CSL 333 Database Management Systems Lab

LABORATORY RECORD

SUBMITTED BY

MADHURYA R NAIR

NSS20CS029

ROLL NO: 26

to

the APJ Abdul Kalam Technological University

Fifth Semester B. Tech Degree Lab Examination (2019 Scheme)

Department of Computer Science and Engineering

N.S.S. College of Engineering, Palakkad

December 2022

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: 24/09/2022**

Design a database schema for an application with ER diagram

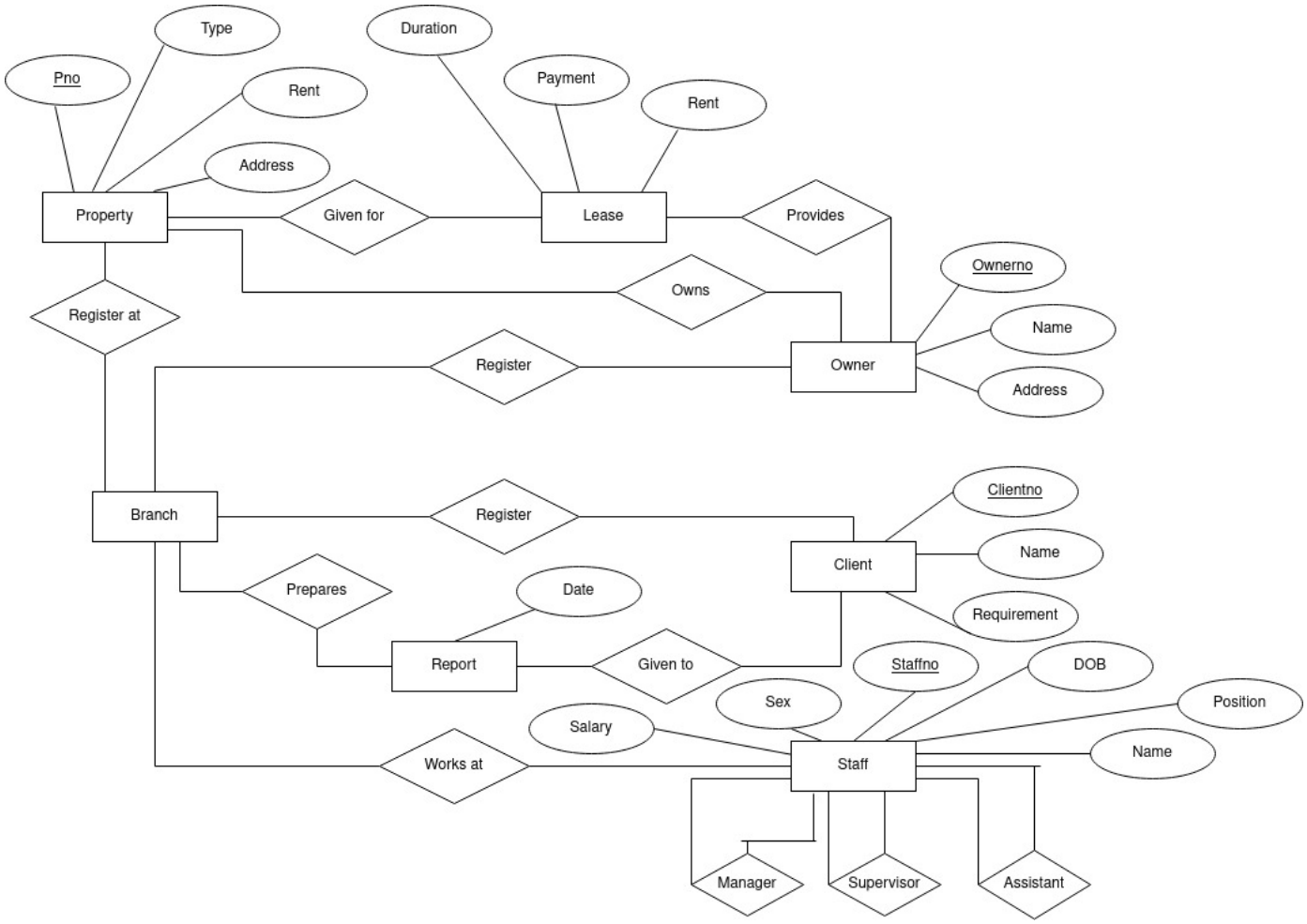
AIM:

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	SName	Sex	DOB	Position	Salary	SID

Branch

BNo	BAddress	BTele	MID

Owner

OID	OName	OAddress	OTele

Works

SID	BID

Property

PNo	Type	Rooms	Rent	PAddress	OID

Client

CNo	CName	ReqType	MaxRent	BID

Lease

PID	CID	MonthlyRent	Payment	Duration

Gen

RID	BID	RDate

UserResp

RID	CID	Date	Comment

Report

PID	RID	Type	Rent	Address

CONCLUSION:

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

Exp. No. 2 **Date: 26/09/2022**

Creation of database schema - DDL

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 dreamhome26;

use dreamhome26;

create table Staff(SNo integer NOT NULL, SName varchar(15), Sex varchar(10), DOB date, Position varchar(15), Salary varchar(7), SID integer, primary key(SNo), foreign key(SID) references Staff(SNo));

create table Branch(BNo integer NOT NULL, BAddress varchar(30), BTele varchar(11), MID integer, primary key(BNo), foreign key(MID) references Staff(SNo));

create table Owner(OID integer NOT NULL, OName varchar(15), OAddress varchar(30), OTele varchar(11), primary key(OID));

create table Works(SID integer, BID integer, primary key(SID, BID));

create table Property(PNo integer NOT NULL, Type varchar(10), Rooms integer, Rent varchar(6), PAddress varchar(30), OID integer, primary key(PNo), foreign key(OID) references Owner(OID));

create table Client(CNo integer NOT NULL, CName varchar(15), ReqType varchar(10), MaxRent varchar(6), BID integer, primary key(CNo), foreign key(BID) references Branch(BNo));

create table Lease(PID integer NOT NULL, CID integer NOT NULL, MonthlyRent varchar(6), Payment varchar(10), Duration varchar(7), primary key(PID,CID), foreign key(PID) references Property(PNo), foreign key(CID) references Client(CNo));
```

create table Gen(RID integer NOT NULL, BID integer, RDate date, primary key(RID), foreign key(BID) references Branch(BNo));

create table UserResp(RID integer, CID integer, Date date, Comment varchar(50), primary key(RID,CID), foreign key(RID) references Gen(RID), foreign key(CID) references Client(CNo));

create table Report(PID integer, RID integer, Type varchar(6), Rent varchar(6), Address varchar(30), primary key(PID,RID), foreign key(PID) references Property(PNo), foreign key(RID) references Gen(RID));

RESULT:

```
mysql> desc Staff;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| SNo      | int(11)     | NO   | PRI | NULL    |       |
| SName    | varchar(15) | YES  |     | NULL    |       |
| Sex      | varchar(10) | YES  |     | NULL    |       |
| DOB      | date        | YES  |     | NULL    |       |
| Position | varchar(15) | YES  |     | NULL    |       |
| Salary   | varchar(7)  | YES  |     | NULL    |       |
| SID      | int(11)     | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> desc Branch;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| BNo      | int(11)     | NO   | PRI | NULL    |       |
| BAddress | varchar(30) | YES  |     | NULL    |       |
| BTele    | varchar(11) | YES  |     | NULL    |       |
| MID      | int(11)     | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc Owner;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| OID      | int(11)     | NO   | PRI | NULL    |       |
| OName    | varchar(15) | YES  |     | NULL    |       |
| OAddress | varchar(30) | YES  |     | NULL    |       |
| OTele    | varchar(11) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc Works;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| SID   | int(11) | NO   | PRI | NULL    |       |
| BID   | int(11) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> desc Property;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| PNo      | int(11)     | NO   | PRI | NULL    |       |
| Type     | varchar(10) | YES  |     | NULL    |       |
| Rooms    | int(11)     | YES  |     | NULL    |       |
| Rent     | varchar(6)  | YES  |     | NULL    |       |
| PAddress | varchar(30) | YES  |     | NULL    |       |
| OID      | int(11)     | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> desc Client;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| CNo     | int(11)     | NO   | PRI | NULL    |       |
| CName   | varchar(15) | YES  |     | NULL    |       |
| ReqType | varchar(10) | YES  |     | NULL    |       |
| MaxRent | varchar(6)  | YES  |     | NULL    |       |
| BID     | int(11)     | YES  | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc Lease;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| PID         | int(11)     | NO   | PRI | NULL    |       |
| CID         | int(11)     | NO   | PRI | NULL    |       |
| MonthlyRent | varchar(6)  | YES  |     | NULL    |       |
| Payment     | varchar(10) | YES  |     | NULL    |       |
| Duration    | varchar(7)  | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc Gen;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| RID   | int(11) | NO   | PRI | NULL    |       |
| BID   | int(11) | YES  | MUL | NULL    |       |
| RDate | date    | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> desc UserResp;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| RID     | int(11)     | NO   | PRI | NULL    |       |
| CID     | int(11)     | NO   | PRI | NULL    |       |
| Date    | date        | YES  |     | NULL    |       |
| Comment | varchar(50) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc Report;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| PID     | int(11)     | NO   | PRI | NULL    |       |
| RID     | int(11)     | NO   | PRI | NULL    |       |
| Type    | varchar(6)  | YES  |     | NULL    |       |
| Rent    | varchar(6)  | YES  |     | NULL    |       |
| Address | varchar(30) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql>
```

CONCLUSION:

Successfully created database schema using DDL commands

Exp. No. 3 **Date: 07/10/2022**

Modification of database schema - DDL

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:

show create table UserResp;
alter table UserResp drop foreign key UserResp_ibfk_1;
alter table UserResp drop foreign key UserResp_ibfk_2;

Table: Branch26

```
alter table Branch rename Branch26;
show create table Client; alter table Client drop foreign key Client_ibfk_1;
alter table Branch26 change BNo branchNo int;
alter table Branch26 change BAddress street varchar(10);
alter table Branch26 change BTele city varchar(10);
alter table Branch26 drop foreign key Branch26_ibfk_1;
alter table Branch26 drop column MID; alter table Branch26 add column postcode int;
```

Table: Staff26

```
alter table Staff rename Staff26;
show create table Staff26; alter table Staff26 drop foreign key Staff26_ibfk_1;
alter table Staff26 change SNo staffNo int; alter table Staff26 drop column SID;
alter table Staff26 change SName fName varchar(15);
alter table Staff26 change Sex lName varchar(15);
alter table Staff26 change Position sex varchar(7);
alter table Staff26 change DOB position varchar(15);
alter table Staff26 change Salary DOB date;
alter table Staff26 add column salary varchar(7);
alter table Staff26 add column branchNo int;
alter table Staff26 add foreign key(branchNo) references Branch26(branchNo);
```

Table: PrivateOwner26

```
alter table Owner rename PrivateOwner26;
alter table Property drop foreign key Property_ibfk_1;
alter table PrivateOwner26 change OID ownerNo int;
alter table PrivateOwner26 change OName fName varchar(15);
alter table PrivateOwner26 change OAddress lName varchar(15);
alter table PrivateOwner26 change OTele address varchar(30);
alter table PrivateOwner26 add column telNo varchar(11);
```

Table: PropertyForRent26

```
alter table Property rename PropertyForRent26;
alter table PropertyForRent26 change PNo propertyNo int;
alter table PropertyForRent26 change Type street varchar(10);
alter table PropertyForRent26 change Rooms city varchar(10);
alter table PropertyForRent26 change Rent postcode int;
alter table PropertyForRent26 change PAddress type varchar(10);
alter table PropertyForRent26 add column rooms int;
alter table PropertyForRent26 add column rent varchar(7);
alter table PropertyForRent26 add column ownerNo int;
alter table PropertyForRent26 add foreign key(ownerNo) references PrivateOwner26(ownerNo);
alter table PropertyForRent26 add column staffNo int;
```

alter table PropertyForRent26 add foreign key(staffNo) references Staff26(staffNo);
alter table PropertyForRent26 add column branchNo int;
alter table PropertyForRent26 add foreign key(branchNo) references Branch26(branchNo);
alter table PropertyForRent26 drop column OID;

Table: Client26

alter table Client rename Client26;
alter table Client26 change CNo clientNo int;
alter table Client26 change CName fName varchar(15);
alter table Client26 change ReqType lName varchar(15);
alter table Client26 change MaxRent telNo varchar(11);
alter table Client26 add column prefType varchar(10);
alter table Client26 add column maxRent varchar(7);
alter table Client26 drop column BID;

Table: Viewing26

alter table UserResp rename Viewing26;
alter table Viewing26 change RID clientNo int;
alter table Viewing26 change CID propertyNo int;
alter table Viewing26 change Date viewDate date;
alter table Viewing26 change Comment comment varchar(50);
alter table Viewing26 add foreign key(clientNo) references Client26(clientNo);
alter table Viewing26 add foreign key(propertyNo) references PropertyForRent26(propertyNo);

Table: Registration26

create table Registration26(clientNo int, branchNo int, staffNo int, dateJoined date, Primary Key(clientNo,branchNo), Foreign Key(clientno) references Client26(clientNo), Foreign Key(branchNo) references Branch26(branchNo), Foreign Key(staffNo) references Staff26(staffNo));

RESULT:

```
mysql> desc Branch26;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| branchNo | varchar(7)  | NO   | PRI | NULL    |       |
| street   | varchar(30) | YES  |     | NULL    |       |
| city     | varchar(15) | YES  |     | NULL    |       |
| postcode | varchar(15) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc Staff26;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| staffNo  | varchar(7)    | NO   | PRI | NULL    |       |
| fName    | varchar(10)   | YES  |     | NULL    |       |
| lName    | varchar(10)   | YES  |     | NULL    |       |
| position | varchar(10)   | YES  |     | NULL    |       |
| sex      | varchar(7)    | YES  |     | NULL    |       |
| DOB      | varchar(20)   | YES  |     | NULL    |       |
| salary   | decimal(10,0) | YES  |     | NULL    |       |
| branchNo | varchar(7)    | YES  | MUL | NULL    |       |
+----------+---------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

```
mysql> desc PrivateOwner26;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| ownerNo | varchar(7)  | NO   | PRI | NULL    |       |
| fName   | varchar(10) | YES  |     | NULL    |       |
| lName   | varchar(10) | YES  |     | NULL    |       |
| address | varchar(50) | YES  |     | NULL    |       |
| telNo   | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> desc PropertyForRent26;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| propertyNo | varchar(7)    | NO   | PRI | NULL    |       |
| street     | varchar(30)   | YES  |     | NULL    |       |
| city       | varchar(15)   | YES  |     | NULL    |       |
| postcode   | varchar(15)   | YES  |     | NULL    |       |
| type       | varchar(10)   | YES  |     | NULL    |       |
| rooms      | int           | YES  |     | NULL    |       |
| rent       | decimal(10,0) | YES  |     | NULL    |       |
| ownerNo    | varchar(7)    | YES  | MUL | NULL    |       |
| staffNo    | varchar(7)    | YES  | MUL | NULL    |       |
| branchNo   | varchar(7)    | YES  | MUL | NULL    |       |
+------------+---------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> desc Client26;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| clientNo | varchar(7)    | NO   | PRI | NULL    |       |
| fName    | varchar(10)   | YES  |     | NULL    |       |
| lName    | varchar(10)   | YES  |     | NULL    |       |
| telNo    | varchar(20)   | YES  |     | NULL    |       |
| prefType | varchar(10)   | YES  |     | NULL    |       |
| maxRent  | decimal(10,0) | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
mysql> desc Viewing26;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| clientNo   | varchar(7)  | NO   | PRI | NULL    |       |
| propertyNo | varchar(7)  | NO   | PRI | NULL    |       |
| viewDate   | varchar(20) | YES  |     | NULL    |       |
| comment    | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc Registration26;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| clientNo   | varchar(7)  | NO   | PRI | NULL    |       |
| branchNo   | varchar(7)  | NO   | PRI | NULL    |       |
| staffNo    | varchar(7)  | YES  | MUL | NULL    |       |
| dateJoined | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

CONCLUSION:

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

Exp. No. 4 **Date: 17/10/2022**

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_26 values
('B005','22 Deer Rd','London','SW1 4EH'),
('B007','16 Argyll St','Aberdeen','AB2 3SU'),
('B003','163 Main St','Glasgow','G11 9QX'),
('B002','56 Clover Dr','London','NW10 6EU');
```

```
insert into Staff_26 values
('SL21','John','M','1-Oct-45','Manager',30000,'White','B005'),
('SG37','Ann','F','10-Nov-60','Assistant',12000,'Beech','B003'),
('SG14','David','M','24-Mar26','Supervisor',18000,'Ford','B003'),
('SA9','Mary','F','19-Feb-70','Assistant',9000,'Howe','B007'),
('SG5','Susan','F','3-Jun-40','Manager',24000,'Brand','B003'),
('SL41','Julie','F','13-Jun-65','Assistant',9000,'Lee','B005');
```

```
insert into PropertyForRent_26 values
('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',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_26 values
('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);
```

insert into PrivateOwner_26 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 OQR','0141-225-7025');

insert into Viewing_26 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_26 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 Branch26;
+----------+--------------+----------+----------+
| 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 (0.00 sec)
```

```
mysql> select * from Staff26;
+---------+-------+-------+------------+------+-----------+--------+----------+
| 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 (0.26 sec)
```

```
mysql> select * from PrivateOwner26;
+---------+-------+--------+------------------------------+---------------+
| 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   | 0141-225-7025 |
+---------+-------+--------+------------------------------+---------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from PropertyForRent26;
+------------+--------------+---------+----------+-------+-------+------+---------+---------+----------+
| 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 | 400  | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St  | London  | NW2      | Flat  |     4 | 400  | CO87    | SL41    | B005     |
+------------+--------------+---------+----------+-------+-------+------+---------+---------+----------+
6 rows in set (0.00 sec)
```

```
mysql> select * from Client26;
+----------+-------+---------+---------------+----------+---------+
| clientNo | fName | lName   | telNo         | prefType | maxRent |
+----------+-------+---------+---------------+----------+---------+
| CR56     | Aline | Stewart | 0141-848-1825 | 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 Viewing26;
+----------+------------+-----------+----------------+
| 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)
```

```
mysql> select * from Registration26;
+----------+----------+---------+------------+
| 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)
```

CONCLUSION:

Successfully performed insertion operation in the given database

Exp. No. 5 **Date: 28/10/2022**

Practice SQL commands for DML

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

1. Give all staff a 3% pay increase

   update Staff26 set salary=salary*1.03;

```
mysql> select * from Staff26;
+---------+-------+-------+------------+------+-----------+--------+----------+
| staffNo | fname | lName | Position   | Sex  | DOB       | Salary | branchNo |
+---------+-------+-------+------------+------+-----------+--------+----------+
| SA9     | Mary  | Howe  | Assistant  | F    | 19-Feb-70 | 9270   | B007     |
| SG14    | David | Ford  | Supervisor | M    | 24-Mar-58 | 18540  | B003     |
| SG37    | Ann   | Beech | Assistant  | F    | 10-Nov-60 | 12360  | B003     |
| SG5     | Susan | Brand | Manager    | F    | 3-Jun-40  | 24720  | B003     |
| SL21    | John  | White | Manager    | M    | 1-Oct-45  | 30900  | B005     |
| SL41    | Julie | Lee   | Assistant  | F    | 13-Jun-65 | 9270   | B005     |
+---------+-------+-------+------------+------+-----------+--------+----------+
6 rows in set (0.00 sec)
```

2. Give all Managers a 5% pay increase.

   update Staff26 set salary=salary*1.05 where position='Manager';

```
mysql> select * from Staff26;
+---------+-------+-------+------------+------+-----------+--------+----------+
| staffNo | fname | lName | Position   | Sex  | DOB       | Salary | branchNo |
+---------+-------+-------+------------+------+-----------+--------+----------+
| SA9     | Mary  | Howe  | Assistant  | F    | 19-Feb-70 | 9270   | B007     |
| SG14    | David | Ford  | Supervisor | M    | 24-Mar-58 | 18540  | 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)
```

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

   update PropertyForRent26 set rent=rent*1.05;

```
mysql> select * from PropertyForRent26;
+------------+--------------+----------+----------+-------+-------+------+---------+---------+----------+
| 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     |
| PL94       | 6 Argyll St  | London   | NW2      | Flat  |     4 |  420 | CO87    | SL41    | B005     |
+------------+--------------+----------+----------+-------+-------+------+---------+---------+----------+
6 rows in set (0.00 sec)
```

4. Promote David Ford to Manager and change his salary to £18,000.

   update Staff26 set salary='18000', position='Manager' where fName='David' and lName='Ford';

```
mysql> select * from Staff26;
+---------+-------+-------+-----------+------+-----------+--------+----------+
| 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)
```

5. Delete all viewings that relate to property PG4.

   delete from Viewing26 where propertyNo='PG4';

```
mysql> select * from Viewing26;
+----------+------------+-----------+----------------+
| 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 |
+----------+------------+-----------+----------------+
3 rows in set (0.00 sec)
```

6. Delete all rows from the Viewing table.

   delete from Viewing26;

```
mysql> select * from Viewing26;
Empty set (0.00 sec)
```

Part 2 Select

1. List full details of all staff.

   select * from Staff26;

```
mysql> select * from Staff26;
+---------+-------+-------+-----------+------+-----------+--------+----------+
| 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 Staff26;

```
mysql> select staffNo, fName, lName, salary from Staff26;
+---------+-------+-------+--------+
| 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.

   select propertyNo from Viewing26;

```
mysql> select propertyNo from Viewing26;
+------------+
| propertyNo |
+------------+
| PA14       |
| PA14       |
| PG36       |
| PG4        |
| PG4        |
+------------+
5 rows in 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 from Staff26;

```
mysql> select staffNo, fName, lName, salary from Staff26;
+---------+-------+-------+--------+
| 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)
```

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

   select * from Staff26 where salary>10000;

```
mysql> select * from Staff26 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     |
+---------+-------+-------+-----------+------+-----------+--------+----------+
4 rows in set (0.00 sec)
```

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

   select street from Branch26 where city='London' or city='Glasgow';

```
mysql> select street from Branch26 where city='London' or city='Glasgow';
+--------------+
| street       |
+--------------+
| 56 Clover Dr |
| 163 Main St  |
| 22 Deer Rd   |
+--------------+
3 rows in set (0.00 sec)
```

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

   select * from Staff26 where salary between 20000 and 30000;

```
mysql> select * from Staff26 where salary between 20000 and 30000;
+---------+-------+-------+----------+------+----------+--------+----------+
| staffNo | fName | lName | position | sex  | DOB      | salary | branchNo |
+---------+-------+-------+----------+------+----------+--------+----------+
| SG5     | Susan | Brand | Manager  | F    | 3-Jun-40 | 25956  | B003     |
+---------+-------+-------+----------+------+----------+--------+----------+
1 row in set (0.00 sec)
```

8. List all managers and supervisors.

   select * from Staff26 where position='Manager' or position='Supervisor';

```
mysql> select * from Staff26 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     |
+---------+-------+-------+----------+------+-----------+--------+----------+
3 rows in set (0.01 sec)
```

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

   select * from PrivateOwner26 where address like '%Glasgow%';

```
mysql> select * from PrivateOwner26 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  | 0141-225-7025 |
+---------+-------+--------+-----------------------------+---------------+
3 rows in set (0.00 sec)
```

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

    select * from Viewing26 where propertyNo='PG4' and comment is NULL;

```
mysql> select * from Viewing26 where propertyNo='PG4' and comment is NULL;
+----------+------------+-----------+---------+
| clientNo | propertyNo | viewDate  | comment |
+----------+------------+-----------+---------+
| CR56     | PG4        | 26-May-04 | NULL    |
+----------+------------+-----------+---------+
1 row in set (0.00 sec)
```

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

    select salary from Staff26 order by salary desc;

```
mysql> select salary from Staff26 order by salary desc;
+--------+
| salary |
+--------+
| 9270   |
| 9270   |
| 32445  |
| 25956  |
| 18000  |
| 12360  |
+--------+
6 rows in set (0.00 sec)
```

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

    select * from PropertyForRent26 order by type;

```
mysql> select * from PropertyForRent26 order by type;
+-----------+--------------+----------+----------+-------+-------+--------+---------+---------+----------+
| propertyNo | street       | city     | postcode | type  | rooms | rent   | ownerNo | staffNo | branchNo |
+-----------+--------------+----------+----------+-------+-------+--------+---------+---------+----------+
| PG16      | 5 Novar Dr   | Glasgow  | G12 9AX  | Flat  |     4 | 472.5  | CO93    | SG14    | B003     |
| PG36      | 2 Manor Rd   | Glasgow  | G32 4QX  | Flat  |     3 | 393.75 | CO93    | SG37    | B003     |
| PG4       | 6 Lawrence St | Glasgow | G11 9QX  | Flat  |     3 | 420    | CO40    | NULL    | B003     |
| PL94      | 6 Argyll St  | London   | NW2      | Flat  |     4 | 420    | CO87    | SL41    | B005     |
| PA14      | 16 Holhead   | Aberdeen |  AB7 5SU | House |     6 | 682.5  | CO46    | SA9     | B007     |
| PG21      | 18 Dale Rd   | Glasgow  | G12      | House |     5 | 630    | CO87    | SG37    | B003     |
+-----------+--------------+----------+----------+-------+-------+--------+---------+---------+----------+
6 rows in set (0.00 sec)
```

CONCLUSION:

Successfully used DML commands for manipulating data

Exp. No. 6 **Date: 04/11/2022**

Implementation of Join Queries and Aggregate Functions

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:

I. **PART A**

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

   select c.fName, c.lName, v.comment from Client26 as c, Viewing26 as v where c.clientNo = v.clientNo;

```
mysql> select c.fName, c.lName, v.comment from Client26 as c, Viewing26 as v where c.clientNo = v.clientNo;
+-------+---------+----------------+
| fName | lName   | comment        |
+-------+---------+----------------+
| Aline | Stewart | too small      |
| Aline | Stewart | NULL           |
| Aline | Stewart | NULL           |
| Mary  | Tregear | no dining room |
| John  | Kay     | too remote     |
+-------+---------+----------------+
5 rows in set (0.02 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, s.fName, s.lName, p.propertyNo from Staff26 as s, PropertyForRent26 as p where s.staffNo = p.staffNo order by s.branchNo;

```
mysql> select s.branchNo, s.staffNo, s.fName, s.lName, p.propertyNo from Staff26 as s, PropertyForRent26 as p where s.staffNo = p.staffNo order by s.branchNo;
+----------+---------+-------+-------+------------+
| 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 (0.06 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 s.branchNo, s.staffNo, s.fName, s.lName, p.propertyNo, p.city from Staff26 as s, PropertyForRent26 as p where s.staffNo = p.staffNo order by s.branchNo;

```
mysql> select s.branchNo, s.staffNo, s.fName, s.lName, p.propertyNo, p.city from Staff26 as s, PropertyForRent26 as p where s.staffNo = p.staffNo order by s.branchNo;
+----------+---------+-------+-------+------------+----------+
| branchNo | staffNo | fName | lName | propertyNo | city     |
+----------+---------+-------+-------+------------+----------+
| B003     | SG14    | David | Ford  | PG16       | Glasgow  |
| B003     | SG37    | Ann   | Beech | PG21       | Glasgow  |
| B003     | SG37    | Ann   | Beech | PG36       | Glasgow  |
| B005     | SL41    | Julie | Lee   | PL94       | London   |
| B007     | SA9     | Mary  | Howe  | PA14       | Aberdeen |
+----------+---------+-------+-------+------------+----------+
5 rows in set (0.00 sec)
```

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

   select b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b left join PropertyForRent26 as p on b.city = p.city;

```
mysql> select b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b left join PropertyForRent26 as p on b.city = p.city;
+----------+-------------+------------+---------------+
| branchNo | branch_city | propertyNo | property_city |
+----------+-------------+------------+---------------+
| 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, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b right join PropertyForRent26 as p on b.city = p.city;

```
mysql> select b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b right join PropertyForRent26 as p on b.city = p.city;
+----------+-------------+------------+---------------+
| branchNo | branch_city | propertyNo | property_city |
+----------+-------------+------------+---------------+
| 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 (0.00 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 branch_city, p.propertyNo, p.city as property_city from Branch26 as b left join PropertyForRent26 as p on b.city = p.city union select b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b right join PropertyForRent26 as p on b.city = p.city;

```
mysql> select b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b left join PropertyForRent26 as p on b.city = p.city union sele
ct b.branchNo, b.city as branch_city, p.propertyNo, p.city as property_city from Branch26 as b right join PropertyForRent26 as p on b.city = p.city;
+----------+-------------+-----------+--------------+
| branchNo | branch_city | propertyNo | property_city |
+----------+-------------+-----------+--------------+
| 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)
```

II. **PART B**

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

   select count(*) as count_property from PropertyForRent26 where rent>350;

```
mysql> select count(*) as count_property from PropertyForRent26 where rent>350;
+----------------+
| count_property |
+----------------+
|              6 |
+----------------+
1 row in set (0.09 sec)
```

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

   select count(distinct propertyNo) as count_distinct_property from Viewing26 where viewDate like '%-May-04';

```
mysql> select count(distinct propertyNo) as count_distinct_property from Viewing26 where viewDate like '%-May-04';
+------------------------+
| count_distinct_property |
+------------------------+
|                      2 |
+------------------------+
1 row in set (0.00 sec)
```

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

   select count(staffNo) as no_of_managers, sum(salary) as total_salary from Staff26 where position='Manager';

```
mysql> select count(staffNo) as no_of_managers, sum(salary) as total_salary from Staff26 where position='Manager';
+---------------+--------------+
| no_of_managers | total_salary |
+---------------+--------------+
|             3 |        76401 |
+---------------+--------------+
1 row in set (0.00 sec)
```

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

    select min(salary) as min_salary, max(salary) as max_salary, avg(salary) as avg_salary from Staff26;

```
mysql> select min(salary) as min_salary, max(salary) as max_salary, avg(salary) as avg_salary from Staff26;
+------------+------------+------------+
| min_salary | max_salary | avg_salary |
+------------+------------+------------+
|       9270 |      32445 | 17883.5000 |
+------------+------------+------------+
1 row in set (0.02 sec)
```

11. Find total number of properties with 3 rooms.

    select count(propertyNo) as count_property from PropertyForRent26 where rooms=3;

```
mysql> select count(propertyNo) as count_property from PropertyForRent26 where rooms=3;
+----------------+
| count_property |
+----------------+
|              2 |
+----------------+
1 row in set (0.00 sec)
```

12. Find minimum, maximum and average property rent.

    select min(rent) as min_rent, max(rent) as max_rent, avg(rent) as avg_rent from PropertyForRent26;

```
mysql> select min(rent) as min_rent, max(rent) as max_rent, avg(rent) as avg_rent from PropertyForRent26;
+----------+----------+----------+
| min_rent | max_rent | avg_rent |
+----------+----------+----------+
|      368 |      683 | 494.6667 |
+----------+----------+----------+
1 row in set (0.00 sec)
```

CONCLUSION:

Successfully implemented DQL Commands

Exp. No. 7 **Date: 14/11/2022**

Implementation of Group By & Having clause

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 _ ete. 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 branchNo, count(staffNo) as no_of_staffs, sum(salary) as total_salary from Staff26 group by branchNo;

```
mysql> select branchNo, count(staffNo) as no_of_staffs, sum(salary) as total_salary from Staff26 group by branchNo;
+----------+--------------+--------------+
| branchNo | no_of_staffs | total_salary |
+----------+--------------+--------------+
| B003     |            3 |        56316 |
| B005     |            2 |        41715 |
| B007     |            1 |         9270 |
+----------+--------------+--------------+
3 rows in set (0.02 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 branchNo, count(staffNo) as no_of_staffs, sum(salary) as total_salary from Staff26 group by branchNo having count(staffNo)>1;

```
mysql> select branchNo, count(staffNo) as no_of_staffs, sum(salary) as total_salary from Staff26 group by branchNo having count(staffNo)>1;
+----------+--------------+--------------+
| branchNo | no_of_staffs | total_salary |
+----------+--------------+--------------+
| B003     |            3 |        56316 |
| B005     |            2 |        41715 |
+----------+--------------+--------------+
2 rows in set (0.02 sec)
```

3. Find average salaries of staff at various positions.

select position, avg(salary) as average_salary from Staff26 group by position;

```
mysql> select position, avg(salary) as average_salary from Staff26 group by position;
+------------+----------------+
| position   | average_salary |
+------------+----------------+
| Assistant |     10300.0000 |
| Manager   |     25467.0000 |
+------------+----------------+
2 rows in set (0.05 sec)
```

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

select city, count(propertyNo) as no_of_properties from PropertyForRent26 group by city;

```
mysql> select city, count(propertyNo) as no_of_properties from PropertyForRent26 group by city;
+----------+------------------+
| city     | no_of_properties |
+----------+------------------+
| Aberdeen |                1 |
| Glasgow  |                4 |
| London   |                1 |
+----------+------------------+
3 rows in set (0.00 sec)
```

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(propertyNo) as no_of_properties from PropertyForRent26 group by city having count(propertyNo)>2;

```
mysql> select city, count(propertyNo) as no_of_properties from PropertyForRent26 group by city having count(propertyNo)>2;
+---------+------------------+
| city    | no_of_properties |
+---------+------------------+
| Glasgow |                4 |
+---------+------------------+
1 row in set (0.03 sec)
```

6. Find the number of houses and flats available for rent.

select type, count(propertyNo) as no_of_properties from PropertyForRent26 group by type;

```
mysql> select type, count(propertyNo) as no_of_properties from PropertyForRent26 group by type;
+-------+-------------------+
| type  | no_of_properties |
+-------+-------------------+
| House |                 2 |
| Flat  |                 4 |
+-------+-------------------+
2 rows in set (0.00 sec)
```

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

select city, count(propertyNo) as no_of_properties, avg(rent) as average_rent from PropertyForRent26 group by city having count(propertyNo)>1;

```
mysql> select city, count(propertyNo) as no_of_properties, avg(rent) as average_rent from PropertyForRent26 group by city having count(propertyNo)>1;
+--------+-------------------+--------------+
| city   | no_of_properties | average_rent |
+--------+-------------------+--------------+
| Glasgow |                 4 |     466.2500 |
+--------+-------------------+--------------+
1 row in set (0.00 sec)
```

CONCLUSION:

Implementation of Group By & Having clause was successful

Exp. No. 8 **Date: 14/11/2022**

Implementation of set operators andnested queries

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 Branch26 union select city from PropertyForRent26;

```
mysql> select city from Branch26 union select city from PropertyForRent26;
+----------+
| city     |
+----------+
| London   |
| Glasgow  |
| Bristol  |
| Aberdeen |
+----------+
4 rows in set (0.09 sec)
```

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

   select distinct city from Branch26 where city in (select city from PropertyForRent26);

```
mysql> select distinct city from Branch26 where city in (select city from PropertyForRent26);
+----------+
| city     |
+----------+
| London   |
| Glasgow  |
| Aberdeen |
+----------+
3 rows in set (0.00 sec)
```

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

   select distinct city from Branch26 where city not in (select city from PropertyForRent26);

```
mysql> select distinct city from Branch26 where city not in (select city from PropertyForRent26);
+----------+
| city     |
+----------+
| Bristol |
+----------+
1 row in set (0.03 sec)
```

Nested Queries

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

   select staffNo from Staff26 where branchNo in (select branchNo from Branch26 where street='163 Main St');

```
mysql> select staffNo from Staff26 where branchNo in (select branchNo from Branch26 where street='163 Main St');
+----------+
| staffNo |
+----------+
| SG14     |
| SG37     |
| SG5      |
+----------+
3 rows in set (0.00 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, (salary-(select avg(salary) from Staff26)) as salary_difference from Staff26 where salary>(select avg(salary) from Staff26);

```
mysql> select staffNo, (salary-(select avg(salary) from Staff26)) as salary_difference from Staff26 where salary>(select avg(salary) from Staff26);
+---------+-------------------+
| staffNo | salary_difference |
+---------+-------------------+
| SG14    |          116.5000 |
| SG5     |         8072.5000 |
| SL21    |        14561.5000 |
+---------+-------------------+
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 from PropertyForRent26 where staffNo in (select staffNo from Staff26 where branchNo in (select branchNo from Branch26 where street='163 Main St'));

```
mysql> select propertyNo from PropertyForRent26 where staffNo in (select staffNo from Staff26 where branchNo in (select branchNo from Branch26 where street='163 Main St
'));
+------------+
| propertyNo |
+------------+
| PG16       |
| PG21       |
| PG36       |
+------------+
3 rows in set (0.00 sec)
```

7. Find all staff whose salary is larger than the salary of at least one member of staff at branch B003.

   select staffNo from Staff26 where salary>some(select salary from Staff26 where branchNo='B003');

```
mysql> select staffNo from Staff26 where salary>some(select salary from Staff26 where branchNo='B003');
+---------+
| staffNo |
+---------+
| SG14    |
| SG5     |
| SL21    |
+---------+
3 rows in set (0.03 sec)
```

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

   select staffNo from Staff26 where salary>all(select salary from Staff26 where branchNo='B003');

```
mysql> select staffNo from Staff26 where salary>all(select salary from Staff26 where branchNo='B003');
+---------+
| staffNo |
+---------+
| SL21    |
+---------+
1 row in set (0.04 sec)
```

CONCLUSION:

Successfully implemented set operators and nested queries.

Exp. No. 9 **Date: 21/11/2022**

Implementation of views, practice DCL commands and practice TCL commands

AIM:

Implementation of views, practice 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 managerB003 AS select * from Staff26 where branchNo='B003';

```
mysql> create VIEW managerB003 AS select * from Staff26 where branchNo='B003';
Query OK, 0 rows affected (0.18 sec)

mysql> select * from managerB003;
+---------+-------+-------+-----------+------+-----------+--------+----------+
| 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     |
+---------+-------+-------+-----------+------+-----------+--------+----------+
3 rows in set (0.00 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 branchB003 AS select staffNo, fName, lName, position, sex, DOB, branchNo from Staff26 where branchNo='B003';

```
mysql> create VIEW branchB003 AS select staffNo, fName, lName, position, sex, DOB, branchNo from Staff26 where branchNo='B003';
Query OK, 0 rows affected (1.45 sec)

mysql> select * from branchB003;
+---------+-------+-------+-----------+------+-----------+----------+
| 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 manageproperty AS select p.branchNo, p.staffNo, count(*) as count_property from PropertyForRent26 as p, Staff26 as s where p.staffNo=s.staffNo group by staffNo, branchNo;

```
mysql> create VIEW manageproperty AS select p.branchNo, p.staffNo, count(*) as count_property from PropertyForRent26 as p, Staff26 as s where p.staffNo=s.staffNo group
by staffNo, branchNo;
Query OK, 0 rows affected (0.16 sec)

mysql> select * from manageproperty;
+----------+---------+----------------+
| branchNo | staffNo | count_property |
+----------+---------+----------------+
| B007     | SA9     |              1 |
| B003     | SG14    |              1 |
| B003     | SG37    |              2 |
| B005     | SL41    |              1 |
+----------+---------+----------------+
4 rows in set (0.17 sec)
```

Part 2: DCL (Data Control Language)

Commands to grant and revoke privileges.

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

create user USER1@'localhost' identified by 'root1';

```
mysql> create user USER1@'localhost' identified by 'root1';
Query OK, 0 rows affected (0.01 sec)

mysql> select Host, User from mysql.user;
+-----------+------------------+
| Host      | User             |
+-----------+------------------+
| localhost | USER1            |
| localhost | arish            |
| localhost | debian-sys-maint |
| localhost | mysql.session    |
| localhost | mysql.sys        |
| localhost | root             |
+-----------+------------------+
6 rows in set (0.00 sec)
```

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

   GRANT ALL ON DreamHome.Branch30 TO USER1@'localhost';

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

   GRANT SELECT ON DreamHome.PropertyForRent30 TO USER1@'localhost';

4. Validate the privilege assignments with proper queries as the new user. (Write all queries and outputs used for validation)

```
mysql> select * from PropertyForRent30;
+------------+--------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street       | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+--------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead   | Aberdeen | AB7 55U  | 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     |
| PL94       | 6 Argyll St  | London   | NW2      | Flat  |     4 |  420 | CO87    | SL41    | B005     |
+------------+--------------+----------+----------+-------+-------+------+---------+---------+----------+
6 rows in set (0.00 sec)

mysql> update PropertyForRent30 set rooms=6 where branchNo='B005';
ERROR 1142 (42000): UPDATE command denied to user 'USER1'@'localhost' for table 'PropertyForRent30'
```

```
mysql> select * from Branch30;
+----------+-------------+----------+----------+
| 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 Branch30 where branchNo='B004';
Query OK, 1 row affected (0.03 sec)

mysql> select * from Branch30;
+----------+-------------+----------+----------+
| branchNo | street      | city     | postcode |
+----------+-------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU |
| B003     | 163 Main St  | Glasgow  | G11 9QX  |
| B005     | 22 Deer Rd   | London   | SW1 4EH  |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  |
+----------+-------------+----------+----------+
4 rows in set (0.00 sec)

mysql> insert into Branch30 values('B004', '32 Manse Rd', 'London', 'BS99 1NZ');
Query OK, 1 row affected (0.48 sec)

mysql> select * from Branch30;
+----------+-------------+----------+----------+
| branchNo | street      | city     | postcode |
+----------+-------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU |
| B003     | 163 Main St  | Glasgow  | G11 9QX  |
| B004     | 32 Manse Rd  | London   | BS99 1NZ |
| B005     | 22 Deer Rd   | London   | SW1 4EH  |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  |
+----------+-------------+----------+----------+
5 rows in set (0.00 sec)
```

```
mysql> update Branch30 set city='Bristol' where branchNo='B004';
Query OK, 1 row affected (0.47 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Branch30;
+----------+-------------+----------+----------+
| 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)
```

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

```
mysql> REVOKE ALL ON DreamHome.Branch30 FROM USER1@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> REVOKE SELECT ON DreamHome.PropertyForRent30 FROM 'USER1'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> use DreamHome;
ERROR 1044 (42000): Access denied for user 'USER1'@'localhost' to database 'DreamHome'
mysql>
```

Part 3: TCL (Transaction Control Language)

Commit, Rollback and Savepoint.

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

   create table TEMP(A1 int, A2 varchar(10), A3 int);

```
mysql> desc TEMP;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| A1    | int         | YES  |     | NULL    |       |
| A2    | varchar(10) | YES  |     | NULL    |       |
| A3    | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.26 sec)
```

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

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into TEMP values(1, 'FIRST', 100);
Query OK, 1 row affected (0.04 sec)

mysql> insert into TEMP values(2, 'SECOND', 200);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT A;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into TEMP values(3, 'THIRD', 300);
Query OK, 1 row affected (0.00 sec)

mysql> insert into TEMP values(4, 'FOURTH', 400);
Query OK, 1 row affected (0.00 sec)

mysql> insert into TEMP values(5, 'FIFTH', 500);
Query OK, 1 row affected (0.00 sec)

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

mysql> insert into TEMP values(6, 'SIXTH', 600);
Query OK, 1 row affected (0.00 sec)

mysql> insert into TEMP values(7, 'SEVENTH', 700);
Query OK, 1 row affected (0.01 sec)

mysql> insert into TEMP values(8, 'EIGHTH', 800);
Query OK, 1 row affected (0.00 sec)

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

mysql> insert into TEMP values(9, 'NINETH', 900);
Query OK, 1 row affected (0.00 sec)

mysql> insert into TEMP values(10, 'TENTH', 1000);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from TEMP;
+------+---------+------+
| A1   | A2      | A3   |
+------+---------+------+
|    1 | FIRST   |  100 |
|    2 | SECOND  |  200 |
|    3 | THIRD   |  300 |
|    4 | FOURTH  |  400 |
|    5 | FIFTH   |  500 |
|    6 | SIXTH   |  600 |
|    7 | SEVENTH |  700 |
|    8 | EIGHTH  |  800 |
|    9 | NINETH  |  900 |
|   10 | TENTH   | 1000 |
+------+---------+------+
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 (0.08 sec)

mysql> select * from TEMP;
+------+---------+------+
| A1   | A2      | A3   |
+------+---------+------+
|    1 | FIRST   |  100 |
|    2 | SECOND  |  200 |
|    3 | THIRD   |  300 |
|    4 | FOURTH  |  400 |
|    5 | FIFTH   |  500 |
|    6 | SIXTH   |  600 |
|    7 | SEVENTH |  700 |
|    8 | EIGHTH  |  800 |
+------+---------+------+
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 | FIRST  |  100 |
|    2 | SECOND |  200 |
|    3 | THIRD  |  300 |
|    4 | FOURTH |  400 |
|    5 | FIFTH  |  500 |
+------+--------+------+
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 | FIRST  |  100 |
|    2 | SECOND |  200 |
+------+--------+------+
2 rows in set (0.00 sec)
```

4. Commit the current state of TEMP. (Validate the commit operation)

```
mysql> COMMIT;
Query OK, 0 rows affected (0.55 sec)

mysql> ROLLBACK TO A;
ERROR 1305 (42000): SAVEPOINT A does not exist
mysql> ROLLBACK TO B;
ERROR 1305 (42000): SAVEPOINT B does not exist
mysql> ROLLBACK TO C;
ERROR 1305 (42000): SAVEPOINT C does not exist
```

CONCLUSION:

Successfully implemented set operators and nested queries.

Exp. No. 10 **Date: 27/11/2022**

Familiarize various Built-in functions available 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. AGGREGATE FUNCTIONS:

- **AVG()**

  Return the average of non-NULL values.

  AVG(expression)

```
mysql> select AVG(salary) from Staff26;
+-------------+
| AVG(salary) |
+-------------+
|  17883.5000 |
+-------------+
1 row in set (0.01 sec)
```

- **MAX()**

  Return the highest value (maximum) in a set of non-NULL values.

  MAX(expression)

```
mysql> select MAX(salary) from Staff26;
+-------------+
| MAX(salary) |
+-------------+
|       32445 |
+-------------+
1 row in set (0.00 sec)
```

- **MIN()**
  Return the lowest value (minimum) in a set of non-NULL values.

  MIN(expression)

```
mysql> select MIN(salary) from Staff26;
+-------------+
| MIN(salary) |
+-------------+
|        9270 |
+-------------+
1 row in set (0.00 sec)
```

- **COUNT()**
  Return the number of rows in a group, including rows with NULL values.

  COUNT(expression): excludes NULL values
  COUNT(*): includes NULL values and duplicate rows

```
mysql> select COUNT(*) from Staff26;
+----------+
| COUNT(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

- **SUM()**
  Return the summation of all non-NULL values a set.

  SUM(expression)

```
mysql> select SUM(salary) from Staff26;
+-------------+
| SUM(salary) |
+-------------+
|      107301 |
+-------------+
1 row in set (0.00 sec)
```

- **GROUP_CONCAT()**
  Return a concatenated string.

  GROUP_CONCAT(
  DISTINCT expression
  ORDER BY expression
  SEPARATOR sep
  );

```
mysql> select GROUP_CONCAT(city) from Branch26;
+---------------------------------------+
| GROUP_CONCAT(city)                    |
+---------------------------------------+
| London,Glasgow,Bristol,London,Aberdeen |
+---------------------------------------+
1 row in set (0.02 sec)
```

II. DATE FUNCTIONS:

- **CURDATE()**
  Returns the current date.

  CURDATE();

```
mysql> select CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2022-11-27 |
+------------+
1 row in set (0.00 sec)
```

- **DAY()**
  Gets the day of the month of a specified date.

  DAY(date)

```
mysql> select DAY('2022-11-27');
+-------------------+
| DAY('2022-11-27') |
+-------------------+
|                27 |
+-------------------+
1 row in set (0.00 sec)
```

- **DAYNAME()**
  Gets the name of a weekday for a specified date.

  DAYNAME(date)

```
mysql> select DAYNAME('2022-11-27');
+-----------------------+
| DAYNAME('2022-11-27') |
+-----------------------+
| Sunday                |
+-----------------------+
1 row in set (0.00 sec)
```

- **DAYOFWEEK()**
  Returns the weekday index for a date.

  DAYOFWEEK(date)

```
mysql> select DAYOFWEEK('2022-11-27');
+-------------------------+
| DAYOFWEEK('2022-11-27') |
+-------------------------+
|                       1 |
+-------------------------+
1 row in set (0.00 sec)
```

- **NOW()**
  Returns the current date and time at which the statement executed.

  NOW();

```
mysql> select NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-11-27 18:21:13 |
+---------------------+
1 row in set (0.00 sec)
```

- **WEEK()**
  Returns a week number of a date.

  WEEK(date, mode)

```
mysql> select WEEK('2022-11-27');
+--------------------+
| WEEK('2022-11-27') |
+--------------------+
|                 48 |
+--------------------+
1 row in set (0.00 sec)
```

- **WEEKDAY()**
  Returns a weekday index for a date.

  WEEKDAY(date)

```
mysql> select WEEKDAY('2022-11-27');
+-----------------------+
| WEEKDAY('2022-11-27') |
+-----------------------+
|                     6 |
+-----------------------+
1 row in set (0.00 sec)
```

- **YEAR()**
  Return the year for a specified date

  YEAR(date)

```
mysql> select YEAR('2022-11-27');
+--------------------+
| YEAR('2022-11-27') |
+--------------------+
|               2022 |
+--------------------+
1 row in set (0.00 sec)
```

- **MONTH()**
  Returns an integer that represents a month of a specified date.

  MONTH(date)

```
mysql> select MONTH('2022-11-27');
+---------------------+
| MONTH('2022-11-27') |
+---------------------+
|                  11 |
+---------------------+
1 row in set (0.00 sec)
```

III. STRING FUNCTIONS:

- **CONCAT()**

  Concatenate two or more strings into a single string

  CONCAT(string1,string2, ...)

```
mysql> select CONCAT('ABCD', '1234');
+------------------------+
| CONCAT('ABCD', '1234') |
+------------------------+
| ABCD1234               |
+------------------------+
1 row in set (0.00 sec)
```

- **LENGTH()**

  Get the length of a string in bytes and in characters

  LENGTH(str)

```
mysql> select LENGTH('abcd1234');
+--------------------+
| LENGTH('abcd1234') |
+--------------------+
|                  8 |
+--------------------+
1 row in set (0.00 sec)
```

- **LEFT()**

  Get a specified number of leftmost characters from a string

  LEFT(str,length)

```
mysql> select LEFT('abcd1234', 5);
+---------------------+
| LEFT('abcd1234', 5) |
+---------------------+
| abcd1               |
+---------------------+
1 row in set (0.00 sec)
```

- **LOWER()**

  Convert a string to lowercase

  LOWER(str)

```
mysql> select LOWER('AbCdEf');
+-----------------+
| LOWER('AbCdEf') |
+-----------------+
| abcdef          |
+-----------------+
1 row in set (0.00 sec)
```

- **UPPER()**
  Convert a string to uppercase

  UPPER(str)

```
mysql> select UPPER('aBcDeF');
+-----------------+
| UPPER('aBcDeF') |
+-----------------+
| ABCDEF          |
+-----------------+
1 row in set (0.00 sec)
```

- **RIGHT()**
  Get a specified number of rightmost characters from a string

  RIGHT(str,length)

```
mysql> select RIGHT('abcd1234', 5);
+----------------------+
| RIGHT('abcd1234', 5) |
+----------------------+
| d1234                |
+----------------------+
1 row in set (0.00 sec)
```

IV. MATH FUNCTIONS:

- **ABS()**
  Returns the absolute value of a number

  ABS(n)

```
mysql> select ABS(-26), ABS(0), ABS(26);
+----------+--------+---------+
| ABS(-26) | ABS(0) | ABS(26) |
+----------+--------+---------+
|       26 |      0 |      26 |
+----------+--------+---------+
1 row in set (0.00 sec)
```

- **CEIL()**
  Returns the smallest integer value greater than or equal to the input number (n).

  CEIL (numeric_expression)

```
mysql> select CEIL(-1.234), CEIL(1.234);
+--------------+-------------+
| CEIL(-1.234) | CEIL(1.234) |
+--------------+-------------+
|           -1 |           2 |
+--------------+-------------+
1 row in set (0.00 sec)
```

- **FLOOR()**
  Returns the largest integer value not greater than the argument

  FLOOR(expression)

```
mysql> select FLOOR(-1.234), FLOOR(1.234);
+---------------+--------------+
| FLOOR(-1.234) | FLOOR(1.234) |
+---------------+--------------+
|            -2 |            1 |
+---------------+--------------+
1 row in set (0.00 sec)
```

- **MOD()**
  Returns the remainder of a number divided by another

  MOD(dividend,divisor)

```
mysql> select MOD(33, 2), MOD(45, 9);
+------------+------------+
| MOD(33, 2) | MOD(45, 9) |
+------------+------------+
|          1 |          0 |
+------------+------------+
1 row in set (0.00 sec)
```

- **ROUND()**
  Rounds a number to a specified number of decimal places.

  ROUND(n,[d])

```
mysql> select ROUND(11.3), ROUND(11.6), ROUND(11.385, 1), ROUND(11.345, 0), ROUND(11.385, -1);
+-------------+-------------+-----------------+-----------------+------------------+
| ROUND(11.3) | ROUND(11.6) | ROUND(11.385, 1) | ROUND(11.345, 0) | ROUND(11.385, -1) |
+-------------+-------------+-----------------+-----------------+------------------+
|          11 |          12 |            11.4 |              11 |               10 |
+-------------+-------------+-----------------+-----------------+------------------+
1 row in set (0.00 sec)
```

- **TRUNCATE()**
  Truncates a number to a specified number of decimal places

  TRUNCATE(X,D)

```
mysql> select TRUNCATE(11.385, 1), TRUNCATE(11.345, 0), TRUNCATE(11.385, -1), TRUNCATE(11.345, 2);
+---------------------+---------------------+----------------------+---------------------+
| TRUNCATE(11.385, 1) | TRUNCATE(11.345, 0) | TRUNCATE(11.385, -1) | TRUNCATE(11.345, 2) |
+---------------------+---------------------+----------------------+---------------------+
|                11.3 |                  11 |                   10 |               11.34 |
+---------------------+---------------------+----------------------+---------------------+
1 row in set (0.00 sec)
```

CONCLUSION:

Successfully familiarized with various Built-in functions available in MySQL

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

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 labellingdata 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 $$
drop procedure if exists largestno$$
create procedure largestno(IN a int, IN b int, IN c int)
begin
        if a>b then
                if a>c then
                        select a as largest;
                else
                        select c as largest;
                end if;
        else
                if b>c then
                        select b as largest;
                else
                        select c as largest;
                end if;
        end if;
end$$
delimiter ;
```

```
mysql> source largest.sql;
Query OK, 0 rows affected (0.80 sec)

Query OK, 0 rows affected (0.43 sec)

mysql> call largestno(10, 20, 30);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.06 sec)

Query OK, 0 rows affected (0.06 sec)

mysql> call largestno(30, 20, 10);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call largestno(20, 30, 10);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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 $$
drop procedure if exists largestno$$
create procedure largestno(IN a int, IN b int, IN c int, OUT large int)
begin
        if a>b then
                if a>c then
                        set large=a;
                else
                        set large=c;
                end if;
        else
                if b>c then
                        set large=b;
                else
                        set large=c;
                end if;
        end if;
end$$
delimiter ;
```

```
mysql> source largestinout.sql;
Query OK, 0 rows affected (0.17 sec)

Query OK, 0 rows affected (0.21 sec)

mysql> call largestno(10, 20, 30, @large);
Query OK, 0 rows affected (0.00 sec)

mysql> select @large;
+--------+
| @large |
+--------+
|     30 |
+--------+
1 row in set (0.00 sec)

mysql> call largestno(30, 20, 10, @large);
Query OK, 0 rows affected (0.00 sec)

mysql> select @large;
+--------+
| @large |
+--------+
|     30 |
+--------+
1 row in set (0.00 sec)

mysql> call largestno(20, 30, 10, @large);
Query OK, 0 rows affected (0.00 sec)

mysql> select @large;
+--------+
| @large |
+--------+
|     30 |
+--------+
1 row in set (0.00 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists largestno$$
create procedure largestno(INOUT a int, IN b int, IN c int)
begin
        if a>b then
                if a>c then
                        set a=a;
                        select a as largest;
```

```
                else
                        set a=c;
                        select c as largest;
                end if;
        else
                if b>c then
                        set a=b;
                        select b as largest;
                else
                        set a=c;
                        select c as largest;
                end if;
        end if;
end$$
delimiter ;
```

```
mysql> source inoutlargest.sql;
Query OK, 0 rows affected (0.21 sec)

Query OK, 0 rows affected (0.39 sec)

mysql> set @a=10;
Query OK, 0 rows affected (0.02 sec)

mysql> call largestno(@a, 20, 30);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> set @a=20;
Query OK, 0 rows affected (0.00 sec)

mysql> call largestno(@a, 30, 10);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> set @a=30;
Query OK, 0 rows affected (0.00 sec)

mysql> call largestno(@a, 20, 10);
+---------+
| largest |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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(mark int)
begin
	case
		when mark>90 then select 'A' as grade;
		when mark>80 then select 'B' as grade;
		when mark>70 then select 'C' as grade;
		when mark>60 then select 'D' as grade;
		when mark>50 then select 'E' as grade;
		else
			select 'F' as grade;
	end case;
end$$
delimiter ;
```

```
mysql> source grade.sql;
Query OK, 0 rows affected, 1 warning (0.12 sec)

Query OK, 0 rows affected (0.20 sec)

mysql> call grade(95);
+-------+
| grade |
+-------+
| A     |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call grade(65);
+-------+
| grade |
+-------+
| D     |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call grade(35);
+-------+
| grade |
+-------+
| F     |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists sumofn$$
create procedure sumofn(n int)
begin
	declare s int;
	declare i int;
	set s=0;
	set i=1;
	l1:loop
		if i>n then
			leave l1;
		end if;
		set s=s+i;
		set i=i+1;
	end loop;
	select s as sumofn;
end$$
delimiter ;
```

```
mysql> source sumofn.sql;
Query OK, 0 rows affected (0.25 sec)

Query OK, 0 rows affected (0.20 sec)

mysql> call sumofn(10);
+--------+
| sumofn |
+--------+
|     55 |
+--------+
1 row in set (0.03 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> call sumofn(100);
+--------+
| sumofn |
+--------+
|   5050 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call sumofn(1000);
+--------+
| sumofn |
+--------+
| 500500 |
+--------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists sumofn$$
create procedure sumofn(n int)
begin
        declare s int;
        declare i int;
        set s=0;
        set i=1;
        l1:repeat
                set s=s+i;
                set i=i+2;
          until i>n
        end repeat;
        select s as sumofn;
end$$
delimiter ;
```

```
mysql> source repeatsum.sql;
Query OK, 0 rows affected (0.19 sec)

Query OK, 0 rows affected (0.28 sec)

mysql> call sumofn(10);
+--------+
| sumofn |
+--------+
|     25 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call sumofn(100);
+--------+
| sumofn |
+--------+
|   2500 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call sumofn(1000);
+--------+
| sumofn |
+--------+
| 250000 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists sumofn$$
create procedure sumofn(n int)
begin
	declare s int;
	declare i int;
	set s=0;
	set i=2;
	l1:while i<=n do
		set s=s+i;
		set i=i+2;
	end while;
	select s as sumofn;
end$$
delimiter ;
```

```
mysql> source whilesum.sql;
Query OK, 0 rows affected (0.20 sec)

Query OK, 0 rows affected (0.19 sec)

mysql> call sumofn(10);
+--------+
| sumofn |
+--------+
|     30 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call sumofn(100);
+--------+
| sumofn |
+--------+
|   2550 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call sumofn(1000);
+--------+
| sumofn |
+--------+
| 250500 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists digits$$
create procedure digits(n int)
begin
        declare nod int;
        set nod=0;
        l1:while n>0 do
                set n=n div 10;
                set nod=nod+1;
        end while;
        select nod as no_of_digits;
end$$
delimiter ;
```

```
mysql> source digits.sql;
Query OK, 0 rows affected, 1 warning (0.06 sec)

Query OK, 0 rows affected (0.26 sec)

mysql> call digits(111);
+--------------+
| no_of_digits |
+--------------+
|            3 |
+--------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call digits(11123);
+--------------+
| no_of_digits |
+--------------+
|            5 |
+--------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call digits(11123432);
+--------------+
| no_of_digits |
+--------------+
|            8 |
+--------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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;
        select a as num1, b as num2;
        set temp=a;
        set a=b;
        set b=temp;
        select a as num1, b as num2;
end$$
delimiter ;
```

```
mysql> source swap.sql;
Query OK, 0 rows affected (0.32 sec)

Query OK, 0 rows affected (0.20 sec)

mysql> set @a=10;
Query OK, 0 rows affected (0.05 sec)

mysql> set @b=20;
Query OK, 0 rows affected (0.00 sec)

mysql> call swap(@a, @b);
+------+------+
| num1 | num2 |
+------+------+
|   10 |   20 |
+------+------+
1 row in set (0.02 sec)

+------+------+
| num1 | num2 |
+------+------+
|   20 |   10 |
+------+------+
1 row in set (0.03 sec)

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

CONCLUSION:

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

Exp. No. 12 **Date: 05/12/2022**

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 is having values from 1 to 10 and corresponding message is either even or odd.

```
delimiter $$
drop procedure if exists evenodd$$
create procedure evenodd()
begin
	declare i int default 1;
	drop table if exists TEMP01;
	create table TEMP01(num int, message varchar(6));
	desc TEMP01;
	l1: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;
	select * from TEMP01;
end$$
delimiter ;
```

```
mysql> source evenodd.sql;
Query OK, 0 rows affected, 1 warning (0.44 sec)

Query OK, 0 rows affected (0.80 sec)

mysql> call evenodd();
+---------+------------+------+-----+---------+-------+
| Field   | Type       | Null | Key | Default | Extra |
+---------+------------+------+-----+---------+-------+
| num     | int        | YES  |     | NULL    |       |
| message | varchar(6) | YES  |     | NULL    |       |
+---------+------------+------+-----+---------+-------+
2 rows in set (2.00 sec)

+------+---------+
| 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 (3.33 sec)

Query OK, 0 rows affected (3.33 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

```
delimiter $$
drop procedure if exists employee$$
create procedure employee(empno varchar(5))
begin
    declare salary, bp, d, hr int default 0;
    select Basic_Pay, DA, HRA into bp, d, hr from Employee where SSN=empno;
    set salary=(bp+d+hr)*12;
    if salary<250000 then
        select 0 as tax;
    elseif salary<500000 then
        select 0.05*salary as tax;
    elseif salary<750000 then
        select 0.1*salary as tax;
    elseif salary<1000000 then
        select 0.15*salary as tax;
    elseif salary<1250000 then
        select 0.2*salary as tax;
    elseif salary<1500000 then
        select 0.25*salary as tax;
    else
```

```
            select 0.3*salary as tax;
    end if;
end$$
delimiter ;
```

```
mysql> source employee.sql;
Query OK, 0 rows affected (0.20 sec)

Query OK, 0 rows affected (0.45 sec)

mysql> call employee('SL21');
+---------+
| tax     |
+---------+
| 67680.0 |
+---------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call employee('SA9');
+-----+
| tax |
+-----+
|   0 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 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(empno varchar(5))
begin
    declare bp int;
    select Basic_Pay into bp from Employee where SSN=empno;
    if bp>30000 then
            select bp+500 as salary;
    else
            select bp as salary;
    end if;
end$$
delimiter ;
```

```
mysql> source salary.sql;
Query OK, 0 rows affected (0.24 sec)

Query OK, 0 rows affected (0.42 sec)

mysql> call salary('SA9');
+--------+
| salary |
+--------+
|   9000 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call salary('SG5');
+--------+
| salary |
+--------+
|  35500 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

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

```
delimiter $$
drop procedure if exists bonus$$
create procedure bonus(empno varchar(5))
begin
        declare expe, b, pay int default 0;
        declare empname varchar(15);
        desc Bonus;
        select Years_of_Exp, Basic_Pay, Name into expe, pay, empname from Employee where
SSN=empno;
        if (expe<5) then
                set b=pay;
        elseif (expe<=9) then
                set b=.2*pay*12;
        else
                set b=pay+(.25*pay*12);
        end if;
        insert into Bonus values(empno, empname, b);
```

```
    select * from Bonus;
end$$
delimiter ;
```

```
mysql> source bonus.sql;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call bonus('SA9');
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| SSN   | varchar(5)  | YES  |     | NULL    |       |
| Name  | varchar(15) | YES  |     | NULL    |       |
| Bonus | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

+------+-----------+-------+
| SSN  | Name      | Bonus |
+------+-----------+-------+
| SA9  | Mary Howe | 21600 |
+------+-----------+-------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call bonus('SL21');
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| SSN   | varchar(5)  | YES  |     | NULL    |       |
| Name  | varchar(15) | YES  |     | NULL    |       |
| Bonus | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

+------+------------+-------+
| SSN  | Name       | Bonus |
+------+------------+-------+
| SA9  | Mary Howe  | 21600 |
| SL21 | John White | 30000 |
+------+------------+-------+
2 rows in set (0.01 sec)
```

```
mysql> call bonus('SG5');
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| SSN   | varchar(5)  | YES  |     | NULL    |       |
| Name  | varchar(15) | YES  |     | NULL    |       |
| Bonus | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

+------+-------------+--------+
| SSN  | Name        | Bonus  |
+------+-------------+--------+
| SA9  | Mary Howe   |  21600 |
| SL21 | John White  |  30000 |
| SG5  | Susan Brand | 140000 |
+------+-------------+--------+
3 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 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.

```
delimiter $$
drop procedure if exists account$$
create procedure account(accno int, amt decimal)
begin
	declare bal int default 0;
	select * from account_master;
	select balance into bal from account_master where acct_no=accno;
	if (amt<=bal) then
		select 'Transaction Successful' as Message;
		update account_master set balance=bal-amt where acct_no=accno;
		select * from account_master;
	else
		select 'Transaction Failed, Insufficient Balance' as Message;
	end if;
end$$
delimiter ;
```

```
mysql> source account.sql;
Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call account(720937, 100000);
+---------+---------------+---------+
| acct_no | customer_name | balance |
+---------+---------------+---------+
|  720937 | Tina Murphy   |  200000 |
|  264828 | Joe Keogh     |  350000 |
|  563836 | Carol Farrel  |  600000 |
|  104729 | Tony Shaw     |  150000 |
+---------+---------------+---------+
4 rows in set (0.00 sec)

+------------------------+
| Message                |
+------------------------+
| Transaction Successful |
+------------------------+
1 row in set (0.00 sec)

+---------+---------------+---------+
| acct_no | customer_name | balance |
+---------+---------------+---------+
|  720937 | Tina Murphy   |  100000 |
|  264828 | Joe Keogh     |  350000 |
|  563836 | Carol Farrel  |  600000 |
|  104729 | Tony Shaw     |  150000 |
+---------+---------------+---------+
4 rows in set (0.00 sec)

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

```
mysql> call account(720937, 200000);
+---------+---------------+---------+
| acct_no | customer_name | balance |
+---------+---------------+---------+
|  720937 | Tina Murphy   |  100000 |
|  264828 | Joe Keogh     |  350000 |
|  563836 | Carol Farrel  |  600000 |
|  104729 | Tony Shaw     |  150000 |
+---------+---------------+---------+
4 rows in set (0.01 sec)

+-----------------------------------------+
| Message                                 |
+-----------------------------------------+
| Transaction Failed, Insufficient Balance |
+-----------------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

CONCLUSION:

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

Exp. No. 13 **Date: 16/12/2022**

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 **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].

```
delimiter $$
drop procedure if exists marks$$
create procedure marks()
begin
	declare studid, m1, m2, m3, m4, tot, flag int default 0;
	declare curs1 cursor for select RNO from STUDENT;
	declare continue handler for not found set flag=1;
	open curs1;
	select * from STUDENT;
	c1:loop
		fetch curs1 into studid;
		select S1, S2, S3, S4 into m1, m2, m3, m4 from STUDENT where RNO=studid;
		set tot=m1+m2+m3+m4;
		update STUDENT set total=tot, percentage=tot/4 where RNO=studid;
		if (flag=1) then
			leave c1;
		end if;
	end loop;
	close curs1;
	select * from STUDENT;
end$$
```

delimiter ;

```
mysql> source marks.sql;
Query OK, 0 rows affected (0.22 sec)

Query OK, 0 rows affected (0.32 sec)

mysql> call marks();
+------+------+------+------+------+-------+------------+
| RNO  | S1   | S2   | S3   | S4   | total | percentage |
+------+------+------+------+------+-------+------------+
|    1 |   95 |   85 |   75 |   65 |  NULL |       NULL |
|    2 |   79 |   80 |   78 |   98 |  NULL |       NULL |
|    3 |   81 |   68 |   59 |   65 |  NULL |       NULL |
|    4 |   97 |   93 |   95 |   99 |  NULL |       NULL |
|    5 |   76 |   82 |   77 |   83 |  NULL |       NULL |
+------+------+------+------+------+-------+------------+
5 rows in set (0.00 sec)

+------+------+------+------+------+-------+------------+
| RNO  | S1   | S2   | S3   | S4   | total | percentage |
+------+------+------+------+------+-------+------------+
|    1 |   95 |   85 |   75 |   65 |   320 |         80 |
|    2 |   79 |   80 |   78 |   98 |   335 |         84 |
|    3 |   81 |   68 |   59 |   65 |   273 |         68 |
|    4 |   97 |   93 |   95 |   99 |   384 |         96 |
|    5 |   76 |   82 |   77 |   83 |   318 |         80 |
+------+------+------+------+------+-------+------------+
5 rows in set (0.85 sec)

Query OK, 0 rows affected (0.85 sec)
```

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

```
delimiter $$
drop function if exists fact$$
create function fact(n int)
        returns int
deterministic
begin
        declare fact int default 1;
        declare counter int;
        set counter=n;
        factorial:repeat
                set fact=fact*counter;
                set counter=counter-1;
                until counter=1
        end repeat;
        return fact;
end$$
delimiter ;
```

```
mysql> source fact.sql;
Query OK, 0 rows affected, 1 warning (0.04 sec)

Query OK, 0 rows affected (1.39 sec)

mysql> select fact(5);
+---------+
| fact(5) |
+---------+
|     120 |
+---------+
1 row in set (0.17 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].

```
delimiter $$
drop function if exists deptsal$$
create function deptsal(id varchar(5))
	returns int
deterministic
begin
	declare tot int default 0;
	select sum(SALARY) into tot from WORK where DEPT_NO=id;
	return tot;
end$$
delimiter ;
```

```
mysql> source totsal.sql;
Query OK, 0 rows affected (0.82 sec)

Query OK, 0 rows affected (0.50 sec)

mysql> select deptsal('D003');
+-----------------+
| deptsal('D003') |
+-----------------+
|           30360 |
+-----------------+
1 row in set (0.17 sec)

mysql> select deptsal('D007');
+-----------------+
| deptsal('D007') |
+-----------------+
|           35226 |
+-----------------+
1 row in set (0.00 sec)
```

4. Create a **trigger** that converts all department names into capital letters before inserting to 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)

```
delimiter $$
drop trigger if exists name$$
create trigger name before insert on department
for each row
```

```
begin
        set new.dept_name=ucase(new.dept_name);
end$$
delimiter ;
```

```
mysql> source name.sql;
Query OK, 0 rows affected (0.24 sec)

Query OK, 0 rows affected (0.15 sec)

mysql> insert into department values ('D009', 'project', 'Mumbai', 'E007');
Query OK, 1 row affected (0.40 sec)

mysql> select * from department;
+----------+-----------+----------+--------------+
| dept_num | dept_name | location | head_of_dept |
+----------+-----------+----------+--------------+
| D009     | PROJECT   | Mumbai   | E007         |
+----------+-----------+----------+--------------+
1 row in set (0.04 sec)

mysql> insert into department values ('D041', 'finance', 'Delhi', 'E005');
Query OK, 1 row affected (0.35 sec)

mysql> select * from department;
+----------+-----------+----------+--------------+
| dept_num | dept_name | location | head_of_dept |
+----------+-----------+----------+--------------+
| D009     | PROJECT   | Mumbai   | E007         |
| D041     | FINANCE   | Delhi    | E005         |
+----------+-----------+----------+--------------+
2 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)

```
delimiter $$
drop trigger if exists insert_trig$$
create trigger insert_trig after insert on Emp
for each row
begin
        insert into Emp_audit_log(emp_num, name, change_date, action) values(new.emp_num,
new.name, now(), 'insert');
end$$
drop trigger if exists update_trig$$
create trigger update_trig after update on Emp
for each row
begin
        insert into Emp_audit_log(emp_num, name, change_date, action) values(new.emp_num,
new.name, now(), 'update');
end$$
drop trigger if exists delete_trig$$
```

```
create trigger delete_trig after delete on Emp
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 ;
```

```
mysql> source log.sql;
Query OK, 0 rows affected (0.19 sec)

Query OK, 0 rows affected (0.25 sec)

Query OK, 0 rows affected (0.17 sec)

Query OK, 0 rows affected (0.25 sec)

Query OK, 0 rows affected (0.18 sec)

Query OK, 0 rows affected (0.18 sec)

mysql> insert into Emp values('E002', 'Mary Howe', '1970-02-19', 'Assistant');
Query OK, 1 row affected (0.28 sec)

mysql> insert into Emp values('E014', 'David Ford', '1958-03-24', 'Supervisor');
Query OK, 1 row affected (0.20 sec)

mysql> select * from Emp_audit_log;
+----------+---------+------------+-------------+--------+
| audit_id | emp_num | name       | change_date | action |
+----------+---------+------------+-------------+--------+
|        1 | E002    | Mary Howe  | 2022-12-16  | insert |
|        2 | E014    | David Ford | 2022-12-16  | insert |
+----------+---------+------------+-------------+--------+
2 rows in set (0.00 sec)

mysql> update Emp set designation='Manager' where emp_num='E002';
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Emp_audit_log;
+----------+---------+------------+-------------+--------+
| audit_id | emp_num | name       | change_date | action |
+----------+---------+------------+-------------+--------+
|        1 | E002    | Mary Howe  | 2022-12-16  | insert |
|        2 | E014    | David Ford | 2022-12-16  | insert |
|        3 | E002    | Mary Howe  | 2022-12-16  | update |
+----------+---------+------------+-------------+--------+
3 rows in set (0.00 sec)
```

```
mysql> delete from Emp where emp_num='E002';
Query OK, 1 row affected (0.14 sec)

mysql> select * from Emp_audit_log;
+----------+---------+------------+-------------+--------+
| audit_id | emp_num | name       | change_date | action |
+----------+---------+------------+-------------+--------+
|        1 | E002    | Mary Howe  | 2022-12-16  | insert |
|        2 | E014    | David Ford | 2022-12-16  | insert |
|        3 | E002    | Mary Howe  | 2022-12-16  | update |
|        4 | E002    | Mary Howe  | 2022-12-16  | delete |
+----------+---------+------------+-------------+--------+
4 rows in set (0.00 sec)
```

CONCLUSION:

Successfully practised creation of cursors, functions and triggers in MySQL

Exp. No. 14 **Date: 07/01/2023**

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 practiced CRUD operations in NoSQL (MongoDB)

Exp. No. 15 **Date: 10/01/2023**

Hotel Management System

AIM:

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

PROBLEM:

The Hotel Management System is a web-based application that allows the hotel manager to handle all hotel activities online.

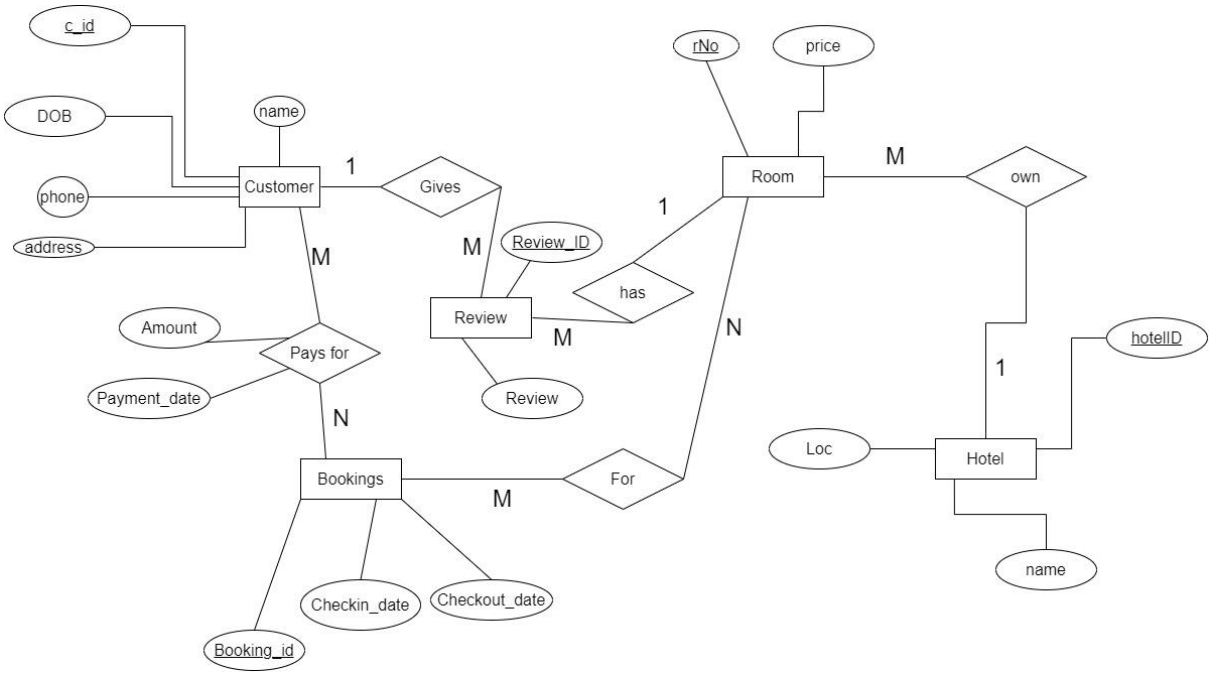
PROBLEM DESCRIPTION:

A web-based hotel management system is used for booking and cancellation of rooms. This system permits the clients globally to reserve the rooms in the hotel at any time. People face problem in reserving the hotels in the places where they want to visit. In that situation, this system can be of great relief to them. The user interface must be simple and easy to understand even by the common man. The admin has the right to update or add the hotel or the room information. The clients can cancel the already booked room in the particular hotel on a particular day. The clients can also give ratings and reviews for the room according to their stay.

TECHNOLOGIES USED:

- **Front End:** Vue.js
- **Back End:** Node.js, Express.js, Prisma
- **Database:** PostgreSQL

ER Diagram:



RELATIONAL SCHEMA:

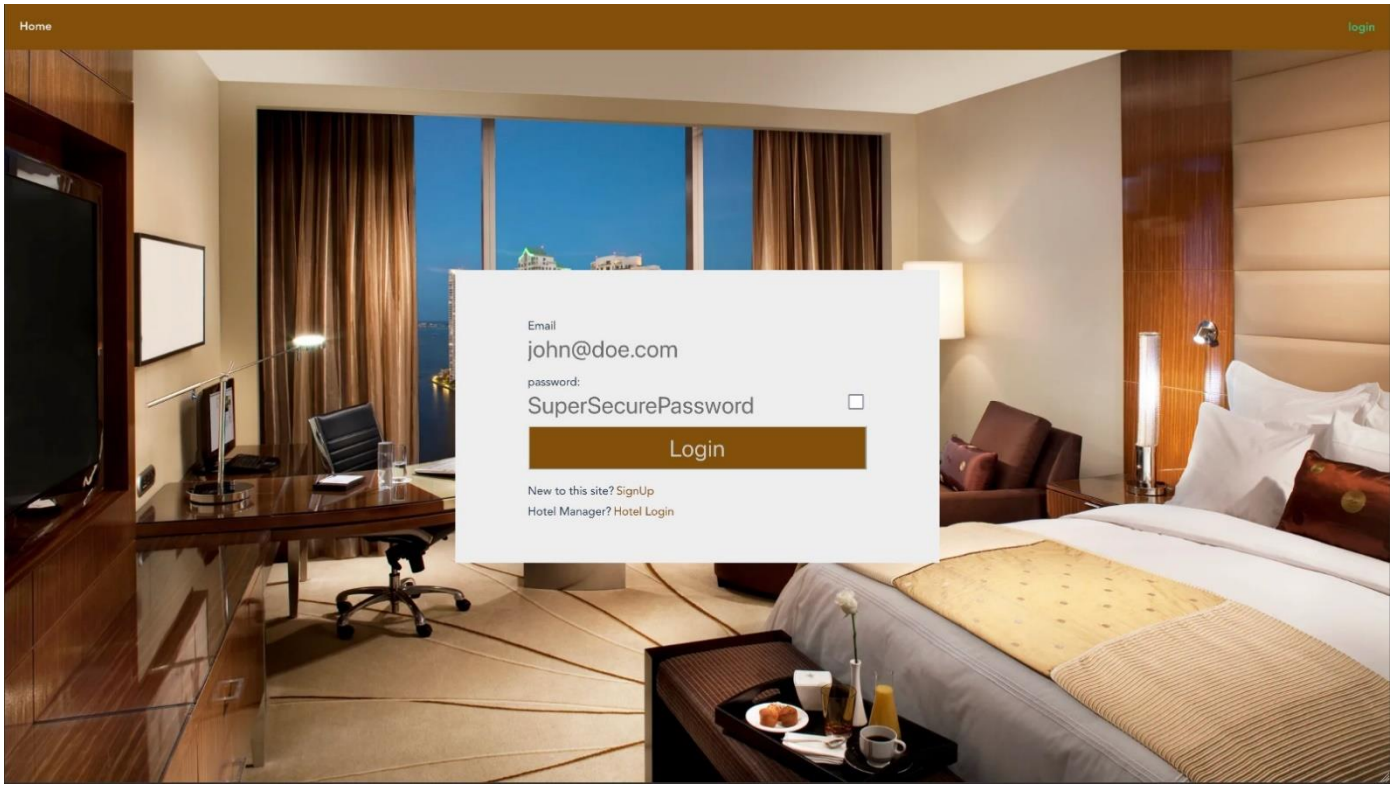
```
model Customer {
   customerID Int       @id @default(autoincrement())
   email      String
   name       String
   address    String
   DOB        DateTime
   password   String
   Review     Review[]
   Booking    Booking[]
}

model Booking {
   bookingID    Int      @id @default(autoincrement())
   checkinDate  DateTime
   checkoutDate DateTime
   billID       Int
   customerID   Int
   roomID       Int
   hotelID      Int
   hotel        Hotel    @relation(fields: [hotelID], references: [hotelID])
   room         Room     @relation(fields: [roomID], references: [roomID])
   bill         Bill     @relation(fields: [billID], references: [billID])
   customer     Customer @relation(fields: [customerID], references: [customerID])
}

model Bill {
   billID      Int       @id @default(autoincrement())
   amount      Int
   paymentDate DateTime
   Booking     Booking[]
}
```

```
model Hotel {
  hotelID  Int      @id @default(autoincrement())
  email    String
  name     String
  address  String
  password String
  Room     Room[]
  Images   Images[]
  Booking  Booking[]
}

model Room {
  roomID      Int      @id @default(autoincrement())
  description String
  price       Int
  type        String
  booked      Boolean
  hotelID     Int
  hotel       Hotel    @relation(fields: [hotelID], references: [hotelID])
  Review      Review[]
  Booking     Booking[]
  Images      Images[]
}

model Review {
  reviewID    Int      @id @default(autoincrement())
  review      Int
  title       String
  description String?
  customerID  Int
  roomID      Int
  customer    Customer @relation(fields: [customerID], references: [customerID])
  room        Room     @relation(fields: [roomID], references: [roomID])
```

```
}

model Images {
   imageId Int    @id @default(autoincrement())
   link    String
   roomID  Int
   hotelID Int
   hotel   Hotel  @relation(fields: [hotelID], references: [hotelID])
   room    Room   @relation(fields: [roomID], references: [roomID])
}
```

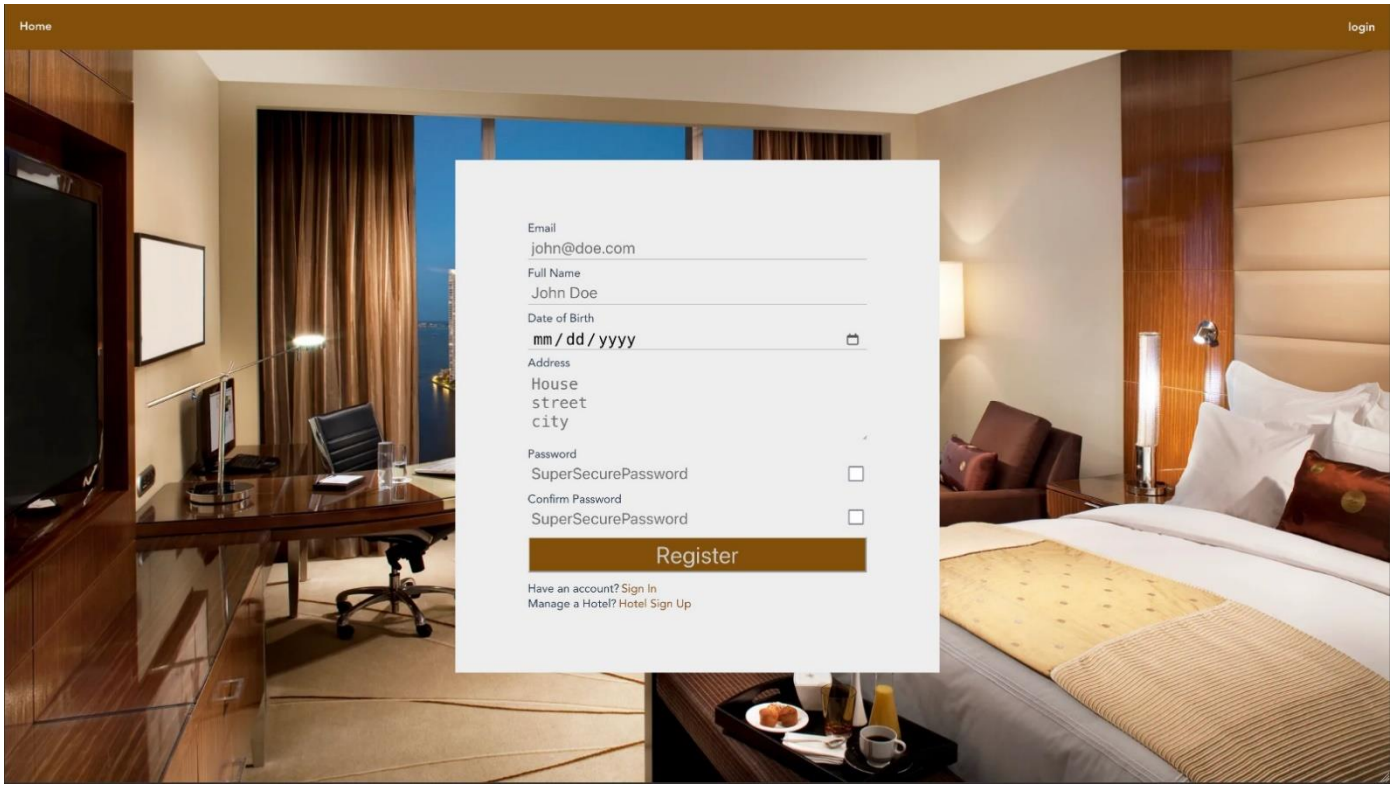
WEBSITE DEMO:

Login Page



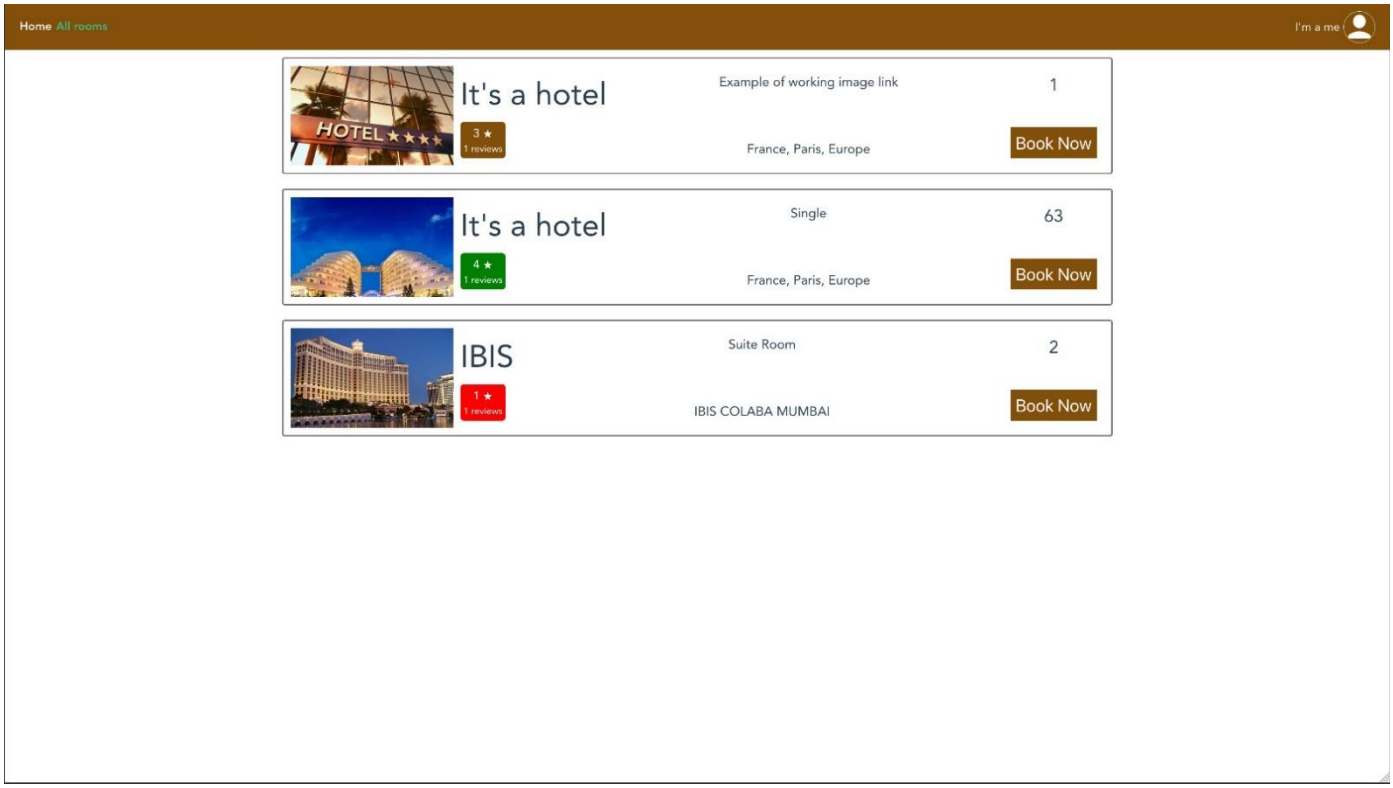
The application starts with the login page. Here, the already registered clients can login using their email d and password. For new clients there is a signup option which redirects the user to a registration page. This page also contains an option for manager login which redirects to Hotel Login page.

Registration Page



This page is designed for new clients who wants to register and create an account for themselves. This page allows users to create a password for themselves. This page can also redirect to hotel manager sign up.

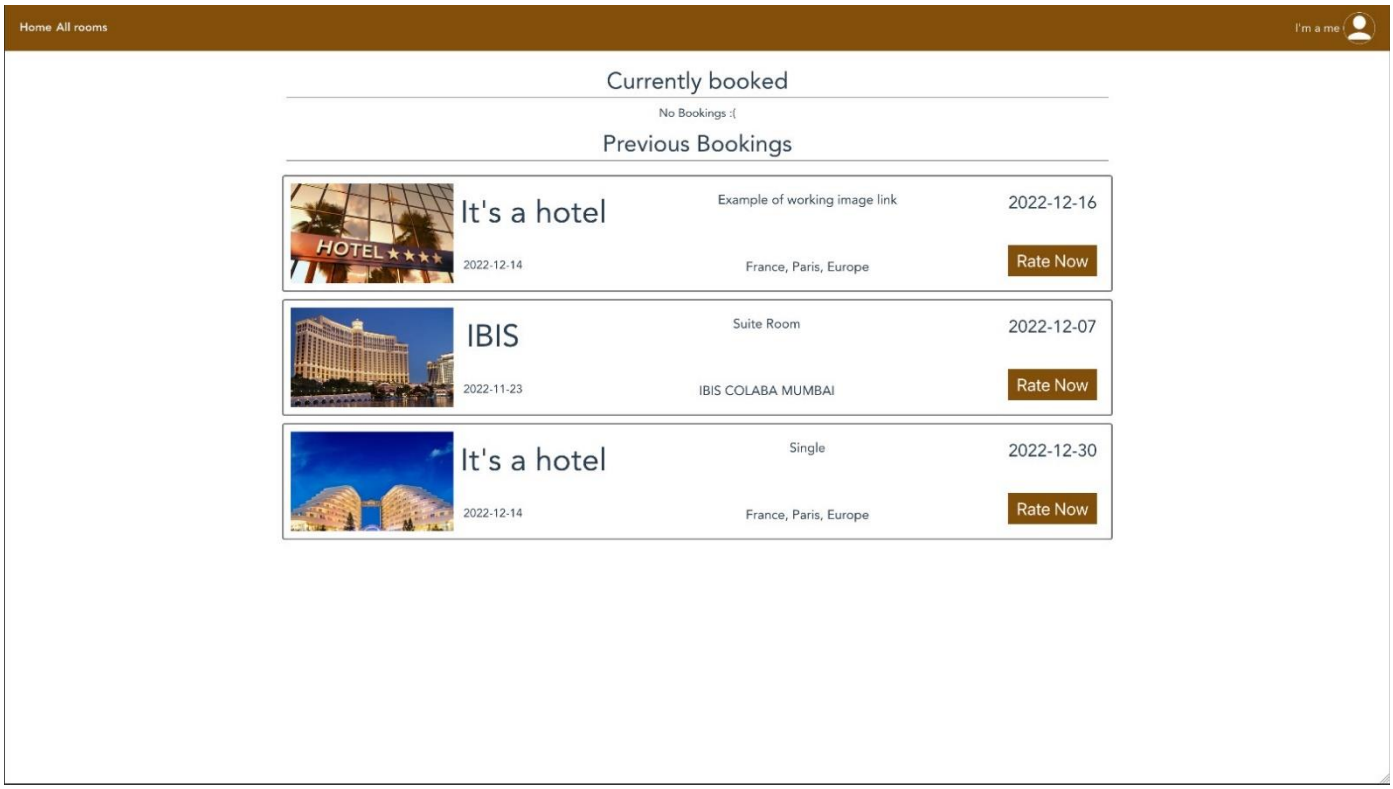
Hotel Viewing



This page gives the details of hotels including address, ratings, types of rooms available and original price of the room. This page also contains the image of the hotel and a Book Now option

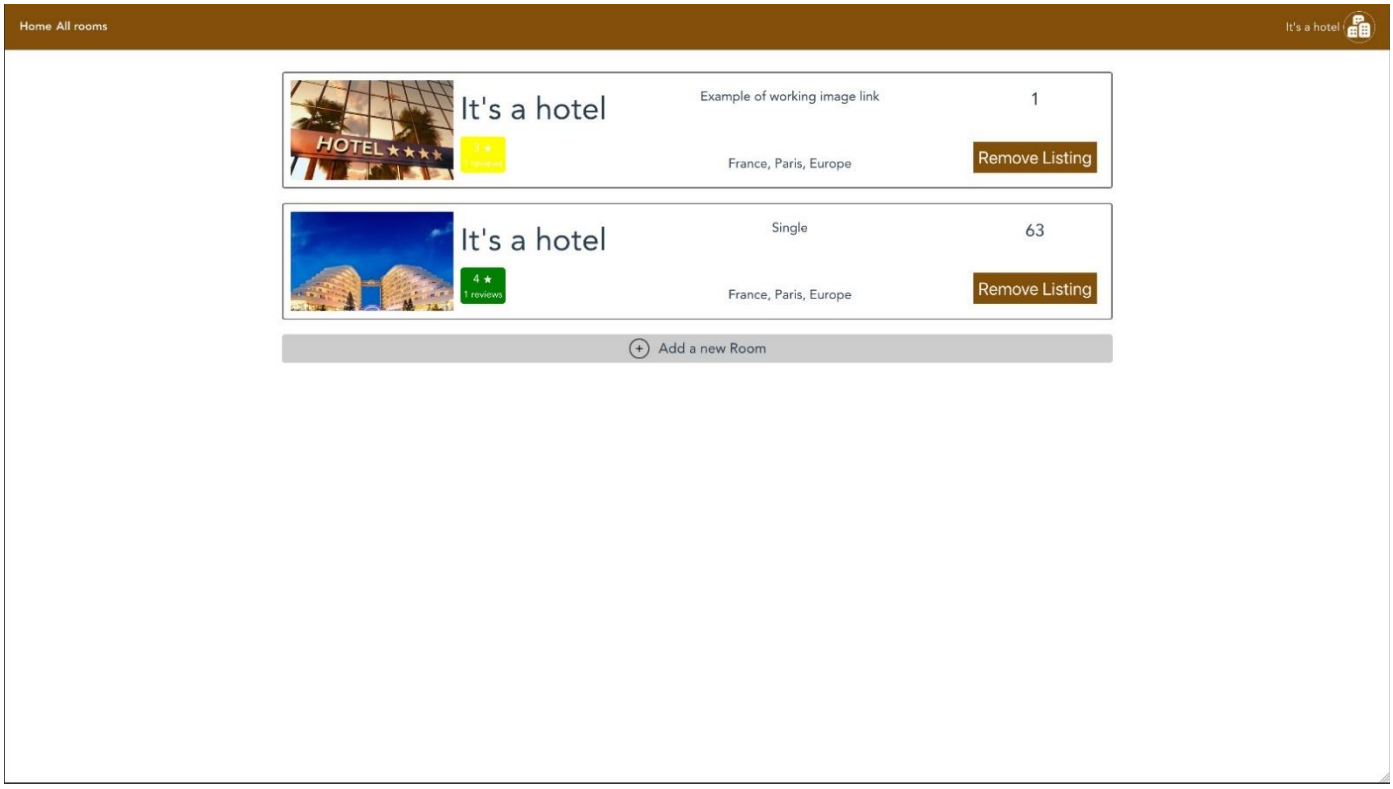
for clients.

Booking Viewing



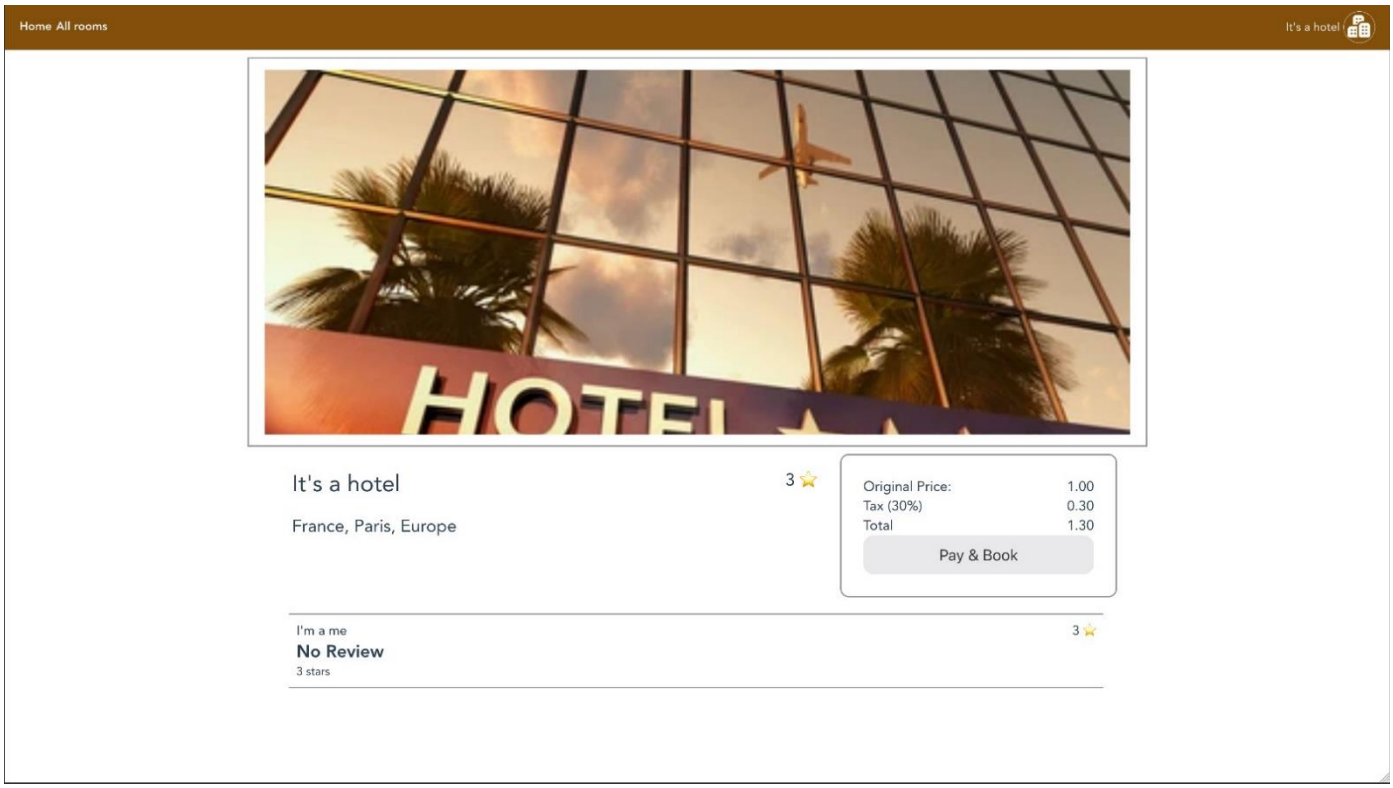
This page gives details of current booking and previous booking along with the date of booking. Clients can rate and provide review for their stay using Rate Now option.

Listed Rooms



This page lists the rooms available with hotel details and price. Rooms can be removed from the list and also new rooms can be added.

Hotel Detailed View



This page provides a detailed view of the hotel with images, locations, rating, reviews and total price of the room after adding tax. Clients can book the room for the required date using Pay & Book option.

CONCLUSION:

The application was created successfully and its working was verified.