N.S.S. COLLEGE OF ENGINEERING

PALAKKAD – 678 008

DEPT. OF COMPUTER SCIENCE
&
ENGINEERING

CSL 333 Database Management Systems Lab

LABORATORY RECORD

N.S.S. COLLEGE OF ENGINEERING

Name …….

Class ……. ***Roll No*** ……..

Certified that this is the bonafide record of the work done by the above student in the

Subject Code:	**Subject:**

Staff-in-charge *Head of the Department*

Submitted for practical examination held on ..……

Registration No:

Internal Examiner *External Examiner*

Contents

Exp. No. 1 Date: 19-11-2021

DESIGN ER DIAGRAM AND DATABASE SCHEMA

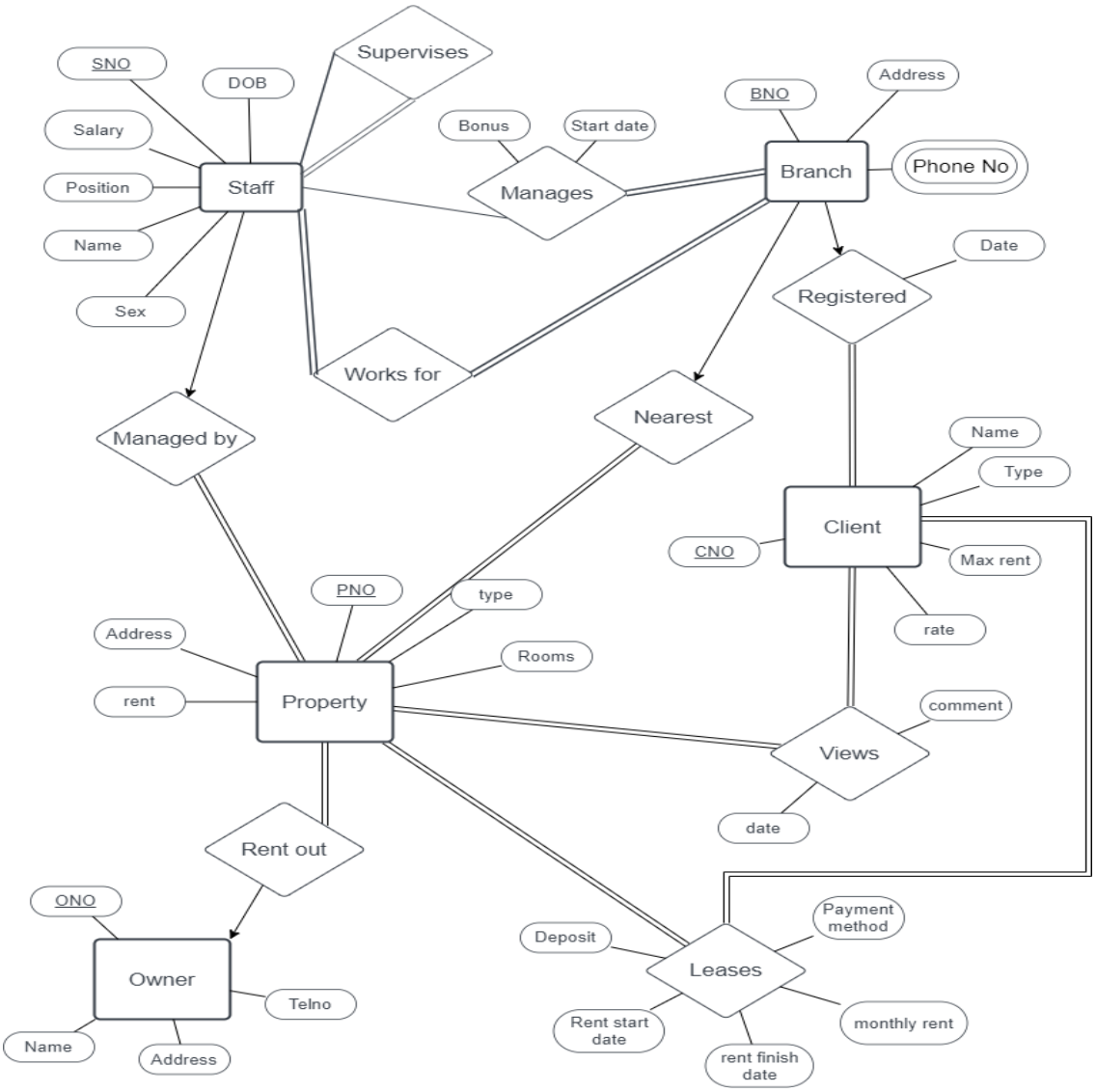
AIM:

Design a database schema for an application with ER diagram from a problem description. The report should include: Experiment No., Date, Student Name, Roll No., Aim, Requirements (Summary), E-R diagram, Relational Schema. You must correctly identify 'keys' in both conceptual and logical design and mark the same in E-R diagram and relational schema.

REQUIREMENT:

The purpose of the Dream Home Database System is to maintain the data that is used to generate to support the property rentals business for our client and property owners and to facilitate the cooperation and sharing of information between branches. They play an intermediate role between branches. They play an intermediate role between owners who wish to rent out their property and client who require to rent the furnished property for a fixed period.

ER DIAGRAM:



RELATIONAL SCHEMA:

Staff

SNO	Name	DOB	Sex	Position	Salary	SSNO	BNO

Branch

BNO	Address	Mbonus	MstartDate	MSNO

Owner

ONO	Name	Address	Telno

Property

PNO	Type	Rooms	Rent	Address	Owno	Msnq	Lrent

Deposit	Duration	Rentstart	Date	Rentfinish	Bno	Payment	CNO

Client

CNO	Name	Type	Max rent	Date	Rsno	Bnq

Viewed_by

CNO	PNO	Date	Comment

Branch_phone_no

BNO	Phone no

CONCLUSION:

The database schema design for an application with ER diagram for a given problem description was completed successfully

Exp. No. 2 Date: 22/11/2021

DDL Commands I

AIM:

Creation of DreamHome database schema - DDL (create tables, set constraints, enforce relationships)

THEORETICAL BACKGROUND:

DDL or Data Definition Language actually consists of the 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 the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
DROP: This command is used to delete objects from the database.
ALTER: This is used to alter the structure of the database.
TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.
COMMENT: This is used to add comments to the data dictionary.
RENAME: This is used to rename an object existing in the database.

QUERIES:

CREATE DATABASE DREAMHOME46;

USE DREAMHOME46;

CREATE TABLE STAFF_46(S_no int PRIMARY KEY, Name Varchar(30), dob date, sex varchar(10), position varchar(20), salary decimal(10,2),Ss_no int,FOREIGN KEY (Ss_no) references STAFF_46(S_no), B_no int);

CREATE TABLE BRANCH_46(B_no int PRIMARY KEY,Address varchar(30), M_bonus decimal(10,2), M_start_date date, Msno int,FOREIGN KEY (Msno) REFERENCES STAFF_46(S_no));

ALTER TABLE STAFF_46 ADD FOREIGN KEY (B_no) REFERENCES BRANCH_46(B_no);

CREATE TABLE OWNER_46(No int PRIMARY KEY, name varchar(30), address varchar(50), telno Decimal(10,0));

CREATE TABLE PROPERTY_46(P_no int PRIMARY KEY, type varchar(10), rooms int, rent decimal(10,2), address varchar(50), ow_no int, FOREIGN KEY(ow_no) REFERENCES OWNER_46(No), msno int,FOREIGN KEY (msno) REFERENCES STAFF_46(S_no), Lrent decimal(10,2), deposit decimal(10,2), duration varchar(30) , rent_start_date date, rent_finish_date date, B_no int,FOREIGN KEY (B_no) REFERENCES BRANCH_46(B_no), payment decimal(10,2),C_no int);

CREATE TABLE CLIENT_46 (C_no int PRIMARY KEY, name varchar(30), type varchar(10), max_rent decimal(10,2), date date, rsno int,FOREIGN KEY (rsno) references STAFF_46(S_no), B_no int,FOREIGN KEY (B_no) references BRANCH_46(B_no));

ALTER TABLE PROPERTY_46 ADD FOREIGN KEY (C_no) REFERENCES CLIENT_46(C_no);
CREATE TABLE VIEWED_BY_46(C_no int,FOREIGN KEY (C_no) references CLIENT_46(C_no), pno int,FOREIGN KEY (pno) REFERENCES PROPERTY_46(P_no), date date, comment varchar(20), PRIMARY KEY (C_no, pno));

CREATE TABLE BRANCH_PHONENO_46 (B_no int,FOREIGN KEY (B_no) REFERENCES BRANCH_46(B_no), phoneno decimal(10,0), PRIMARY KEY (B_no, phoneno));

RESULT:

```
mysql> DESC STAFF_46;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| S_no     | int           | NO   | PRI | NULL    |       |
| Name     | varchar(30)   | YES  |     | NULL    |       |
| dob      | date          | YES  |     | NULL    |       |
| sex      | varchar(10)   | YES  |     | NULL    |       |
| position | varchar(20)   | YES  |     | NULL    |       |
| salary   | decimal(10,2) | YES  |     | NULL    |       |
| Ss_no    | int           | YES  | MUL | NULL    |       |
| B_no     | int           | YES  | MUL | NULL    |       |
+----------+---------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

```
mysql> DESC CLIENT_46;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| C_no     | int           | NO   | PRI | NULL    |       |
| name     | varchar(30)   | YES  |     | NULL    |       |
| type     | varchar(10)   | YES  |     | NULL    |       |
| max_rent | decimal(10,2) | YES  |     | NULL    |       |
| date     | date          | YES  |     | NULL    |       |
| rsno     | int           | YES  | MUL | NULL    |       |
| B_no     | int           | YES  | MUL | NULL    |       |
+----------+---------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

```
mysql> DESC OWNER_46;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| No      | int           | NO   | PRI | NULL    |       |
| name    | varchar(30)   | YES  |     | NULL    |       |
| address | varchar(50)   | YES  |     | NULL    |       |
| telno   | decimal(10,0) | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC BRANCH_PHONENO_46;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| B_no    | int           | NO   | PRI | NULL    |       |
| phoneno | decimal(10,0) | NO   | PRI | NULL    |       |
+---------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> DESC PROPERTY_46;
+------------------+---------------+------+-----+---------+-------+
| Field            | Type          | Null | Key | Default | Extra |
+------------------+---------------+------+-----+---------+-------+
| P_no             | int           | NO   | PRI | NULL    |       |
| type             | varchar(10)   | YES  |     | NULL    |       |
| rooms            | int           | YES  |     | NULL    |       |
| rent             | decimal(10,2) | YES  |     | NULL    |       |
| address          | varchar(50)   | YES  |     | NULL    |       |
| ow_no            | int           | YES  | MUL | NULL    |       |
| msno             | int           | YES  | MUL | NULL    |       |
| Lrent            | decimal(10,2) | YES  |     | NULL    |       |
| deposit          | decimal(10,2) | YES  |     | NULL    |       |
| duration         | varchar(30)   | YES  |     | NULL    |       |
| rent_start_date  | date          | YES  |     | NULL    |       |
| rent_finish_date | date          | YES  |     | NULL    |       |
| B_no             | int           | YES  | MUL | NULL    |       |
| payment          | decimal(10,2) | YES  |     | NULL    |       |
| C_no             | int           | YES  | MUL | NULL    |       |
+------------------+---------------+------+-----+---------+-------+
15 rows in set (0.00 sec)
```

```
mysql> DESC CLIENT_46;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| C_no     | int           | NO   | PRI | NULL    |       |
| name     | varchar(30)   | YES  |     | NULL    |       |
| type     | varchar(10)   | YES  |     | NULL    |       |
| max_rent | decimal(10,2) | YES  |     | NULL    |       |
| date     | date          | YES  |     | NULL    |       |
| rsno     | int           | YES  | MUL | NULL    |       |
| B_no     | int           | YES  | MUL | NULL    |       |
+----------+---------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

```
mysql> DESC VIEWED_BY_46;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| C_no    | int         | NO   | PRI | NULL    |       |
| pno     | int         | NO   | PRI | NULL    |       |
| date    | date        | YES  |     | NULL    |       |
| comment | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

CONCLUSION:

Successfully created database schema using DDL commands

Exp. No. 3 Date: 03/12/2021

DDL Commands II

AIM:

Modify the 'DreamHome' database structure according to the given schema

DreamHome

1. Branch (branchNo, street, city, postcode)
2. Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
3. PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo,
4. branchNo)
5. Client (clientNo, fName, lName, telNo, prefType, maxRent)
6. PrivateOwner (ownerNo, fName, lName, address, telNo)
7. Viewing (clientNo, propertyNo, viewDate, comment) [NB: Composite primary key]
8. Registration (clientNo, branchNo, staffNo, dateJoined) [NB: Composite primary key]

   [NB: Foreign Keys must be inferred from the relations].

THEORETICAL BACKGROUND:

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

Syntax:
The basic syntax of an ALTER TABLE command to add a New Column in an existing table is as follows.

ALTER TABLE table_name ADD column_name datatype;

QUERIES:

```
DROP TABLE BRANCH_PHONENO_46;
ALTER TABLE STAFF_46 DROP FOREIGN KEY STAFF_46_ibfk_1;
ALTER TABLE STAFF_46 DROP FOREIGN KEY STAFF_46_ibfk_2,DROP KEY B_no;
ALTER TABLE PROPERTY_46 DROP FOREIGN KEY PROPERTY_46_ibfk_1, DROP KEY ow_no;
ALTER TABLE PROPERTY_46 DROP FOREIGN KEY PROPERTY_46_ibfk_2, DROP KEY msno;
ALTER TABLE PROPERTY_46 DROP FOREIGN KEY PROPERTY_46_ibfk_3, DROP KEY B_no;
ALTER TABLE VIEWED_BY_46 DROP FOREIGN KEY VIEWED_BY_46_ibfk_1;
ALTER TABLE VIEWED_BY_46 DROP FOREIGN KEY VIEWED_BY_46_ibfk_2, DROP KEY pno;
ALTER TABLE CLIENT_46 DROP FOREIGN KEY CLIENT_46_ibfk_1;
ALTER TABLE CLIENT_46 DROP FOREIGN KEY CLIENT_46_ibfk_2;

ALTER TABLE BRANCH_46 CHANGE B_no branchNo varchar(20);
ALTER TABLE BRANCH_46 DROP COLUMN Address;
ALTER TABLE BRANCH_46 ADD COLUMN street varchar(20);
ALTER TABLE BRANCH_46 ADD COLUMN city VARCHAR(20) ;
```

```
ALTER TABLE BRANCH_46 ADD COLUMN postcode varchar(10);
ALTER TABLE BRANCH_46 DROP FOREIGN KEY BRANCH_46_ibfk_1;
ALTER TABLE BRANCH_46 DROP COLUMN Msno;
ALTER TABLE BRANCH_46 DROP COLUMN M_start_date;
ALTER TABLE BRANCH_46 DROP COLUMN M_bonus;
ALTER TABLE BRANCH_46 RENAME TO branch;

ALTER TABLE STAFF_46 CHANGE S_no staffNo varchar(20);
ALTER TABLE STAFF_46 CHANGE Name fName varchar(20);
ALTER TABLE STAFF_46 CHANGE dob DOB varchar(20);
ALTER TABLE STAFF_46 DROP COLUMN Ss_no;
ALTER TABLE STAFF_46 CHANGE B_no branchNo varchar(20);
ALTER TABLE STAFF_46 ADD lName varchar(20);
ALTER TABLE STAFF_46 RENAME TO Staff;

ALTER TABLE PROPERTY_46 RENAME TO PropertyForRent;
ALTER TABLE PropertyForRent CHANGE P_no propertyNo varchar(20);
ALTER TABLE PropertyForRent CHANGE ow_no ownerNo varchar(20);
ALTER TABLE PropertyForRent ADD city varchar(20);
ALTER TABLE PropertyForRent CHANGE address street varchar(20);
ALTER TABLE PropertyForRent DROP COLUMN rent_start_date;
ALTER TABLE PropertyForRent DROP COLUMN rent_finish_date;
ALTER TABLE PropertyForRent DROP COLUMN payment;
ALTER TABLE PropertyForRent CHANGE B_no branchNo varchar(20);
ALTER TABLE PropertyForRent DROP COLUMN duration;
ALTER TABLE PropertyForRent DROP COLUMN deposit;
ALTER TABLE PropertyForRent DROP COLUMN LRent;
ALTER TABLE PropertyForRent ADD postcode varchar(20);
ALTER TABLE PropertyForRent CHANGE msno staffNo varchar(20);
ALTER TABLE PropertyForRent DROP FOREIGN KEY PropertyForRent_ibfk_4;
ALTER TABLE PropertyForRent DROP COLUMN C_no;

ALTER TABLE CLIENT_46 CHANGE C_no clientNo varchar(20);
ALTER TABLE CLIENT_46 CHANGE name fname varchar(20);
ALTER TABLE CLIENT_46 ADD lname varchar(20);
ALTER TABLE CLIENT_46 ADD telNo varchar(20);
ALTER TABLE CLIENT_46 CHANGE type prefType varchar(20);
ALTER TABLE CLIENT_46 CHANGE max_rent maxRent int;
ALTER TABLE CLIENT_46 DROP COLUMN date;
ALTER TABLE CLIENT_46 DROP COLUMN B_no;
ALTER TABLE CLIENT_46 DROP COLUMN rsno;
ALTER TABLE Client_46 RENAME TO Client;

ALTER TABLE OWNER_46 RENAME TO PrivateOwner;
ALTER TABLE PrivateOwner CHANGE No ownerNo varchar(20);
ALTER TABLE PrivateOwner CHANGE name fname varchar(20);
ALTER TABLE PrivateOwner ADD lName varchar(20);
```

ALTER TABLE privateowner CHANGE address address varchar(255);
ALTER TABLE privateowner CHANGE telNo telNo varchar(20);

ALTER TABLE VIEWED_BY_46 RENAME TO Viewing;
ALTER TABLE Viewing CHANGE C_no clientNo varchar(20);
ALTER TABLE Viewing CHANGE pno propertyNo varchar(20);
ALTER TABLE Viewing CHANGE date viewDate varchar(20);

ALTER TABLE STAFF_46 CHANGE salary salary int;
ALTER TABLE staff CHANGE sex sex varchar(20);

CREATE TABLE Registration (clientNo varchar(20), branchNo varchar(20), staffNo varchar(20), dateJoined varchar(20), PRIMARY KEY(clientNo, branchNo),FOREIGN KEY (clientNo) REFERENCES CLIENT(clientNo),FOREIGN KEY (branchNo) REFERENCES BRANCH(branchNo),FOREIGN KEY (staffNo) REFERENCES STAFF(staffNo));

ALTER TABLE Staff ADD CONSTRAINT FOREIGN KEY(branchNo) REFERENCES Branch(branchNo);
ALTER TABLE PropertyForRent ADD CONSTRAINT FOREIGN KEY(ownerNo) REFERENCES PrivateOwner(ownerNo);
ALTER TABLE PropertyForRent ADD CONSTRAINT FOREIGN KEY(staffNo) REFERENCES Staff(staffNo);
ALTER TABLE PropertyForRent ADD CONSTRAINT FOREIGN KEY(branchNo) REFERENCES Branch(branchNo);
ALTER TABLE Viewing ADD CONSTRAINT FOREIGN KEY(clientNo) REFERENCES Client(clientNo);
ALTER TABLE Viewing ADD CONSTRAINT FOREIGN KEY(propertyNo) REFERENCES PropertyForRent(propertyNo);

RESULT:

```
mysql> desc propertyforrent;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| propertyNo | varchar(20) | NO   | PRI | NULL    |       |
| type       | varchar(20) | YES  |     | NULL    |       |
| rooms      | int         | YES  |     | NULL    |       |
| rent       | int         | YES  |     | NULL    |       |
| street     | varchar(20) | YES  |     | NULL    |       |
| ownerNo    | varchar(20) | YES  | MUL | NULL    |       |
| staffNo    | varchar(20) | YES  | MUL | NULL    |       |
| branchNo   | varchar(20) | YES  | MUL | NULL    |       |
| city       | varchar(20) | YES  |     | NULL    |       |
| postcode   | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> desc branch;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| branchNo | varchar(20) | NO   | PRI | NULL    |       |
| street   | varchar(20) | YES  |     | NULL    |       |
| city     | varchar(20) | YES  |     | NULL    |       |
| postcode | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.04 sec)
```

```
mysql> desc registration;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| clientNo   | varchar(20) | NO   | PRI | NULL    |       |
| branchNo   | varchar(20) | NO   | PRI | NULL    |       |
| staffNo    | varchar(20) | YES  | MUL | NULL    |       |
| dateJoined | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.10 sec)
```

```
mysql> desc staff;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| staffNo  | varchar(20) | NO   | PRI | NULL    |       |
| fname    | varchar(20) | YES  |     | NULL    |       |
| DOB      | varchar(20) | YES  |     | NULL    |       |
| sex      | varchar(20) | YES  |     | NULL    |       |
| position | varchar(20) | YES  |     | NULL    |       |
| salary   | int         | YES  |     | NULL    |       |
| branchNo | varchar(20) | YES  | MUL | NULL    |       |
| lname    | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
8 rows in set (0.02 sec)
```

```
mysql> desc client;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| clientNo | varchar(20) | NO   | PRI | NULL    |       |
| fname    | varchar(20) | YES  |     | NULL    |       |
| maxRent  | int         | YES  |     | NULL    |       |
| lname    | varchar(20) | YES  |     | NULL    |       |
| telNo    | varchar(20) | YES  |     | NULL    |       |
| prefType | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

```
mysql> desc viewing;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| clientNo   | varchar(20) | NO   | PRI | NULL    |       |
| propertyNo | varchar(20) | NO   | PRI | NULL    |       |
| viewDate   | varchar(20) | YES  |     | NULL    |       |
| comment    | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

```
mysql> desc privateowner;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ownerNo | varchar(20)  | NO   | PRI | NULL    |       |
| fname   | varchar(20)  | YES  |     | NULL    |       |
| address | varchar(255) | YES  |     | NULL    |       |
| telNo   | varchar(20)  | YES  |     | NULL    |       |
| lName   | varchar(20)  | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.09 sec)
```

CONCLUSION:

Successfully modified ‘Dreamhome’ database structure according to the given schema

Exp. No. 4 Date: 10/12/2021

Database initialization - Data insert

AIM:

Populate the tables with given data using the INSERT command

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Client

clientNo	fName	lName	telNo	prefType	maxRent
CR76	John	Kay	0207-774-5632	Flat	425
CR56	Aline	Stewart	0141-848-1825	Flat	350
CR74	Mike	Ritchie	01475-392178	House	750
CR62	Mary	Tregear	01224-196720	Flat	600

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-04
CR56	B003	SG37	11-Apr-03
CR74	B003	SG37	16-Nov-02
CR62	B007	SA9	7-Mar-03

THEORETICAL BACKGROUND:

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax
It is possible to write the INSERT INTO statement in two ways:

Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

QUERIES:

```
insert into Branch(branchNo,street,city,postcode) VALUES('B005','22 Deer Rd','London','SW1 4EH'),
('B007','16 Argyll St','Aberdeen','AB2 3SU'),
('B003','163 Main St','Glasgow','G11 9QX'),
('B004','32 Manse Rd','Bristol','BS99 INZ'),
('B002','56 Clover Dr','London','NW10 6EU');

insert into Staff VALUES('SL21','John','White','Manager','M','1-Oct-45',30000,'B005'),
('SG37','Ann','Beech','Assistant','F','10-Nov-60',12000,'B003'),
('SG14','David','Ford','Supervisor','M','24-Mar-58',18000,'B003'),
('SA9','Mary','Howe','Assistant','F','19-Feb-70',9000,'B007'),
('SG5','Susan','Brand','Manager','F','3-Jun-40',24000,'B003'),
('SL41','Julie','Lee','Assistant','F','13-Jun-65',9000,'B005');

insert into PrivateOwner VALUES('CO46','Joe','Keogh','2 Fergus Dr,Aberdeen AB2 7SX','01224-861212'),
('CO87','Carol','Farrel','6 Achray St,Glasgow G32 9DX','0141-357-7419'),
('CO40','Tina','Murphy','63 Well St,Glasgow G42','0141-943-1728'),
('CO93','Tony','Shaw','12 Park Pl,Glasgow G4 0QR','0141-255-7025');

insert into PropertyForRent VALUES('PA14','16 Holhead','Aberdeen','AB7
5SU','House',6,650,'CO46','SA9','B007'),
('PL96','6 Argyll St','London','NW2','Flat',4,400,'CO87','SL41','B005'),
('PG4','6 Lawrence St','Glasgow','G11 9QX','Flat',3,350,'CO40',NULL,'B003'),
('PG36','2 Manor Rd','Glasgow','G32 4QX','Flat',3,375,'CO93','SG37','B003'),
('PG21','18 Dale Rd','Glasgow','G12','House',5,600,'CO87','SG37','B003'),
('PG16','5 Novar Dr','Glasgow','G12 9AX','Flat',4,450,'CO93','SG14','B003');

insert into Client VALUES ('CR76','John','Kay','0207-774-5632','Flat',425),
('CR56','Aline','Stewart','0141-848-5632','Flat',350),
('CR74','Mike','Ritchie','01475-392178','House',750),
('CR62','Mary','Tregear','01224-196720','Flat',600);

insert into Viewing VALUES('CR56','PA14','24-May-04','too small'),
('CR76','PG4','20-Apr-04','too remote'),
('CR56','PG4','26-May-04',NULL),
```

('CR62','PA14','14-May-04','no dining room'),
('CR56','PG36','28-Apr-04',NULL);

insert into Registration VALUES('CR76','B005','SL41','2-Jan-04'),
('CR56','B003','SG37','11-Apr-03'),
('CR74','B003','SG37','16-Nov-02'),
('CR62','B007','SA9','7-Mar-03');

RESULT:

```
mysql> select * from branch;
+----------+--------------+----------+----------+
| branchNo | street       | city     | postcode |
+----------+--------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU |
| B003     | 163 Main St  | Glasgow  | G11 9QX  |
| B004     | 32 Manse Rd  | Bristol  | BS99 INZ |
| B005     | 22 Deer Rd   | London   | SW1 4EH  |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  |
+----------+--------------+----------+----------+
5 rows in set (0.01 sec)
```

```
mysql> select * from client;
+----------+-------+---------+---------------+----------+---------+
| clientNo | fName | lName   | telNo         | prefType | maxRent |
+----------+-------+---------+---------------+----------+---------+
| CR56     | Aline | Stewart | 0141-848-5632 | Flat     |     350 |
| CR62     | Mary  | Tregear | 01224-196720  | Flat     |     600 |
| CR74     | Mike  | Ritchie | 01475-392178  | House    |     750 |
| CR76     | John  | Kay     | 0207-774-5632 | Flat     |     425 |
+----------+-------+---------+---------------+----------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select * from privateowner;
+---------+-------+--------+-----------------------------+---------------+
| ownerNo | fName | lName  | address                     | telNo         |
+---------+-------+--------+-----------------------------+---------------+
| CO40    | Tina  | Murphy | 63 Well St,Glasgow G42      | 0141-943-1728 |
| CO46    | Joe   | Keogh  | 2 Fergus Dr,Aberdeen AB2 7SX | 01224-861212  |
| CO87    | Carol | Farrel | 6 Achray St,Glasgow G32 9DX | 0141-357-7419 |
| CO93    | Tony  | Shaw   | 12 Park Pl,Glasgow G4 0QR   | 0141-255-7025 |
+---------+-------+--------+-----------------------------+---------------+
4 rows in set (0.03 sec)
```

```
mysql> select * from propertyforrent;
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL96       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
6 rows in set (0.01 sec)
```

```
mysql> select * from registration;
+----------+----------+---------+------------+
| clientNo | branchNo | staffNo | dateJoined |
+----------+----------+---------+------------+
| CR56     | B003     | SG37    | 11-Apr-03  |
| CR62     | B007     | SA9     | 7-Mar-03   |
| CR74     | B003     | SG37    | 16-Nov-02  |
| CR76     | B005     | SL41    | 2-Jan-04   |
+----------+----------+---------+------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from staff;
+---------+-------+-------+------------+------+-----------+--------+----------+
| staffNo | fName | lName | position   | sex  | DOB       | salary | branchNo |
+---------+-------+-------+------------+------+-----------+--------+----------+
| SA9     | Mary  | Howe  | Assistant  | F    | 19-Feb-70 |   9000 | B007     |
| SG14    | David | Ford  | Supervisor | M    | 24-Mar-58 |  18000 | B003     |
| SG37    | Ann   | Beech | Assistant  | F    | 10-Nov-60 |  12000 | B003     |
| SG5     | Susan | Brand | Manager    | F    | 3-Jun-40  |  24000 | B003     |
| SL21    | John  | White | Manager    | M    | 1-Oct-45  |  30000 | B005     |
| SL41    | Julie | Lee   | Assistant  | F    | 13-Jun-65 |   9000 | B005     |
+---------+-------+-------+------------+------+-----------+--------+----------+
6 rows in set (0.01 sec)
```

```
mysql> select * from viewing;
+----------+------------+-----------+----------------+
| clientNo | propertyNo | viewDate  | comment        |
+----------+------------+-----------+----------------+
| CR56     | PA14       | 24-May-04 | too small      |
| CR56     | PG36       | 28-Apr-04 | NULL           |
| CR56     | PG4        | 26-May-04 | NULL           |
| CR62     | PA14       | 14-May-04 | no dining room |
| CR76     | PG4        | 20-Apr-04 | too remote     |
+----------+------------+-----------+----------------+
5 rows in set (0.00 sec)
```

CONCLUSION:

Successfully performed insertion operation in the given database

Exp. No. 5 Date: 10/12/2021

DML Commands I

AIM:

Practice SQL commands for DML (insertion, updating, altering, deletion of data, and viewing/querying records based on condition in databases).

THEORETICAL BACKGROUND:

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table

QUERIES:

Part 1 - Update/Delete

1. Give all staff a 3% pay increase.

update Staff set salary = salary + (salary * 3.0 / 100.0);

```
mysql> SELECT * FROM staff;
+---------+-------+-------+------------+------+------------+--------+----------+
| staffNo | fName | lName | position   | sex  | DOB        | salary | branchNo |
+---------+-------+-------+------------+------+------------+--------+----------+
| SA9     | Mary  | Howe  | Assistant  | F    | 1970-02-19 |   9270 | B007     |
| SG14    | David | Ford  | Supervisor | M    | 1958-03-24 |  18540 | B003     |
| SG37    | Ann   | Beech | Assistant  | F    | 1960-11-10 |  12360 | B003     |
| SG5     | Susan | Brand | Manager    | F    | 1940-06-03 |  24720 | B003     |
| SL21    | John  | White | Manager    | M    | 1945-10-01 |  30900 | B005     |
| SL41    | Julie | Lee   | Assistant  | F    | 1965-06-13 |   9270 | B005     |
+---------+-------+-------+------------+------+------------+--------+----------+
6 rows in set (0.00 sec)
```

2. Give all Managers a 5% pay increase.

update Staff set salary = salary + (salary * 5.0 / 100.0) WHERE position='Manager';

```
mysql> SELECT * FROM staff;
+---------+-------+-------+------------+------+------------+--------+----------+
| staffNo | fName | lName | position   | sex  | DOB        | salary | branchNo |
+---------+-------+-------+------------+------+------------+--------+----------+
| SA9     | Mary  | Howe  | Assistant  | F    | 1970-02-19 |   9270 | B007     |
| SG14    | David | Ford  | Supervisor | M    | 1958-03-24 |  18540 | B003     |
| SG37    | Ann   | Beech | Assistant  | F    | 1960-11-10 |  12360 | B003     |
| SG5     | Susan | Brand | Manager    | F    | 1940-06-03 |  25956 | B003     |
| SL21    | John  | White | Manager    | M    | 1945-10-01 |  32445 | B005     |
| SL41    | Julie | Lee   | Assistant  | F    | 1965-06-13 |   9270 | B005     |
+---------+-------+-------+------------+------+------------+--------+----------+
6 rows in set (0.00 sec)
```

3. Update the price of all rooms by 5%.

update PropertyForRent set rent = rent + (rent * 5.0 / 100.0);

```
mysql> select * from propertyforrent;
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  683 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  473 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  630 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  394 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  368 | CO40    | NULL    | B003     |
| PL96       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  420 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
```

4. Promote David Ford (staffNo = ‘SG14’) to Manager and change his salary to £18,000.

update Staff set position='Manager',salary=18000 WHERE staffNo='SG14';

```
mysql> SELECT * FROM staff;
+---------+-------+-------+-----------+------+------------+--------+----------+
| staffNo | fName | lName | position  | sex  | DOB        | salary | branchNo |
+---------+-------+-------+-----------+------+------------+--------+----------+
| SA9     | Mary  | Howe  | Assistant | F    | 1970-02-19 |   9270 | B007     |
| SG14    | David | Ford  | Manager   | M    | 1958-03-24 |  18000 | B003     |
| SG37    | Ann   | Beech | Assistant | F    | 1960-11-10 |  12360 | B003     |
| SG5     | Susan | Brand | Manager   | F    | 1940-06-03 |  25956 | B003     |
| SL21    | John  | White | Manager   | M    | 1945-10-01 |  32445 | B005     |
| SL41    | Julie | Lee   | Assistant | F    | 1965-06-13 |   9270 | B005     |
+---------+-------+-------+-----------+------+------------+--------+----------+
6 rows in set (0.00 sec)
```

5. Delete all viewings that relate to property PG4.

DELETE FROM Viewing WHERE propertyNo='PG4';

```
mysql> select * from Viewing;
+----------+------------+-----------+----------------+
| clientNo | propertyNo | viewDate  | comment        |
+----------+------------+-----------+----------------+
| CR56     | PA14       | 24-May-04 | too small      |
| CR56     | PG36       | 28-Apr-04 | NULL           |
| CR62     | PA14       | 14-May-04 | no dining room |
+----------+------------+-----------+----------------+
```

6. Delete all rows from the Viewing table.

```
mysql> TRUNCATE Viewing;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> SELECT * FROM viewing;
Empty set (0.00 sec)
```

Part 2 - Select

1. List full details of all staff.

select * from Staff;

```
mysql> select * from Staff;
+---------+-------+-------+-----------+------+-----------+--------+----------+
| staffNo | fName | lName | position  | sex  | DOB       | salary | branchNo |
+---------+-------+-------+-----------+------+-----------+--------+----------+
| SA9     | Mary  | Howe  | Assistant | F    | 19-Feb-70 |   9270 | B007     |
| SG14    | David | Ford  | Manager   | M    | 24-Mar-58 |  18000 | B003     |
| SG37    | Ann   | Beech | Assistant | F    | 10-Nov-60 |  12360 | B003     |
| SG5     | Susan | Brand | Manager   | F    | 3-Jun-40  |  25956 | B003     |
| SL21    | John  | White | Manager   | M    | 1-Oct-45  |  32445 | B005     |
| SL41    | Julie | Lee   | Assistant | F    | 13-Jun-65 |   9270 | B005     |
+---------+-------+-------+-----------+------+-----------+--------+----------+
6 rows in set (0.00 sec)
```

2. Produce a list of salaries for all staff, showing only the staff number, the first and last names, and the salary details.

select staffNo,fName,lName,salary from Staff;

```
mysql> select staffNo,fName,lName,salary from Staff;
+---------+-------+-------+--------+
| staffNo | fName | lName | salary |
+---------+-------+-------+--------+
| SA9     | Mary  | Howe  |   9270 |
| SG14    | David | Ford  |  18000 |
| SG37    | Ann   | Beech |  12360 |
| SG5     | Susan | Brand |  25956 |
| SL21    | John  | White |  32445 |
| SL41    | Julie | Lee   |   9270 |
+---------+-------+-------+--------+
6 rows in set (0.00 sec)
```

3. List the property numbers of all properties that have been viewed.

```
mysql> SELECT DISTINCT(propertyNo) FROM viewing;
Empty set (0.00 sec)
```

4. Produce a list of monthly salaries for all staff, showing the staff number, the first and last names, and the salary details.

select staffNo,fName,lName,salary/12 as monthly_salary from Staff;

```
mysql> select staffNo,fName,lName,salary/12 as monthly_salary from Staff;
+---------+-------+-------+----------------+
| staffNo | fName | lName | monthly_salary |
+---------+-------+-------+----------------+
| SA9     | Mary  | Howe  |       772.5000 |
| SG14    | David | Ford  |      1500.0000 |
| SG37    | Ann   | Beech |      1030.0000 |
| SG5     | Susan | Brand |      2163.0000 |
| SL21    | John  | White |      2703.7500 |
| SL41    | Julie | Lee   |       772.5000 |
+---------+-------+-------+----------------+
```

5. List all staff with a salary greater than £10,000.

select * from Staff WHERE salary > 10000;

```
mysql> select * from Staff WHERE salary > 10000;
+---------+-------+-------+-----------+------+-----------+--------+----------+
| staffNo | fName | lName | position  | sex  | DOB       | salary | branchNo |
+---------+-------+-------+-----------+------+-----------+--------+----------+
| SG14    | David | Ford  | Manager   | M    | 24-Mar-58 |  18000 | B003     |
| SG37    | Ann   | Beech | Assistant | F    | 10-Nov-60 |  12360 | B003     |
| SG5     | Susan | Brand | Manager   | F    | 3-Jun-40  |  25956 | B003     |
| SL21    | John  | White | Manager   | M    | 1-Oct-45  |  32445 | B005     |
+---------+-------+-------+-----------+------+-----------+--------+----------+
```

6. List the addresses of all branch offices in London or Glasgow.

select street,city,postcode from Branch where city='London' OR city='Glasgow';

```
mysql> select street,city,postcode from Branch where city='London' OR city='Glasgow';
+--------------+---------+----------+
| street       | city    | postcode |
+--------------+---------+----------+
| 56 Clover Dr | London  | NW10 6EU |
| 163 Main St  | Glasgow | G11 9QX  |
| 22 Deer Rd   | London  | SW1 4EH  |
+--------------+---------+----------+
```

7. List all staff with a salary between £20,000 and £30,000.

select * from Staff where salary > 20000 and salary < 30000;

```
mysql> select * from Staff where salary > 20000 and salary < 30000;
+---------+-------+-------+----------+------+----------+--------+----------+
| staffNo | fName | lName | position | sex  | DOB      | salary | branchNo |
+---------+-------+-------+----------+------+----------+--------+----------+
| SG5     | Susan | Brand | Manager  | F    | 3-Jun-40 |  25956 | B003     |
+---------+-------+-------+----------+------+----------+--------+----------+
```

8. List all managers and supervisors.

select * from Staff where position='Manager' or position='Supervisor';

```
mysql> select * from Staff where position='Manager' or position='Supervisor';
+---------+-------+-------+----------+------+-----------+--------+----------+
| staffNo | fName | lName | position | sex  | DOB       | salary | branchNo |
+---------+-------+-------+----------+------+-----------+--------+----------+
| SG14    | David | Ford  | Manager  | M    | 24-Mar-58 |  18000 | B003     |
| SG5     | Susan | Brand | Manager  | F    | 3-Jun-40  |  25956 | B003     |
| SL21    | John  | White | Manager  | M    | 1-Oct-45  |  32445 | B005     |
+---------+-------+-------+----------+------+-----------+--------+----------+
```

9. Find all owners with the string ‘Glasgow’ in their address

select * from PrivateOwner where address like '%Glasgow%';

```
mysql> select * from PrivateOwner where address like '%Glasgow%';
+---------+-------+--------+----------------------------+---------------+
| ownerNo | fName | lName  | address                    | telNo         |
+---------+-------+--------+----------------------------+---------------+
| CO40    | Tina  | Murphy | 63 Well St,Glasgow G42     | 0141-943-1728 |
| CO87    | Carol | Farrel | 6 Achray St,Glasgow G32 9DX | 0141-357-7419 |
| CO93    | Tony  | Shaw   | 12 Park Pl,Glasgow G4 0QR  | 0141-255-7025 |
+---------+-------+--------+----------------------------+---------------+
```

10. List the details of all viewings on property PG4 where a comment has not been supplied.

select * from Viewing where propertyNo='PG4' AND comment=NULL;

```
mysql> select * from Viewing where propertyNo='PG4' AND comment=NULL;
Empty set (0.00 sec)
```

11. Produce a list of salaries for all staff, arranged in descending order of salary.

select salary from Staff order by salary desc;

```
mysql> select salary from Staff order by salary desc;
+--------+
| salary |
+--------+
|  32445 |
|  25956 |
|  18000 |
|  12360 |
|   9270 |
|   9270 |
+--------+
```

12. Produce an abbreviated list of properties arranged in order of property type.

select propertyNo,type from PropertyForRent order by type ASC;

```
mysql> select propertyNo,type from PropertyForRent order by type ASC;
+------------+-------+
| propertyNo | type  |
+------------+-------+
| PG16       | Flat  |
| PG36       | Flat  |
| PG4        | Flat  |
| PL96       | Flat  |
| PA14       | House |
| PG21       | House |
+------------+-------+
```

CONCLUSION:

Successfully used DML commands for manipulating data

Exp. No. 6 Date: 22/12/2021

DML Commands II

AIM:

Implementation of DQL queries involving multiple tables and various aggregate functions.

THEORETICAL BACKGROUND:

This statement is used to retrieve fields from multiple tables. To do so, we need to use join query to get data from multiple tables.

SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value. It is also used to summarize the data.

QUERIES:

A. Multi-Table Queries

1. List the names of all clients who have viewed a property along with any comment supplied.

SELECT fName, lName, comment, propertyNo FROM Client C, Viewing V
WHERE C.clientNo = V.clientNo;

fName	lName	comment	propertyNo
Aline	Stewart	too small	PA14
Aline	Stewart	NULL	PG36
Aline	Stewart	NULL	PG4
Mary	Tregar	no dining room	PA14
John	Kay	too remote	PG4

5 rows in set (0.00 sec)

2. For each branch office, list the numbers and names of staff who manage properties and the properties that they manage.

SELECT S.branchNo, S.StaffNo, fName, lName, propertyNo FROM Staff S, PropertyForRent P
WHERE S.StaffNo = P.StaffNo ORDER BY S.branchNo, S.StaffNo, propertyNo;

branchNo	StaffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

5 rows in set (0.00 sec)

3. For each branch, list the numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manages.

SELECT B.branchNo, B.city, S.StaffNo, fName, lName, propertyNo
FROM Branch B, Staff S, PropertyForRent P
where B.branchNo = S.branchNo and S.StaffNo = P.StaffNo order by B.branchNo, S.StaffNo, propertyNo;

branchNo	city	StaffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

5 rows in set (1.06 sec)

4. List all branch offices and any properties that are in the same city.

SELECT B.branchNo, B.city AS branchCity, P.propertyNo, P.city AS propertyCity
FROM Branch B LEFT OUTER JOIN PropertyForRent P ON B.city = P.city;

branchNo	branchCity	propertyNo	propertyCity
B002	London	PL94	London
B003	Glasgow	PG4	Glasgow
B003	Glasgow	PG36	Glasgow
B003	Glasgow	PG21	Glasgow
B003	Glasgow	PG16	Glasgow
B004	Bristol	NULL	NULL
B005	London	PL94	London
B007	Aberdeen	PA14	Aberdeen

8 rows in set (0.00 sec)

5. List all properties and any branch offices that are in the same city.

SELECT B.branchNo, P.city AS branchCity, P.propertyNo, P.city AS propertyCity
FROM Branch B RIGHT OUTER JOIN PropertyForRent P ON B.city = P.city;

branchNo	branchCity	propertyNo	propertyCity
B007	Aberdeen	PA14	Aberdeen
B003	Glasgow	PG16	Glasgow
B003	Glasgow	PG21	Glasgow
B003	Glasgow	PG36	Glasgow
B003	Glasgow	PG4	Glasgow
B005	London	PL94	London
B002	London	PL94	London

7 rows in set (0.24 sec)

6. List the branch offices and properties that are in the same city along with any unmatched branches or properties.

(SELECT B.branchNo, B.city AS branchCity, P.propertyNo, P.city AS propertyCity FROM Branch B LEFT OUTER JOIN PropertyForRent P ON B.city = P.city) UNION (SELECT B.branchNo, P.city AS branchCity, P.propertyNo, P.city AS propertyCity FROM Branch B RIGHT OUTER JOIN PropertyForRent P ON B.city = P.city);

branchNo	branchCity	propertyNo	propertyCity
B002	London	PL94	London
B003	Glasgow	PG4	Glasgow
B003	Glasgow	PG36	Glasgow
B003	Glasgow	PG21	Glasgow
B003	Glasgow	PG16	Glasgow
B004	Bristol	NULL	NULL
B005	London	PL94	London
B007	Aberdeen	PA14	Aberdeen

8 rows in set (0.10 sec)

B. Aggregate Functions.

7. How many properties cost more than 350 per month to rent?

select count(propertyNo) from PropertyForRent where rent>350;

count(propertyNo)
6

1 row in set (0.05 sec)

8. How many different properties were viewed in May 2004?

select count(propertyNo) from Viewing where viewDate like "2004-05%";

count(propertyNo)
3

1 row in set, 1 warning (0.00 sec)

9. Find the total number of Managers and the sum of their salaries.

select count(staffNo),Sum(salary) from Staff where position = "manager";

```
+----------------+-------------+
| count(staffNo) | Sum(salary) |
+----------------+-------------+
|              3 |    76401.00 |
+----------------+-------------+
1 row in set (0.00 sec)
```

10. Find the minimum, maximum and average staff salary.

select min(salary),max(salary),avg(salary) from Staff;

```
+-------------+-------------+--------------+
| min(salary) | max(salary) | avg(salary)  |
+-------------+-------------+--------------+
|     9270.00 |    32445.00 | 17883.500000 |
+-------------+-------------+--------------+
1 row in set (0.01 sec)
```

11. Find total number of properties with 3 rooms.

select count(propertyNo) from PropertyForRent where rooms = 3;

```
+-------------------+
| count(propertyNo) |
+-------------------+
|                 2 |
+-------------------+
1 row in set (0.01 sec)
```

12. Find minimum, maximum and average property rent.

select min(rent),max(rent),avg(rent) from PropertyForRent;

```
+-----------+-----------+------------+
| min(rent) | max(rent) | avg(rent)  |
+-----------+-----------+------------+
|    367.50 |    682.50 | 494.375000 |
+-----------+-----------+------------+
1 row in set (0.00 sec)
```

CONCLUSION:

Successfully implemented DQL Commands

Exp. No. 7 Date:03/01/2022

DML Commands III

AIM:

Implementation of Group By & Having clause.

THEORETICAL BACKGROUND:

Having Clause is basically like the aggregate function with the GROUP BY clause. The HAVING clause is used instead of WHERE with aggregate functions. While the GROUP BY Clause groups rows that have the same values into summary rows. The having clause is used with the where clause in order to find rows with certain conditions. The having clause is always used after the GROUP BY clause.

The GROUP BY clause is often used with aggregate functions (MAX, SUM, AVG) to group the results by one or more columns. In simple words, we can say that The GROUP BY clause is used in collaboration with the SELECT statement to arrange required data into groups.

The GROUP BY statement groups rows that have the same values. This Statement is used after the where clause. This statement is often used with some aggregate function like SUM, AVG, COUNT etc. to group the results by one or more columns.

QUERIES:

1. Find the number of staff working in each branch and the sum of their salaries.

select count(branchNo),branchNo,sum(salary) from Staff group by branchNo;

```
+-----------------+----------+-------------+
| count(branchNo) | branchNo | sum(salary) |
+-----------------+----------+-------------+
|               3 |        3 |    56316.00 |
|               2 |        5 |    41715.00 |
|               1 |        7 |     9270.00 |
+-----------------+----------+-------------+
3 rows in set (0.00 sec)
```

2. For each branch office with more than one member of staff, find the number of staff working in each branch and the sum of their salaries.

select count(branchNo),branchNo,sum(salary) from Staff group by branchNo having count(branchNo)>1;

```
+-----------------+----------+-------------+
| count(branchNo) | branchNo | sum(salary) |
+-----------------+----------+-------------+
|               3 |        3 |    56316.00 |
|               2 |        5 |    41715.00 |
+-----------------+----------+-------------+
```

3. Find average salaries of staff at various positions.

select avg(salary), position from Staff group by position;

```
+--------------+-----------+
| avg(salary)  | position  |
+--------------+-----------+
| 10300.000000 | Assistant |
| 25467.000000 | Manager   |
+--------------+-----------+
```

4. Display the number of properties available in each city along with the city name.

select city,count(city) from PropertyForRent group by city;

city	count(city)
Aberdeen	1
Glasgow	4
London	1

5. Display the number of properties available at each city along with the city name if there exist more than 2 properties.

select city,count(city) from PropertyForRent group by city having count(city)>2;

city	count(city)
Glasgow	4

6. Find the number of houses and flats available for rent.
select count(propertyNo),type from PropertyForRent group by type;

count(propertyNo)	type
4	Flat
2	House

7. For each city with more than one property, find number of properties within each city and average rent.

select city, count(propertyNO),avg(rent) from PropertyForRent group by city having count(propertyNO)>1;

city	count(propertyNO)	avg(rent)
Glasgow	4	465.937500

CONCLUSION:

Implementation of Group By & Having clause was successful

Exp. No. 8 Date: 03/01/2022

DML Commands IV

AIM:

Implementation of set operators and nested queries

THEORETICAL BACKGROUND:

The SQL Set operation is used to combine the two or more SQL SELECT statements.
A subquery in MySQL is a query, which is nested into another SQL query and embedded with SELECT, INSERT, UPDATE or DELETE statement along with the various operators. We can also nest the subquery with another subquery. A subquery is known as the inner query, and the query that contains the subquery is known as the outer query. The inner query executed first gives the result to the outer query, and then the main/outer query will be performed. MySQL allows us to use subquery anywhere, but it must be closed within parenthesis. All subquery forms and operations supported by the SQL standard will be supported in MySQL also.

QUERIES:

Set Operations

1. Construct a list of all cities where there is either a branch office or a property.

select city from Branch union select city from PropertyForRent;

```
+----------+
| city     |
+----------+
| London   |
| Glasgow  |
| Bristol  |
| Aberdeen |
+----------+
```

2. Construct a list of all cities where there is both a branch office and a property.

select city from Branch where city in(select city from PropertyForRent) group by city;

```
+----------+
| city     |
+----------+
| Aberdeen |
| Glasgow  |
| London   |
+----------+
```

3. Construct a list of all cities where there is a branch office but no properties.

select city from Branch where city not in(select city from PropertyForRent) group by city;

```
+---------+
| city    |
+---------+
| Bristol |
+---------+
```

Nested Queries

4. List the staffs who work in the branch at '163 Main St'.

```
SELECT StaffNo, fName, lName, Position
FROM Staff
WHERE branchNo = (SELECT branchNo
                  FROM Branch
                  WHERE street LIKE '163 Main St%');
```

StaffNo	fName	lName	Position
SG14	David	Ford	Manager
SG37	Ann	Beech	Assistant
SG5	Susan	Brand	Manager

3 rows in set (0.09 sec)

5. List all staff whose salary is greater than the average salary, and show by how much their salary is greater than the average.

```
SELECT StaffNo, fName, lName, Position, (salary - (SELECT AVG(salary) FROM Staff)) AS salDiff
FROM Staff
WHERE salary > (SELECT AVG(salary) FROM Staff);
```

StaffNo	fName	lName	Position	salDiff
SG14	David	Ford	Manager	116.500000
SG5	Susan	Brand	Manager	8072.500000
SL21	John	White	Manager	14561.500000

3 rows in set (0.00 sec)

6. List the properties that are handled by staff who work in the branch at '163 Main St'.

```
SELECT propertyNo, street, city, postcode, rooms, rent
FROM PropertyForRent
WHERE staffNo IN (SELECT staffNo
                  FROM Staff
                  WHERE branchNo = (SELECT branchNo
                                    FROM Branch
                                    WHERE street = '163 Main St'));
```

propertyNo	street	city	postcode	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	4	472.50
PG21	18 Dale Rd	Glasgow	G12	5	630.00
PG36	2 Manor Rd	Glasgow	G32 4QX	3	393.75

3 rows in set (0.09 sec)

7. Find all staff whose salary is larger than the salary of at least one member of staff at branch B003. (Hint: use of keyword SOME)

```
SELECT StaffNo, fName, lName, Position, Salary
FROM Staff
WHERE salary > SOME(SELECT salary
            FROM Staff
            WHERE branchNo = 'B003');
```

StaffNo	fName	lName	Position	Salary
SG14	David	Ford	Manager	18000.00
SG5	Susan	Brand	Manager	25956.00
SL21	John	White	Manager	32445.00

3 rows in set (0.10 sec)

8. Find all staff whose salary is larger than the salary of every member of staff at branch B003.

```
SELECT StaffNo, fName, lName, Position, Salary
FROM Staff
WHERE salary > ALL(SELECT salary
              FROM Staff
              WHERE branchNo = 'B003');
```

StaffNo	fName	lName	Position	Salary
SL21	John	White	Manager	32445.00

1 row in set (0.00 sec)

CONCLUSION:

Successfully implemented set operators and nested queries

Exp. No. 9 Date: 11/02/2022

Views, DCL and TCL

AIM:

Implementation of views, practise DCL commands and practice TCL commands

THEORETICAL BACKGROUND:

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

TCL (Transaction Control Language):

COMMIT: Commit command is used to permanently save any transaction into the database.
ROLLBACK: This command restores the database to the last committed state. It is also used with the savepoint command to jump to a savepoint in a transaction.
DCL is the abstract of Data Control Language. DCL includes commands such as GRANT and is concerned with rights, permissions, and other controls of the database system. DCL is used to grant/revoke permissions on databases and their contents. DCL is simple, but MySQL permissions are a bit complex.

QUERIES:

Part 1: Views

1. Create a view so that the manager at branch B003 can see only the details for staff who work in his or her branch office.

create view StaffB003 as select * from Staff s where s.branchNo='B003';

```
mysql> select * from ManagerStaff;
+---------+------+-----------+-----------+----------+----------+-------+-------+
| StaffNo | Sex  | DOB       | Position  | Salary   | branchNo | fName | lName |
+---------+------+-----------+-----------+----------+----------+-------+-------+
| SG14    | M    | 24-Mar-58 | Manager   | 18000.00 | B003     | David | Ford  |
| SG37    | F    | 10-Nov-60 | Assistant | 12360.00 | B003     | Ann   | Beech |
| SG5     | F    | 3-Jun-40  | Manager   | 25956.00 | B003     | Susan | Brand |
+---------+------+-----------+-----------+----------+----------+-------+-------+
3 rows in set (0.24 sec)
```

2. Create a view of the staff details at branch B003 that excludes salary information, so that only managers can access the salary details for staff who work at their branch.

create view ManagerStaffEXSal as select s.StaffNo, s.fname, s.lname, s.Position, s.Sex, s.DOB, s.branchNo from Staff s where s.branchNo='B003';

```
mysql> select * from ManagerStaffEXSal;
+---------+-------+-------+-----------+------+-----------+----------+
| StaffNo | fname | lname | Position  | Sex  | DOB       | branchNo |
+---------+-------+-------+-----------+------+-----------+----------+
| SG14    | David | Ford  | Manager   | M    | 24-Mar-58 | B003     |
| SG37    | Ann   | Beech | Assistant | F    | 10-Nov-60 | B003     |
| SG5     | Susan | Brand | Manager   | F    | 3-Jun-40  | B003     |
+---------+-------+-------+-----------+------+-----------+----------+
3 rows in set (0.00 sec)
```

3. Create a view of staff who manages properties for rent, which includes the branch number they work at, their staff number, and the number of properties they manage.

create view propManage as select s.branchNo, s.StaffNo, count(*) from Staff s, PropertyForRent p where s.StaffNo=p.StaffNo group by s.branchNo,s.StaffNo;

```
mysql> select * from propManage;
+----------+---------+----------+
| branchNo | StaffNo | count(*) |
+----------+---------+----------+
| B007     | SA9     |        1 |
| B003     | SG14    |        1 |
| B003     | SG37    |        2 |
| B005     | SL41    |        1 |
+----------+---------+----------+
4 rows in set (0.09 sec)
```

Part 2: DCL

1. Create a new MySQL user with a username and password using ‘CREATE USER’ command.

create user 'user1'@'localhost' identified by 'password';

```
mysql> select user from mysql.user;
+------------------+
| user             |
+------------------+
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
| user1            |
+------------------+
5 rows in set (0.00 sec)
```

2. Grant the new user all privileges on ‘Branch’ table of ‘DreamHome’ schema.

grant all on Branch to 'user1'@'localhost';

3. Grant the new user read-only privileges on ‘PropertyForRent’ table of ‘DreamHome’ schema.

grant select on DreamHome63.PropertyForRent to 'user1'@'localhost';

```
mysql> show grants for 'user1'@'localhost';
+-----------------------------------------------------------------------+
| Grants for user1@localhost                                            |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `user1`@`localhost`                             |
| GRANT ALL PRIVILEGES ON `dreamhome63`.`branch` TO `user1`@`localhost` |
| GRANT SELECT ON `dreamhome63`.`propertyforrent` TO `user1`@`localhost` |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

4. Validate the privilege assignments with proper queries as the new user.

Alter table Branch rename column postcode to Postcode;

```
+----------+--------------+----------+----------+
| branchNo | street       | city     | Postcode |
+----------+--------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU |
| B003     | 163 Main St  | Glasgow  | G11 9QX  |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ |
| B005     | 22 Deer Rd   | London   | SW1 4EH  |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  |
+----------+--------------+----------+----------+
5 rows in set (0.02 sec)
```

insert into Branch values ('B006','50 Ladykirk Rd','Newcastle','NE4 8AH');

```
+----------+----------------+-----------+----------+
| branchNo | street         | city      | Postcode |
+----------+----------------+-----------+----------+
| B002     | 56 Clover Dr   | London    | NW10 6EU |
| B003     | 163 Main St    | Glasgow   | G11 9QX  |
| B004     | 32 Manse Rd    | Bristol   | BS99 1NZ |
| B005     | 22 Deer Rd     | London    | SW1 4EH  |
| B006     | 50 Ladykirk Rd | Newcastle | NE4 8AH  |
| B007     | 16 Argyll St   | Aberdeen  | AB2 3SU  |
+----------+----------------+-----------+----------+
6 rows in set (0.00 sec)
```

delete from Branch where branchNo='B006';

```
+----------+--------------+----------+----------+
| branchNo | street       | city     | Postcode |
+----------+--------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU |
| B003     | 163 Main St  | Glasgow  | G11 9QX  |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ |
| B005     | 22 Deer Rd   | London   | SW1 4EH  |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  |
+----------+--------------+----------+----------+
5 rows in set (0.00 sec)
```

```
mysql> delete from PropertyForRent where branchNo='B003';
ERROR 1142 (42000): DELETE command denied to user 'user1'@'localhost' for table 'propertyforrent'
```

select propertyNo,city from PropertyForRent_63;

```
+------------+----------+
| propertyNo | city     |
+------------+----------+
| PA14       | Aberdeen |
| PG16       | Glasgow  |
| PG21       | Glasgow  |
| PG36       | Glasgow  |
| PG4        | Glasgow  |
| PL94       | London   |
+------------+----------+
6 rows in set (0.00 sec)
```

5. Revoke all privileges given to the new user and validate the same.

revoke all on Branch from 'user1'@'localhost';

revoke select on PropertyForRent from 'user1'@'localhost';

```
mysql> show grants for 'user1'@'localhost';
+-----------------------------------------+
| Grants for user1@localhost              |
+-----------------------------------------+
| GRANT USAGE ON *.* TO `user1`@`localhost` |
+-----------------------------------------+
1 row in set (0.00 sec)
```

Part 3: TCL

1. Create a table named 'TEMP' with attributes A1(int), A2(varchar), and A3(int).

create table TEMP (A1 int, A2 varchar(50), A3 int);

```
mysql> desc TEMP;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| A1    | int         | YES  |     | NULL    |       |
| A2    | varchar(50) | YES  |     | NULL    |       |
| A3    | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

2. Insert 10 rows into TEMP. While inserting, keep three savepoints (A, B and C) after 2nd, 5th, and 8th row insertion respectively.

START TRANSACTION;
insert into TEMP values(1,"1",1);
insert into TEMP values(2,"2",2);
SAVEPOINT A;

insert into TEMP values(3,"3",3);
insert into TEMP values(4,"4",4);
insert into TEMP values(5,"5",5);
SAVEPOINT B;

insert into TEMP values(6,"6",6);
insert into TEMP values(7,"7",7);
insert into TEMP values(8,"8",8);
SAVEPOINT C;

insert into TEMP values(9,"9",9);
insert into TEMP values(10,"10",10);

```
mysql> select * from TEMP;
+------+------+------+
| A1   | A2   | A3   |
+------+------+------+
|    1 | 1    |    1 |
|    2 | 2    |    2 |
|    3 | 3    |    3 |
|    4 | 4    |    4 |
|    5 | 5    |    5 |
|    6 | 6    |    6 |
|    7 | 7    |    7 |
|    8 | 8    |    8 |
|    9 | 9    |    9 |
|   10 | 10   |   10 |
+------+------+------+
10 rows in set (0.01 sec)
```

3. Rollback to C, B and then A printing the table contents all the time.

```
mysql> rollback to C;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from TEMP;
+------+------+------+
| A1   | A2   | A3   |
+------+------+------+
|    1 | 1    |    1 |
|    2 | 2    |    2 |
|    3 | 3    |    3 |
|    4 | 4    |    4 |
|    5 | 5    |    5 |
|    6 | 6    |    6 |
|    7 | 7    |    7 |
|    8 | 8    |    8 |
+------+------+------+
8 rows in set (0.00 sec)
```

```
mysql> rollback to B;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from TEMP;
+------+------+------+
| A1   | A2   | A3   |
+------+------+------+
|    1 | 1    |    1 |
|    2 | 2    |    2 |
|    3 | 3    |    3 |
|    4 | 4    |    4 |
|    5 | 5    |    5 |
+------+------+------+
5 rows in set (0.00 sec)
```

```
mysql> rollback to A;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from TEMP;
+------+------+------+
| A1   | A2   | A3   |
+------+------+------+
|    1 | 1    |    1 |
|    2 | 2    |    2 |
+------+------+------+
2 rows in set (0.00 sec)
```

4. Commit the current state of TEMP.

```
mysql> commit;
Query OK, 0 rows affected (0.05 sec)
```

CONCLUSION:

Successfully implemented views, DCL and TCL commands

Exp. No. 10 Date: 21/01/2022

Built-in functions in MySQL

AIM:

Familiarize various Built-in functions available in MySQL.

THEORETICAL BACKGROUND:

MySQL Built-in functions

Built in functions are functions that are shipped with MySQL. They are MySQL Aggregate Functions, MySQL Comparison Functions, MySQL Control Flow Functions and Expressions, MySQL Date Functions, MySQL String Functions, MySQL Window Functions, MySQL Math Functions.

QUERIES:

I. STRING FUNCTIONS

a) CONCAT: The CONCAT() function adds two or more expressions together.
Syntax: CONCAT(expressional, expression2, expression3, ...)
Eg: SELECT CONCAT('Hello',' ','World') AS MSG;

b) CHAR_LENGTH: It returns the length of a string in characters.
Syntax: CHAR_LENGTH(String)
Eg: SELECT CHAR_LENGTH('Hello World') AS STRLEN;

c) FORMAT: It formats a nnumber to a format like "#, ##, ###.##", rounded to a specified number of decimal places, then it returns the result as a string.
Syntax: FORMAT(number,decimal_places)
Eg: SELECT FORMAT(250200.5634,2) AS FRMTD;

d) REVERSE: Reverses a string and returns the result.
Syntax: REVERSE(String)
Eg: SELECT REVERSE('HELLO WORLD') AS REVSTR;

e) SUBSTRING: Extract a substring from a string (Starting at any position).
Syntax: SUBSTRING(String,Start,Length)
Eg: SELECT SUBSTRING('Hello World',6,5)AS SUBSTR;

II. NUMERIC FUNCTIONS

a) ABS: Returns the absolute value of a number.
Syntax: ABS(number)
Eg: SELECT ABS(-23.75);

b) CEIL: Returns the smallest integer greater then or equal to the given number.
Syntax: CEIL(number)
Eg: SELECT CEIL(25.75);

c) COS: Returns the cosine of a number.
Syntax: COS(number)
Eg: SELECT COS(2);

d) DEGREES: Converts a value in radians to degrees.
Syntax: DEGREES(number)
Eg: SELECT DEGREES(1.5);

e) POW: returns value of a number raised to the power of another number.
Syntax: POW(x,y)
Eg: SELECT POW(2,4);

III. DATE FUNCTIONS

a) ADDDATE: Adds a time/date internals to a date and then returns the date.
Syntax: ADDDATE(date, INTERVAL value addunit)
Eg: SELECT ADDDATE('2021-01-22',INTERVAL 10 DAY) AS NEWDAY;

b) CURRENT_DATE: Returns the current date in format:
Syntax: CURRENT_DATE()
Eg: SELECT CURRENT_DATE();

c) DATEDIFF: Returns the numbers of days between two dates.
Syntax: DATEDIFF(date1,date2)
Eg: SELECT DATEDIFF("2017-02-25","2017-01-10") AS DIFF;

d) DAYNAME: Returns the weekday name for a date.
Syntax: DAYNAME(date)
Eg: SELECT DAYNAME("2022-01-22");

e) NOW: Returns the current date add time
Syntax: NOW()
Eg: SELECT NOW();

IV. ADVANCED FUNCTIONS

a) BIN: Returns binary representation of a number
Syntax: BIN(number)
Eg: SELECT BIN(15);

b) CONV: Converts a number from one numeric base system to another, and returns the result as a string value.
Syntax: CONV(number,from_base.to_base)
Eg: SELECT CONV(15,10,16);

c) CURRENT_USER: Returns username and hostname for the MySQL account that the server used to authenticate current client.
Syntax: CURRENT_USER()
Eg: SELECT CURRENT_USER();

d) DATABASE: Returns the name of the current database.
Syntax: DATABASE()
Eg: SELECT DATABASE();

e) VERSION: Returns the current version of mutual database,as a string.
Syntax: VERSION();
Eg: SELECT VERSION();

RESULT:

```
mysql> SELECT CONCAT('Hello',' ','World') AS MSG;
+-------------+
| MSG         |
+-------------+
| Hello World |
+-------------+
1 row in set (0.33 sec)

mysql> SELECT CHAR_LENGTH('Hello World') AS STRLEN;
+--------+
| STRLEN |
+--------+
|     11 |
+--------+
1 row in set (0.01 sec)

mysql> SELECT FORMAT(250200.5634,2) AS FRMTD;
+------------+
| FRMTD      |
+------------+
| 250,200.56 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT REVERSE('HELLO WORLD') AS REVSTR;
+-------------+
| REVSTR      |
+-------------+
| DLROW OLLEH |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT SUBSTRING('Hello World',6,5)AS SUBSTR;
+--------+
| SUBSTR |
+--------+
|  Worl  |
+--------+
1 row in set (0.01 sec)
```

```
mysql> SELECT ABS(-23.75);
+-------------+
| ABS(-23.75) |
+-------------+
|       23.75 |
+-------------+
1 row in set (0.04 sec)

mysql> SELECT CEIL(25.75);
+-------------+
| CEIL(25.75) |
+-------------+
|          26 |
+-------------+
1 row in set (0.03 sec)

mysql> SELECT COS(2);
+---------------------+
| COS(2)              |
+---------------------+
| -0.4161468365471424 |
+---------------------+
1 row in set (0.03 sec)

mysql> SELECT DEGREES(1.5);
+-------------------+
| DEGREES(1.5)      |
+-------------------+
| 85.94366926962348 |
+-------------------+
1 row in set (0.01 sec)

mysql> SELECT POW(2,4);
+----------+
| POW(2,4) |
+----------+
|       16 |
+----------+
1 row in set (0.09 sec)
```

```
mysql> SELECT ADDDATE('2021-01-22',INTERVAL 10 DAY) AS NEWDAY;
+------------+
| NEWDAY     |
+------------+
| 2021-02-01 |
+------------+
1 row in set (0.10 sec)

mysql> SELECT CURRENT_DATE();
+----------------+
| CURRENT_DATE() |
+----------------+
| 2022-04-10     |
+----------------+
1 row in set (0.00 sec)

mysql> SELECT DATEDIFF("2017-02-25","2017-01-10") AS DIFF;
+------+
| DIFF |
+------+
|   46 |
+------+
1 row in set (0.01 sec)

mysql> SELECT DAYNAME("2022-01-22");
+-----------------------+
| DAYNAME("2022-01-22") |
+-----------------------+
| Saturday              |
+-----------------------+
1 row in set (0.03 sec)

mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-04-10 00:35:56 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT BIN(15);
+---------+
| BIN(15) |
+---------+
| 1111    |
+---------+
1 row in set (0.08 sec)

mysql> SELECT CONV(15,10,16);
+----------------+
| CONV(15,10,16) |
+----------------+
| F              |
+----------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_USER();
+----------------+
| CURRENT_USER() |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)

mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| NULL       |
+------------+
1 row in set (0.00 sec)

mysql> SELECT VERSION();
+-----------+
| VERSION() |
+-----------+
| 8.0.27    |
+-----------+
1 row in set (0.00 sec)
```

CONCLUSION:

Successfully familiarized with various Built-in functions available in MySQL

Exp. No. 11 Date: 28/01/2022

MySQL Stored Procedure Programming I

AIM:

Practice the use of variables, conditional structures and loops with MySQL stored procedures.

THEORETICAL BACKGROUND:

Variables are used for storing data or information during the execution of a program. It is a way of labelling data with an appropriate name that helps to understand the program more clearly by the reader. The main purpose of the variable is to store data in memory and can be used throughout the program.

MySQL simple IF-THEN statement

If the condition evaluates to TRUE , the statements between IF-THEN and END IF will execute. Otherwise, the control is passed to the next statement following the END IF . Second, specify the code that will execute if the condition evaluates to TRUE .

The MySQL LOOP statement could be used to run a block of code or set of statements, again and again, depending on the condition.

QUERIES:

1.
a) Write a stored procedure to accept values of a, b and c and display which one is greatest.

```
delimiter $$
drop PROCEDURE IF EXISTS max_3$$
CREATE PROCEDURE max_3(a int,b int,c int)
BEGIN
   IF a>b AND a>c THEN
      SELECT a AS "Greatest number";
   ELSEIF b>a AND b>c THEN
      SELECT b AS "Greatest number";
   ELSE
      SELECT c AS "Greatest Number";
   END IF;
END$$
delimiter ;
```

```
mysql> call max_3(1,4,2);
+-----------------+
| Greatest number |
+-----------------+
|               4 |
+-----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

b) Write a stored procedure to accept values of a, b and c and display which one is greatest with the use of ‘in’ and ‘out’ parameters.

```
delimiter $$
drop PROCEDURE IF EXISTS max_3B$$
CREATE PROCEDURE max_3B(IN a int,IN b int,IN c int, OUT d int)
BEGIN
   IF a>b AND a>c THEN
      SET d = a;
   ELSEIF b>a AND b>c THEN
      SET d = b;
   ELSE
      SET d = c;
   END IF;
END$$
delimiter ;
```

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

mysql> call max_3B(1,4,2,@a);
Query OK, 0 rows affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

c) Write a stored procedure to accept values of a, b and c and display which one is greatest using ‘inout’ parameter

```
delimiter $$
drop PROCEDURE IF EXISTS max_3c$$
CREATE PROCEDURE max_3c(IN a int,IN b int,INOUT c int)
BEGIN
   IF a>b AND a>c THEN
      SET c = a;
   ELSEIF b>a AND b>c THEN
      SET c = b;
   ELSE
      SET c = c;
   END IF;
END$$
delimiter ;
```

```
mysql> call max_3c(1,4,@c);
Query OK, 0 rows affected (0.00 sec)

mysql> select @c;
+------+
| @c   |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

2. Pass a student mark for one subject as input to a procedure displaying the grade (A to F). (Do using CASE statement)

```
delimiter $$
drop PROCEDURE IF EXISTS grade$$
CREATE PROCEDURE grade(IN mark INT)
BEGIN
        DECLARE grade varchar(2);
        CASE
           WHEN mark >= 90 THEN SET grade = "S+";
           WHEN mark>=80 AND mark<90 THEN SET grade = "A";
           WHEN mark>=70 AND mark<80 THEN SET grade = "B";
           WHEN mark>=60 AND mark<70 THEN SET grade = "C";
           WHEN mark>=50 AND mark<60 THEN SET grade = "D";
           WHEN mark>=40 AND mark<50 THEN SET grade = "E";
           ELSE SET grade = "F";
        END CASE;
        SELECT mark, grade;
        END$$
delimiter ;
```

```
mysql>
mysql> call grade(70);
+------+-------+
| mark | grade |
+------+-------+
|   70 | B     |
+------+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

3. Write a stored procedure to find the sum of ‘n’ natural numbers. (Do using LOOP-END LOOP)

```
delimiter $$
drop PROCEDURE IF EXISTS sum_n$$
CREATE PROCEDURE sum_n(IN n INT)
BEGIN
   DECLARE sm INT;
```

```
   DECLARE i INT;
   SET sm = 0;SET i =1;
   sm_count:LOOP
      SET sm = sm+i;
      IF i = n THEN
         LEAVE sm_count;
      END IF;
      SET i = i+1;
   END LOOP sm_count;
   SELECT n, sm AS "Sum of integer upto n";
END$$
delimiter ;
```

```
mysql> call sum_n(10);
+------+-----------------------+
| n    | Sum of integer upto n |
+------+-----------------------+
|   10 |                    55 |
+------+-----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

4. Write a stored procedure to find the sum of odd numbers up to ‘n’. (Do using REPEAT – UNTIL)

```
delimiter $$
drop PROCEDURE IF EXISTS sum_odd$$
CREATE PROCEDURE sum_odd(IN n INT)
BEGIN
   DECLARE sm, i INT;
   SET sm = 0;SET i =1;
   REPEAT
      SET sm = sm+i;
      SET i = i+2;
      UNTIL i > n
   END REPEAT;
   SELECT n, sm AS "Sum of odd integer upto n";
END$$
delimiter ;
```

```
mysql> call sum_odd(10);
+------+---------------------------+
| n    | Sum of odd integer upto n |
+------+---------------------------+
|   10 |                        25 |
+------+---------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

5. Write a stored procedure to find the sum of even numbers up to ‘n’. (Do using WHILE).

```
delimiter $$
drop PROCEDURE IF EXISTS sum_even$$
CREATE PROCEDURE sum_even(IN n INT)
BEGIN
        DECLARE sm INT;
   DECLARE i INT;
   SET sm = 0;SET i =0;
   sm_even: WHILE
      i<=n DO
         SET sm = sm+i;
         SET i = i+2;
   END WHILE sm_even;
   SELECT n, sm AS "Sum of odd integer upto n";
END$$
delimiter ;
```

```
mysql> call sum_even(10);
+------+---------------------------+
| n    | Sum of even integer upto n |
+------+---------------------------+
|   10 |                        30 |
+------+---------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

6. Write a stored procedure to find the number of digits in an input integer.

```
delimiter $$
drop PROCEDURE IF EXISTS digitno$$
CREATE PROCEDURE digitno(IN n INT)
BEGIN
   DECLARE sm INT;DECLARE i INT;
   SET sm = 0;
   SET i = n;
   digitn: WHILE
      n>0 DO
         SET sm = sm+1;
         SET n = n/10;
   END WHILE digitn;
   SELECT i, sm AS "Number of digits in n";
END$$
delimiter ;
```

```
mysql> call digitno(1234);
+------+----------------------+
| i    | Number of digits in n |
+------+----------------------+
| 1234 |                    4 |
+------+----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

7. Accept two values ‘a’ and ‘b’ and swap them within a procedure using ‘inout’ parameter

```
delimiter $$
drop PROCEDURE IF EXISTS swap$$
CREATE PROCEDURE swap(INOUT a INT, INOUT b INT)
BEGIN
   DECLARE temp INT DEFAULT a;
   SET a = b;
   SET b = temp;
END$$
delimiter ;
```

```
mysql> set @a=5;
Query OK, 0 rows affected (0.00 sec)

mysql> set @b=1;
Query OK, 0 rows affected (0.00 sec)

mysql> call swap(@a,@b);
Query OK, 0 rows affected (0.00 sec)

mysql> select @a,@b;
+------+------+
| @a   | @b   |
+------+------+
|    1 |    5 |
+------+------+
1 row in set (0.00 sec)
```

CONCLUSION:

Successfully practised use of variables, conditional structures and loops with MySQL stored procedures

Exp. No. 12 Date: 03/02/2022

MySQL Stored Procedure Programming II

AIM:

Practice the use of Non-SELECT SQL statements and SELECT-INTO clauses within stored procedures.

THEORETICAL BACKGROUND:

The term non-SELECT statement refers to any SQL statement that can be prepared, except SELECT and EXECUTE FUNCTION. This term includes the EXECUTE PROCEDURE statement. The INSERT statement is an exception to the rules for non-SELECT statements.

QUERIES:

1. Create a table temp with two fields, **TEMP01(num:INTEGER, message TEXT)** Insert values into this table using a stored procedure such that the num field has values from 1 to 10 and the corresponding message is either even or odd.

```
drop table if exists TEMP01;
create table TEMP01(num int, message varchar(10));
DELIMITER $$
drop procedure if exists oddEven $$
create procedure oddEven()
begin
	declare i int default 1;
	while i<=10 do
		if i%2=0 then
			insert into TEMP01 values(i,"even");
		else
			insert into TEMP01 values(i,"odd");
		end if;
		set i=i+1;
	end while;
end $$
delimiter ;
call oddEven();
```

```
select * from TEMP01;
```

```
+------+---------+
| num  | message |
+------+---------+
|    1 | odd     |
|    2 | even    |
|    3 | odd     |
|    4 | even    |
|    5 | odd     |
|    6 | even    |
|    7 | odd     |
|    8 | even    |
|    9 | odd     |
|   10 | even    |
+------+---------+
10 rows in set (0.00 sec)
```

2. Create an employee table and insert 5 rows. Write a procedure to calculate income tax of a specified employee. [Give the employee SSN as input parameter]
Employee(SSN,Name,Designation,Basic_pay,DA,HRA,Gender,Years_of_exp) Note: You can create and insert values outside the procedure as usual. Insert meaningful values to all fields and use original way of calculating tax for a person

```
drop table if exists Employee;
create table Employee(SSN int primary key, Name varchar(20), Designation varchar(20), Basic_pay int,
DA int, HRA int, Gender char, Years_of_exp int);
INSERT INTO Employee(SSN, Name, Designation, Basic_pay, DA, HRA, Gender, Years_of_exp)
VALUES
        (111, "Kumar", "Manager", 32000, 16000, 1400, 'M', 10),
        (222, "Umar", "Account", 14000, 7000, 1200, 'M', 7),
        (333, "Vino", "Clerk", 10000, 5000, 800, 'M', 4),
        (444, "Raj", "Manager", 30000, 15000, 1400, 'M', 9),
        (555, "Omna", "As.manager", 24000, 12000, 1200, 'F', 8);

DELIMITER $$
drop procedure if exists getIncomeTax $$
create procedure getIncomeTax(in_SSN int)
begin
        declare aSSN, aDA, aBP, aHRA, tax, income int;
        declare aName varchar(20);
        SELECT SSN, Name, Basic_pay, DA, HRA
        INTO aSSN, aName, aBP, aDA, aHRA
        FROM Employee
        WHERE SSN=in_SSN;
        SET income=(aBP+aDA+aHRA)*12;
        CASE
                WHEN income <=250000 THEN SET tax=0;
                WHEN income <=500000 THEN SET tax=income*0.05;
                WHEN income <=750000 THEN SET tax=income*0.10;
                WHEN income <=1000000 THEN SET tax=income*0.15;
                WHEN income <=1250000 THEN SET tax=income*0.20;
                WHEN income <=1500000 THEN SET tax=income*0.25;
                ELSE SET tax=income*0.30;
        END CASE;
        SELECT aSSN as SSN, aName as Name, tax as Income_Tax;
end $$
delimiter ;

call getIncomeTax(222);
```

```
+------+------+------------+
| SSN  | Name | Income_Tax |
+------+------+------------+
|  222 | Umar |      13320 |
+------+------+------------+
1 row in set (0.09 sec)
```

3. Write a procedure to Display Salary of a specified employee (as input argument) increased by 500 if his/her salary is more than 30000. [Use above table]

```
DELIMITER $$
drop procedure if exists salary $$
create procedure salary(in_SSN int)
begin
	declare aSSN, aBP int;
	declare aName varchar(20);

	SELECT SSN, Name, Basic_pay
	INTO aSSN, aName, aBP
	FROM Employee
	WHERE SSN=in_SSN;

	if aBP>30000 then
		SET aBP=aBP+500;
	end if;

	SELECT aSSN as SSN, aName as Name, aBP as Salary;
end $$
delimiter ;

call salary(111);
```

```
+------+-------+--------+
| SSN  | Name  | Salary |
+------+-------+--------+
|  111 | Kumar |  32500 |
+------+-------+--------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.03 sec)
```

4. Create a procedure to calculate the bonus of an employee whose SSN is given as input, based on experience and store it into the bonus table: **Bonus(SSN, Name, Bonus)** If exp < 5 years then bonus is 1 month salary If exp between 5 and 9 years then bonus is 20% of annual salary If exp more than 9 years then bonus is 1 month salary plus 25% of annual salary.

```
drop table if exists Bonus;
CREATE TABLE Bonus(SSN int PRIMARY KEY, Name varchar(20), Bonus int);

DELIMITER $$
drop procedure if exists bonus $$
create procedure bonus(in_SSN int)
begin
	declare y,bp,essn, Bonus int;
	declare ename varchar(20);
```

```
        select SSN, Years_of_exp, Basic_pay, Name
        INTO essn, y, bp, ename
        from Employee
        where SSN=in_SSN;

        if y<5 then
                SET Bonus=bp;
        elseif y<=9 then
                SET Bonus=bp*2.4;
        else
                SET Bonus=bp*4;
        end if;

        insert into Bonus values(essn,ename,Bonus);
end $$
delimiter ;

call bonus(111);
call bonus(222);
call bonus(333);
call bonus(444);
call bonus(555);

select * from Bonus;
```

```
+-----+-------+--------+
| SSN | Name  | Bonus  |
+-----+-------+--------+
| 111 | Kumar | 128000 |
| 222 | Umar  |  33600 |
| 333 | Vino  |  10000 |
| 444 | Raj   |  72000 |
| 555 | Omna  |  57600 |
+-----+-------+--------+
5 rows in set (0.00 sec)
```

5. Create a table **account_master (acct_no :int, customer_name: text, balance:decimal)**. Write a stored procedure to accept the account number and the amount to withdraw. Do proper updation on the table only if there is sufficient amount, otherwise display proper message.

```
drop table if exists account_master;
CREATE TABLE account_master(acct_no int PRIMARY KEY, customer_name varchar(20), balance
decimal(10,2));
INSERT INTO account_master(acct_no, customer_name, balance) VALUES
        (4515,"Peter Parker",45000),
        (6789,"Robin Pinto",100450),
        (9301,"Mary Tim",12000),
        (1212,"Thomas Jefferson",32000),
        (6712,"Jack",550);
```

```
DELIMITER $$
drop procedure if exists account $$

create procedure account(in_acc int, amo int)
begin
	declare bal decimal(10,2);

	select balance into bal
	from account_master
	where acct_no=in_acc;

	if bal>amo then
		UPDATE account_master
		SET balance=balance-amo
		WHERE acct_no=in_acc;
	else
		select "Unable to complete the withdrawal. Please check your balance and try again." as
ERROR;
	end if;
end $$
delimiter ;

call account(6712,45000);
call account(9301,200);
select * from account_master;
```

```
+-------------------------------------------------------------------------------+
| ERROR                                                                         |
+-------------------------------------------------------------------------------+
| Unable to complete the withdrawal. Please check your balance and try again. |
+-------------------------------------------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.05 sec)

Query OK, 1 row affected (0.35 sec)

+---------+------------------+-----------+
| acct_no | customer_name    | balance   |
+---------+------------------+-----------+
|    1212 | Thomas Jefferson |  32000.00 |
|    4515 | Peter Parker     |  45000.00 |
|    6712 | Jack             |    550.00 |
|    6789 | Robin Pinto      | 100450.00 |
|    9301 | Mary Tim         |  11800.00 |
+---------+------------------+-----------+
5 rows in set (0.00 sec)
```

CONCLUSION:

Successfully practised the use of Non-SELECT SQL statements and SELECT-INTO clause within stored procedures

Exp. No. 13 Date: 09/02/2022

MySQL Stored Procedure Programming III

AIM:

Practice the creation of cursors, functions and triggers in MySQL.

THEORETICAL BACKGROUND:

In MySQL, a cursor allows row-by-row processing of the result sets. A cursor is used for the result set and returned from a query. By using a cursor, you can iterate, or step through the results of a query and perform certain operations on each row.

In MySQL, a function is a stored program that you can pass parameters into and then return a value.

A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. Some uses for triggers are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update.

QUERIES:

1. Write a stored procedure using a cursor to calculate the total and the percentage of marks of the students in four subjects from the table - Student with the schema given below.
STUDENT (RNO , S1 , S2, S3, S4, total, percentage)
[Initially table is partially filled except the last two columns. Those columns should be updated from your procedure].

```
drop table if exists STUDENT;
create table STUDENT(RNO int primary key, S1 int, S2 int, S3 int, S4 int, total int, percentage
decimal(5,2));
insert into STUDENT(RNO, S1, S2, S3, S4) values
        (1, 95, 90, 98, 92),
        (2, 97, 88, 94, 96),
        (3, 93, 91, 88, 94);
delimiter $$
drop procedure if exists getPercentage $$
create procedure getPercentage()
begin
        DECLARE l_last_row INT DEFAULT 0;
        declare rn int;
        declare as1 ,as2, as3, as4 int;
        declare per float default 0;
        declare tot int default 0;
        declare student cursor for
                select RNO, S1, S2, S3, S4
                        from STUDENT;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET l_last_row=1;
        open student;
        stu_cursor: loop
```

```
            fetch student into rn, as1, as2, as3, as4;
            IF (l_last_row=1) THEN
                LEAVE stu_cursor;
            END IF;
            set tot=as1+as2+as3+as4;
            set per=tot/4;
            update STUDENT
            set total=tot, percentage=per
            where RNO=rn;
            end loop stu_cursor;
        close student;
end $$
delimiter ;
call getPercentage();
select * from STUDENT;
```

RNO	S1	S2	S3	S4	total	percentage
1	95	90	98	92	375	93.75
2	97	88	94	96	375	93.75
3	93	91	88	94	366	91.50

3 rows in set (0.00 sec)

2. Create a stored function to accept a number and return its factorial.

```
delimiter $$
CREATE FUNCTION factorial (n DECIMAL(3,0))
RETURNS DECIMAL(20,0)
DETERMINISTIC
BEGIN
        DECLARE factorial DECIMAL(20,0) DEFAULT 1;
        DECLARE counter DECIMAL(3,0);
        SET counter = n;
        factorial_loop: REPEAT
            SET factorial = factorial * counter;
            SET counter = counter - 1;
            UNTIL counter = 1
        END REPEAT;
        RETURN factorial;
END $$
delimiter ;
select factorial(5);
```

factorial(5)
120

1 row in set (0.07 sec)

3. Write a stored function that accepts department number as input argument and returns the total salary of that department using the below schema:
WORK <EMP_NO, NAME, COMPANY_NAME, JOIN_DATE, DESIGNATION, SALARY, DEPT_NO>
[First create and populate the WORK table with meaningful values].

```
drop table if exists WORK;
create table WORK(EMP_NO int primary key, NAME varchar(20), COMPANY_NAME varchar(40),
JOIN_DATE varchar(20), DESIGNATION varchar(20), SALARY int, DEPT_NO int);
insert into WORK(EMP_NO, NAME, COMPANY_NAME, JOIN_DATE, DESIGNATION, SALARY,
DEPT_NO) values
        (7499, "ALLEN", "APPLE", '20-FEB-81', "SALESMAN", 1600, 30),
        (7521, "WARD", "APPLE", '22-FEB-81', "SALESMAN", 1250, 20),
        (7566, "JONES", "APPLE", '02-APR-81', "MANAGER", 2975, 20),
        (7698, "BLAKE", "APPLE", '01-MAY-81', "MANAGER", 2850, 30);

delimiter $$
DROP FUNCTION if exists totsal $$
create function totsal(depno int)
returns int
deterministic
begin
        declare total int;
        select sum(SALARY) into total
        from WORK
        where DEPT_NO=depno;
        return total;
end $$
delimiter ;

select * from work;
select totsal(30);
```

```
+--------+-------+--------------+-----------+-------------+--------+---------+
| EMP_NO | NAME  | COMPANY_NAME | JOIN_DATE | DESIGNATION | SALARY | DEPT_NO |
+--------+-------+--------------+-----------+-------------+--------+---------+
|   7499 | ALLEN | APPLE        | 20-FEB-81 | SALESMAN    |   1600 |      30 |
|   7521 | WARD  | APPLE        | 22-FEB-81 | SALESMAN    |   1250 |      20 |
|   7566 | JONES | APPLE        | 02-APR-81 | MANAGER     |   2975 |      20 |
|   7698 | BLAKE | APPLE        | 01-MAY-81 | MANAGER     |   2850 |      30 |
+--------+-------+--------------+-----------+-------------+--------+---------+
4 rows in set (0.00 sec)

+------------+
| totsal(30) |
+------------+
|       4450 |
+------------+
1 row in set (0.00 sec)
```

4. Create a trigger that converts all department names into capital letters before inserting them to the department table. (i.e. even if you give department name using lowercase letters in insert query, that should be inserted as capital letters) department(dept-num, dept_name, location, head_of_dept)

```
drop table if exists department;
create table department(dept_num int primary key, dept_name varchar(20), location varchar(20),
head_of_dept varchar(20));
delimiter $$
drop trigger if exists dep_in $$
create trigger dep_in before insert on department
for each row
begin
	set new.dept_name=UCASE(new.dept_name);
end $$
delimiter ;
insert into department(dept_num, dept_name, location, head_of_dept) values
	(10, "accounting", "NEW YORK", "Allen"),
	(20, "research", "DALLAS", "Ward"),
	(30, "sales", "CHICAGO", "Jones");
select * from department;
```

```
+----------+------------+----------+--------------+
| dept_num | dept_name  | location | head_of_dept |
+----------+------------+----------+--------------+
|       10 | ACCOUNTING | NEW YORK | Allen        |
|       20 | RESEARCH   | DALLAS   | Ward         |
|       30 | SALES      | CHICAGO  | Jones        |
+----------+------------+----------+--------------+
3 rows in set (0.00 sec)
```

5. Create a trigger in MySQL to log the changes of the employee table. All updations must be tracked and recorded in a table emp_audit_log.
Employee (emp_num, name, dob, designation)
Emp_audit_log(audit_id, emp_num, name, change_date, action)

```
drop table if exists Employee;
create table Employee (emp_num int primary key, name varchar(20), dob varchar(20), designation
varchar(20));
drop table if exists Emp_audit_log;
create table Emp_audit_log(audit_id int auto_increment primary key, emp_num int, name varchar(20),
change_date date, action varchar(20));
delimiter $$
drop trigger if exists ai_data $$
CREATE TRIGGER ai_data AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
	INSERT INTO Emp_audit_log(emp_num, name, change_date, action) VALUES
		(new.emp_num, new.name, now(), 'insert');
```

```
END $$

drop trigger if exists au_data $$
CREATE TRIGGER au_data AFTER update ON Employee
FOR EACH ROW
BEGIN
	INSERT INTO Emp_audit_log(emp_num, name, change_date, action) VALUES
		(new.emp_num, new.name, now(), 'update');
END $$
drop trigger if exists ad_data $$
CREATE TRIGGER ad_data AFTER delete ON Employee
FOR EACH ROW
BEGIN
	INSERT INTO Emp_audit_log(emp_num, name, change_date, action) VALUES
		(old.emp_num, old.name, now(), 'delete');
END $$
delimiter ;
insert into Employee(emp_num, name, dob, designation) values
	(23, "Smith","22-DEC-80", "Clerk"),
	(24, "Allen", "12-DEC-80", "Manager"),
	(25, "Ward", "02-DEC-80", "Salesman");
update Employee
set designation="Manager"
where emp_num=25;
delete from Employee
where emp_num=23;

select * from Emp_audit_log;
```

```
+----------+---------+-------+-------------+--------+
| audit_id | emp_num | name  | change_date | action |
+----------+---------+-------+-------------+--------+
|        1 |      23 | Smith | 2022-04-10  | insert |
|        2 |      24 | Allen | 2022-04-10  | insert |
|        3 |      25 | Ward  | 2022-04-10  | insert |
|        4 |      25 | Ward  | 2022-04-10  | update |
|        5 |      23 | Smith | 2022-04-10  | delete |
+----------+---------+-------+-------------+--------+
5 rows in set (0.00 sec)

mysql> select * from Employee;
+---------+-------+-----------+-------------+
| emp_num | name  | dob       | designation |
+---------+-------+-----------+-------------+
|      24 | Allen | 12-DEC-80 | Manager     |
|      25 | Ward  | 02-DEC-80 | Manager     |
+---------+-------+-----------+-------------+
2 rows in set (0.00 sec)
```

CONCLUSION:

Successfully practised creation of cursors, functions and triggers in MySQL

Exp. No. 14 Date: 17/02/2022

Familiarization of NoSQL Databases and CRUD operations

AIM:

Practice the CRUD operations in NoSQL(MongoDB).

THEORETICAL BACKGROUND:

NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables.
NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.
MongoDB is a source-available cross-platform document-oriented database program.

Insert

```
db.collection.insertOne()
db.collection.insertMany()
```

Read

```
db.collection.find()
```

Update

```
db.collection.updateOne()
db.collection.updateMany()
db.collection.replaceOne()
```

Delete

```
db.collection.deleteOne()
db.collection.deleteMany()
```

QUERIES:

Create

```
db.products.insertOne( { item: "card", qty: 15 } );
db.products.insertOne( { item: "pencil", qty: 20 } );
db.products.insertMany( [
        { item: "envelope", qty: 20 },
        { item: "stamps" , qty: 30 }
] );
```

Read

```
db.products.find();
```

Update

```
db.products.updateOne( { item: "card" },{$set:{qty:16}} );
```

Delete

```
db.products.deleteOne( { item: "card" } );
```

RESULT:

```
>>> db.products.insertOne( { item: "card", qty: 15 } );
{
  acknowledged: true,
  insertedId: ObjectId("62507c1ad2f5b4fb55c013e2")
}
```

```
>>> db.products.find();
[
  { _id: ObjectId("62507948e62f65bcaee1801e"), item: 'card', qty: 15 },
  {
    _id: ObjectId("62507b75e62f65bcaee1801f"),
    item: 'pencil',
    qty: 20
  },
  {
    _id: ObjectId("62507b75e62f65bcaee18020"),
    item: 'envelope',
    qty: 20
  },
```

```
>>> db.products.updateOne({item:"card"},{$set:{qty:16}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
>>> db.products.deleteOne({item:'card'});
{ acknowledged: true, deletedCount: 1 }
test>
```

CONCLUSION:

Successfully practised CRUD operations in NoSQL(MongoDB)

Exp. No. 15 Date: 03/03/2022

Web-based User Authentication System

AIM:

Design a database application using any front-end tool for any problem selected.

PROBLEM:

A web-based user authentication system that can be used for providing access to only authorized users to a network.

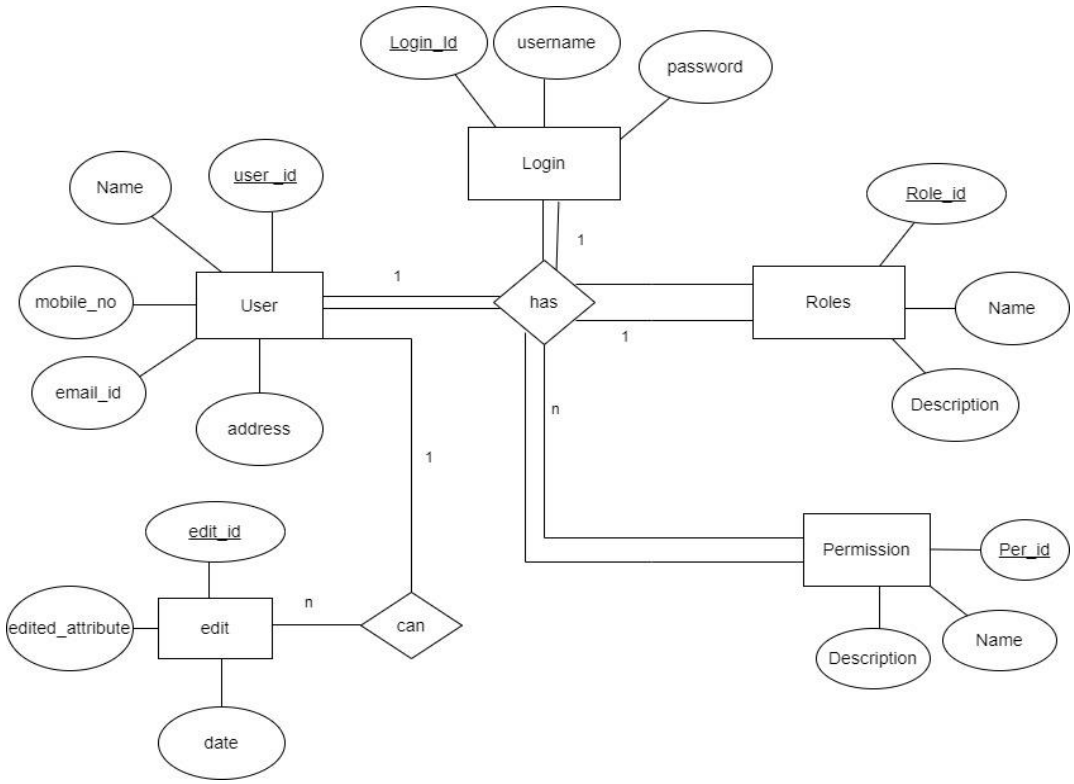
PROBLEM DESCRIPTION:

A web-based user identification system is used to verify the identity of a potential user attempting to gain access to a network or computing resource by authorising a human to machine transfer of credentials during interactions on a network to confirm a user's authenticity. It prevents unauthorised users from gaining access and potentially damaging systems, stealing information or causing other problems. User identification and authentication can be as simple as requiring a user to type a unique identifier, such as a user id, along with a password to access a system. System administrator uses these ids to assign privileges, track user activities, and manage overall operations on a particular system, network or application

TECHNOLOGIES USED:

- MySQL
- HTML5, CSS3
- JavaScript
- PHP

ER Diagram:



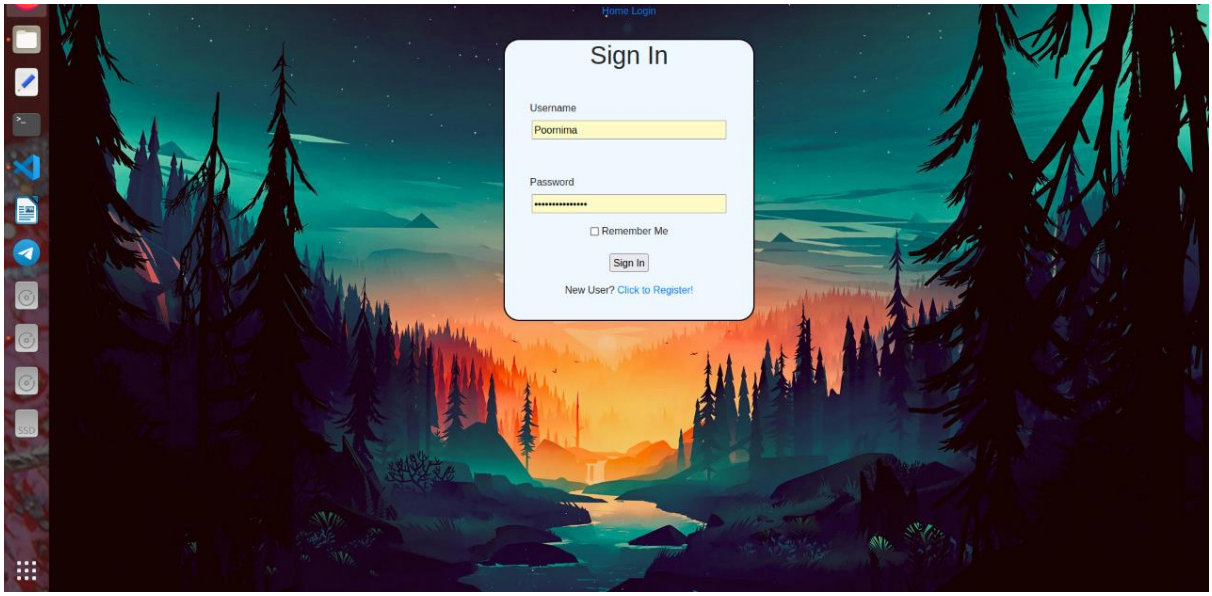
RELATIONAL SCHEMA:

```
mysql> desc User
    -> ;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| user_id   | int         | NO   | PRI | NULL    |       |
| name      | varchar(20) | YES  |     | NULL    |       |
| mobile_no | varchar(12) | YES  |     | NULL    |       |
| email_id  | varchar(20) | YES  |     | NULL    |       |
| address   | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.10 sec)

mysql> desc Login;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| login_id | int         | NO   | PRI | NULL    |       |
| username | varchar(20) | YES  |     | NULL    |       |
| password | varchar(20) | YES  |     | NULL    |       |
| user_id  | int         | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

```
mysql> desc Roles
    -> ;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| role_id     | int         | NO   | PRI | NULL    |       |
| name        | varchar(20) | YES  |     | NULL    |       |
| description | varchar(20) | YES  |     | NULL    |       |
| user_id     | int         | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc Permission;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| per_id     | int         | NO   | PRI | NULL    |       |
| name       | varchar(20) | YES  |     | NULL    |       |
| description | varchar(20) | YES  |     | NULL    |       |
| user_id    | int         | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

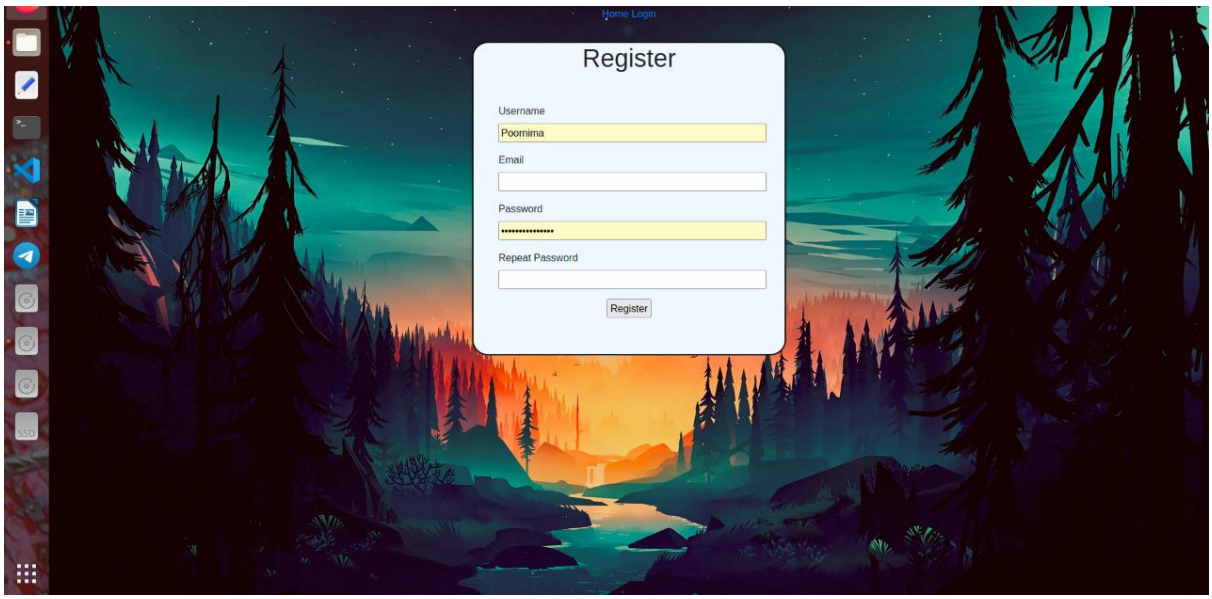
WEBSITE DEMO

Login page



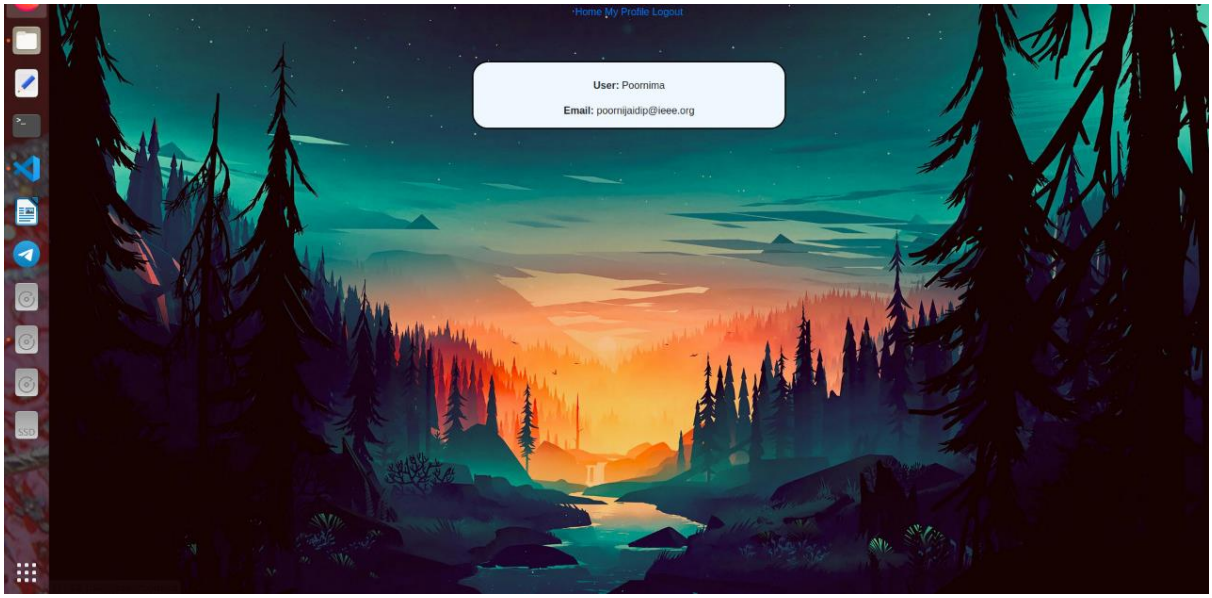
The application starts with the login page. Here, the already registered users can login and access their information or data using a unique user id and their password. The page also features a remember me option which recognizes the user's id and saves it so that he can select it for future use. For new users there is a register option which redirects the user to a registration page.

Registration page



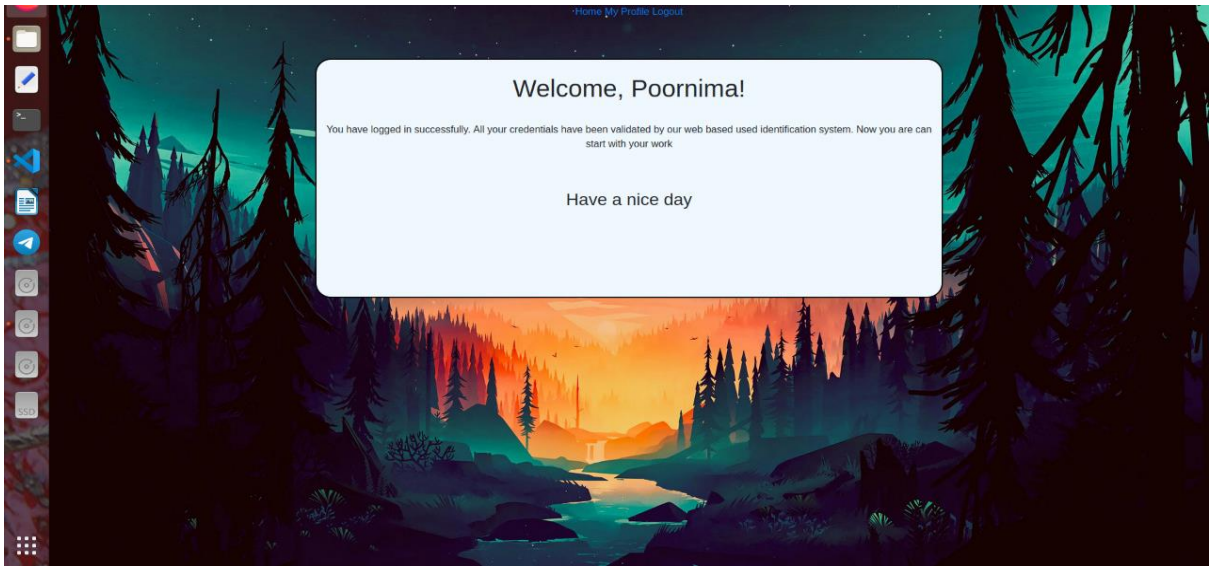
This page is designed for new users who wants to register and create an account for themselves. The page prompts them to create a unique username and a password. The user is also asked to enter his/her email id for reference. A registered user will be redirected to the login page where they can use their username and password to logon to their system.

Account details



This page is a sample preview of the user's details. The page shows the user his/her email id that was used to register along with the username. Depending upon the details used for registration the contents of the page can vary.

Welcome page



The user after logging on to their account is prompted to this welcome page. The page displays a message informing the user that his/her credentials has been verified successfully and that the user is authorized to use the account and its data as per their wish.

CONCLUSION:

The application was created successfully and its working was verified.