         146 | Karen      | Partners  | 13500.00 |
|         201 | Michael    | Hartstein | 13000.00 |
|         205 | Shelley    | Higgins   | 12000.00 |
+-------------+------------+-----------+----------+
9 rows in set (0.00 sec)
```

4. Find all employees whose first_name starts with ‘Da’.

```
mysql> SELECT * FROM employees WHERE first_name like('Da%');
+-------------+------------+-----------+--------------------------+--------------+------------+--------+----------+------------+---------------+
| employee_id | first_name | last_name | email                    | phone_number | hire_date  | job_id | salary   | manager_id | department_id |
+-------------+------------+-----------+--------------------------+--------------+------------+--------+----------+------------+---------------+
|         105 | David      | Austin    | david.austin@sqltutorial.org | 590.423.4569 | 1997-06-25 |      9 | 4800.00 |        103 |             6 |
|         109 | Daniel     | Faviet    | daniel.faviet@sqltutorial.org | 515.124.4169 | 1994-08-16 |      6 | 9000.00 |        108 |            10 |
+-------------+------------+-----------+--------------------------+--------------+------------+--------+----------+------------+---------------+
2 rows in set (0.00 sec)
```

5. Find employee_id, first_name and last_name and salary for those whose salary is greater than 10,00 and display the result in descending order.

```
mysql> SELECT employee_id, first_name, last_name, salary FROM employees WHERE salary > 10000 ORDER BY salary DESC;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         100 | Steven     | King      | 24000.00 |
|         101 | Neena      | Kochhar   | 17000.00 |
|         102 | Lex        | De Haan   | 17000.00 |
|         145 | John       | Russell   | 14000.00 |
|         146 | Karen      | Partners  | 13500.00 |
|         201 | Michael    | Hartstein | 13000.00 |
|         108 | Nancy      | Greenberg | 12000.00 |
|         205 | Shelley    | Higgins   | 12000.00 |
|         114 | Den        | Raphaely  | 11000.00 |
+-------------+------------+-----------+----------+
9 rows in set (0.00 sec)
```

6. Find employee_id, first_name and last_name and salary for those whose salary is greater than 10,00 and display the result in descending and ascending order.

```
mysql> SELECT employee_id, first_name, last_name, salary FROM employees WHERE salary > 10000 ORDER BY salary DESC;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         100 | Steven     | King      | 24000.00 |
|         101 | Neena      | Kochhar   | 17000.00 |
|         102 | Lex        | De Haan   | 17000.00 |
|         145 | John       | Russell   | 14000.00 |
|         146 | Karen      | Partners  | 13500.00 |
|         201 | Michael    | Hartstein | 13000.00 |
|         108 | Nancy      | Greenberg | 12000.00 |
|         205 | Shelley    | Higgins   | 12000.00 |
|         114 | Den        | Raphaely  | 11000.00 |
+-------------+------------+-----------+----------+
9 rows in set (0.00 sec)

mysql> SELECT employee_id, first_name, last_name, salary FROM employees WHERE salary > 10000 ORDER BY salary;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         114 | Den        | Raphaely  | 11000.00 |
|         108 | Nancy      | Greenberg | 12000.00 |
|         205 | Shelley    | Higgins   | 12000.00 |
|         201 | Michael    | Hartstein | 13000.00 |
|         146 | Karen      | Partners  | 13500.00 |
|         145 | John       | Russell   | 14000.00 |
|         101 | Neena      | Kochhar   | 17000.00 |
|         102 | Lex        | De Haan   | 17000.00 |
|         100 | Steven     | King      | 24000.00 |
+-------------+------------+-----------+----------+
9 rows in set (0.01 sec)
```

7. Find employee_id, first_name, last_name whose first_name ends with ‘er’.

```
mysql> SELECT employee_id, first_name, last_name FROM employees WHERE first_name like('%er');
+-------------+------------+-----------+
| employee_id | first_name | last_name |
+-------------+------------+-----------+
|         103 | Alexander  | Hunold    |
|         115 | Alexander  | Khoo      |
|         200 | Jennifer   | Whalen    |
+-------------+------------+-----------+
3 rows in set (0.00 sec)
```

8. Find employee_id, first_name, last_name whose first_name starts with Jo and followed by at most two characters.

```
mysql> SELECT employee_id, first_name, last_name FROM employees WHERE first_name like('Jo__');
+-------------+------------+-----------+
| employee_id | first_name | last_name |
+-------------+------------+-----------+
|         110 | John       | Chen      |
|         145 | John       | Russell   |
+-------------+------------+-----------+
2 rows in set (0.00 sec)
```

9. Find employee_id and salary whose salaries are between 2,500 and 2,900

```
mysql> SELECT employee_id, salary FROM employees WHERE salary BETWEEN 2500 AND 2900;
+-------------+---------+
| employee_id | salary  |
+-------------+---------+
|         116 | 2900.00 |
|         117 | 2800.00 |
|         118 | 2600.00 |
|         119 | 2500.00 |
|         126 | 2700.00 |
+-------------+---------+
5 rows in set (0.00 sec)
```

10. Find employee_id and salary whose salaries are between 2,500 and 2,900 without BETWEEN keyword.

```
mysql> SELECT employee_id, salary FROM employees WHERE salary >= 2500 AND salary <= 2900;
+-------------+---------+
| employee_id | salary  |
+-------------+---------+
|         116 | 2900.00 |
|         117 | 2800.00 |
|         118 | 2600.00 |
|         119 | 2500.00 |
|         126 | 2700.00 |
+-------------+---------+
5 rows in set (0.00 sec)
```

11. Find the first_name and last_name of employees who've both job_id=9 and salary greater than 5,000.

```
mysql> SELECT first_name, last_name FROM employees WHERE job_id=9 AND salary>5000;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Alexander  | Hunold    |
| Bruce      | Ernst     |
+------------+-----------+
2 rows in set (0.00 sec)
```

12. Find employee_id, first_name, last_name and job_id whose job_id is 8 or 9 or 10.

```
mysql> SELECT employee_id, first_name, last_name, job_id FROM employees WHERE job_id=8 OR job_id=9 OR job_id=10;
+-------------+------------+-----------+--------+
| employee_id | first_name | last_name | job_id |
+-------------+------------+-----------+--------+
|         103 | Alexander  | Hunold    |      9 |
|         104 | Bruce      | Ernst     |      9 |
|         105 | David      | Austin    |      9 |
|         106 | Valli      | Pataballa |      9 |
|         107 | Diana      | Lorentz   |      9 |
|         201 | Michael    | Hartstein |     10 |
|         203 | Susan      | Mavris    |      8 |
+-------------+------------+-----------+--------+
7 rows in set (0.00 sec)

mysql> SELECT employee_id, first_name, last_name, job_id FROM employees WHERE job_id IN(8,9,10);
+-------------+------------+-----------+--------+
| employee_id | first_name | last_name | job_id |
+-------------+------------+-----------+--------+
|         103 | Alexander  | Hunold    |      9 |
|         104 | Bruce      | Ernst     |      9 |
|         105 | David      | Austin    |      9 |
|         106 | Valli      | Pataballa |      9 |
|         107 | Diana      | Lorentz   |      9 |
|         201 | Michael    | Hartstein |     10 |
|         203 | Susan      | Mavris    |      8 |
+-------------+------------+-----------+--------+
7 rows in set (0.00 sec)
```

13. Find the first name, last name and job id of employees whose job id is 8 or 9 or 10 and order them based on entries in job_id column.

```
mysql> SELECT employee_id, first_name, last_name, job_id FROM employees WHERE job_id IN(8,9,10) ORDER BY job_id;
+-------------+------------+-----------+--------+
| employee_id | first_name | last_name | job_id |
+-------------+------------+-----------+--------+
|         203 | Susan      | Mavris    |      8 |
|         103 | Alexander  | Hunold    |      9 |
|         104 | Bruce      | Ernst     |      9 |
|         105 | David      | Austin    |      9 |
|         106 | Valli      | Pataballa |      9 |
|         107 | Diana      | Lorentz   |      9 |
|         201 | Michael    | Hartstein |     10 |
+-------------+------------+-----------+--------+
7 rows in set (0.00 sec)
```

14. Find the first name, last name and job id of employees whose job id is 8 or 9 or 10 and order them based on job_id column with a limit of 3 entries.

```
+------------+-----------+--------+
| first_name | last_name | job_id |
+------------+-----------+--------+
| Susan      | Mavris    |      8 |
| Alexander  | Hunold    |      9 |
| Bruce      | Ernst     |      9 |
```

15. Find department id of marketing and sales department.

```
mysql> SELECT department_id FROM departments WHERE department_name IN('Marketing','Sales');
+---------------+
| department_id |
+---------------+
|             2 |
|             8 |
+---------------+
2 rows in set (0.00 sec)
```

Result:

Thus, the simple SQL queries are executed successfully.

20EUCS010 - ANGELIN VARGHESE

Implementation of Nested Queries

Ex.No: 6

AIM:

To perform nested queries in SQL.

LEARNING OUTCOMES:

After the completion of this experiment, students will be able to

- Perform nested queries operations.
- Understand the need of nested queries in real time scenarios.

System and Software Required:

- **Software Required:** SQL
- **Operating System:** WINDOWS 2000/XP/NT OR UBUNTU
- **Computer Required:** Minimum Required: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk

TEST CASES:

1.Find employee_id, first_name, last_name, department_id of the employees who work in Marketing and Sales.

```
mysql> SELECT employee_id,first_name,last_name, department_id from employees where department_id=(select department_id from departments where department_name='Marketin
g') ;
+-------------+------------+-----------+---------------+
| employee_id | first_name | last_name | department_id |
+-------------+------------+-----------+---------------+
|         201 | Michael    | Hartstein |             2 |
|         202 | Pat        | Fay       |             2 |
+-------------+------------+-----------+---------------+
2 rows in set (0.00 sec)
```

2.Find employee_id, first_name, last_name and salary of the employees who have highest salary.

```
mysql> SELECT employee_id, first_name,last_name,salary from  employees where salary =(select max(salary) from employees);
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         100 | Steven     | King      | 24000.00 |
+-------------+------------+-----------+----------+
1 row in set (0.05 sec)
```

3.Find department_id, and salary of the employees who have highest salary in each department.

```
mysql> select department_id,max(salary) from employees group by department_id;
+---------------+-------------+
| department_id | max(salary) |
+---------------+-------------+
|             1 |     4400.00 |
|             2 |    13000.00 |
|             3 |    11000.00 |
|             4 |     6500.00 |
|             5 |     8200.00 |
|             6 |     9000.00 |
|             7 |    10000.00 |
|             8 |    14000.00 |
|             9 |    24000.00 |
|            10 |    12000.00 |
|            11 |    12000.00 |
+---------------+-------------+
11 rows in set (0.00 sec)
```

4.Findemployee_id, first_name, last_name and salary of the employees whose salary are greater than the lowest salary of every department.

```
mysql> SELECT
    ->     employee_id, first_name, last_name, salary
    -> FROM
    ->     employees
    -> WHERE
    ->     salary >= ALL (SELECT
    ->             MIN(salary)
    ->         FROM
    ->             employees
    ->         GROUP BY department_id)
    -> ORDER BY first_name , last_name;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|         102 | Lex        | De Haan   | 17000.00 |
|         101 | Neena      | Kochhar   | 17000.00 |
|         100 | Steven     | King      | 24000.00 |
+-------------+------------+-----------+----------+
3 rows in set (0.02 sec)
```

5.Find all employees who do not locate at the location 1700.

```
mysql> select first_name,last_name from employees where department_id not in(select department_id from departments where location_id<>1700);
+-------------+-----------+
| first_name  | last_name |
+-------------+-----------+
| Steven      | King      |
| Neena       | Kochhar   |
| Lex         | De Haan   |
| Nancy       | Greenberg |
| Daniel      | Faviet    |
| John        | Chen      |
| Ismael      | Sciarra   |
| Jose Manuel | Urman     |
| Luis        | Popp      |
| Den         | Raphaely  |
| Alexander   | Khoo      |
| Shelli      | Baida     |
| Sigal       | Tobias    |
| Guy         | Himuro    |
| Karen       | Colmenares|
| Jennifer    | Whalen    |
| Shelley     | Higgins   |
| William     | Gietz     |
+-------------+-----------+
18 rows in set (0.01 sec)
```

6.Find all the employees whose salary is greater than or equal to the highest salary of any department.

```
mysql> SELECT
    ->     employee_id, first_name, last_name, salary
    -> FROM
    ->     employees
    -> WHERE
    ->     salary >= SOME (SELECT
    ->             MAX(salary)
    ->         FROM
    ->             employees
    ->         GROUP BY department_id);
+-------------+-------------+------------+----------+
| employee_id | first_name  | last_name  | salary   |
+-------------+-------------+------------+----------+
|         100 | Steven      | King       | 24000.00 |
|         101 | Neena       | Kochhar    | 17000.00 |
|         102 | Lex         | De Haan    | 17000.00 |
|         103 | Alexander   | Hunold     |  9000.00 |
|         104 | Bruce       | Ernst      |  6000.00 |
|         105 | David       | Austin     |  4800.00 |
|         106 | Valli       | Pataballa  |  4800.00 |
|         108 | Nancy       | Greenberg  | 12000.00 |
|         109 | Daniel      | Faviet     |  9000.00 |
|         110 | John        | Chen       |  8200.00 |
|         111 | Ismael      | Sciarra    |  7700.00 |
|         112 | Jose Manuel | Urman      |  7800.00 |
|         113 | Luis        | Popp       |  6900.00 |
|         114 | Den         | Raphaely   | 11000.00 |
|         120 | Matthew     | Weiss      |  8000.00 |
|         121 | Adam        | Fripp      |  8200.00 |
|         122 | Payam       | Kaufling   |  7900.00 |
|         123 | Shanta      | Vollman    |  6500.00 |
|         145 | John        | Russell    | 14000.00 |
|         146 | Karen       | Partners   | 13500.00 |
|         176 | Jonathon    | Taylor     |  8600.00 |
|         177 | Jack        | Livingston |  8400.00 |
|         178 | Kimberely   | Grant      |  7000.00 |
|         179 | Charles     | Johnson    |  6200.00 |
|         200 | Jennifer    | Whalen     |  4400.00 |
|         201 | Michael     | Hartstein  | 13000.00 |
|         202 | Pat         | Fay        |  6000.00 |
|         203 | Susan       | Mavris     |  6500.00 |
|         204 | Hermann     | Baer       | 10000.00 |
|         205 | Shelley     | Higgins    | 12000.00 |
|         206 | William     | Gietz      |  8300.00 |
+-------------+-------------+------------+----------+
31 rows in set (0.00 sec)
```

7.Find all employees who have no dependence

```
mysql> select first_name,last_name from employees where employee_id not in(select employee_id from dependents);
+------------+-------------+
| first_name | last_name   |
+------------+-------------+
| Matthew    | Weiss       |
| Adam       | Fripp       |
| Payam      | Kaufling    |
| Shanta     | Vollman     |
| Irene      | Mikkilineni |
| Jack       | Livingston  |
| Kimberely  | Grant       |
| Charles    | Johnson     |
| Sarah      | Bell        |
| Britney    | Everett     |
+------------+-------------+
10 rows in set (0.00 sec)
```

Result: Thus, the nested SQL queries are executed successfully.