DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSA05-DATABASE MANAGEMENT SYSTEMS

LAB MANUAL

DDL Commands – CREATE, ALTER, DROP

AIM:

To Create, Alter, Drop, Rename, Truncate using Data Definition Language (DDL) statements.

Description:

Data Definition Language (DDL) statements are used to define the database structure or schema.

DDL Commands: Create, Alter, Drop, Rename, Truncate

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- RENAME - rename an object

SYNTAX:

CREATE TABLE

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
);
```

ALTER A TABLE

To add a column in a table

```
ALTER TABLE table_name
ADD column_namedatatype;
```

To delete a column in a table

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

DROP TABLE

```
DROP TABLE table_name;
```

TRUNCATE TABLE

```
TRUNCATE TABLE table_name;
```

Questions:

1) Create a table name STUDENT with following structure.

#	**Column Name**	**Description**	**Data Type**
1	RegNo	Registration Number	NUMBER(3)
2	Name	Student Name	VARCHAR(15)
3	Gender	Gender of the student	CHAR(1)
4	DOB	Date of Birth	DATE
5	MobileNo	Mobile Number	NUMBER(10)
6	City	Location of stay	VARCHAR(15)

2) Create a table name FACULTY with following structure.

#	Column Name	Description	Data Type
1	FacNo	Faculty Identifier	VARCHAR(4)
2	FacName	Faculty Name	VARCHAR(15)
3	Gender	Gender of faculty	CHAR(1)
4	DOB	Date of Birth	DATE
5	DOJ	Date of Join	DATE
6	MobileNo	Mobile Number	NUMBER(10)

3) Create a table name DEPARTMENT with following structure.

#	Column Name	Description	Data Type
1	DeptNo	Department Identifier	VARCHAR(4)
2	DeptName	Department Name	VARCHAR(15)
3	DeptHead	Department Head	VARCHAR(4)

4) Create a table name COURSE with following structure.

#	Column Name	Description	Data Type
1	CourseNo	Course Identifier	VARCHAR(3)
2	CourseDesc	Course Description	VARCHAR(14)
3	CourseType	Course Type	CHAR(1)
4	SemNo	Semester Number	CHAR(1)
5	HallNo	Hall Number	VARCHAR(4)
6	FacNo	Faculty Identifier	VARCHAR(4)

5) Modify the table FACULTY by adding a column name Dept of datatype VARCHAR(10)

OUTPUTS:

1)

```
mysql> create table student(Regno int(3),Name char(15),gender char(1),Dob int(10),mobileno int(10),city char(10));
Query OK, 0 rows affected (0.14 sec)

mysql> desc student;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Regno    | int(3)   | YES  |     | NULL    |       |
| Name     | char(15) | YES  |     | NULL    |       |
| gender   | char(1)  | YES  |     | NULL    |       |
| Dob      | int(10)  | YES  |     | NULL    |       |
| mobileno | int(10)  | YES  |     | NULL    |       |
| city     | char(10) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

2)

```
mysql> create table faculty(Facno int(3),FacName char(15),gender char(1),Dob int(10),mobileno int(10),DOJ int(10));
Query OK, 0 rows affected (0.06 sec)

mysql> desc faculty;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Facno    | int(3)   | YES  |     | NULL    |       |
| FacName  | char(15) | YES  |     | NULL    |       |
| gender   | char(1)  | YES  |     | NULL    |       |
| Dob      | int(10)  | YES  |     | NULL    |       |
| mobileno | int(10)  | YES  |     | NULL    |       |
| DOJ      | int(10)  | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

3)

```
mysql> create table department(deptno int(10),deptname char(10),depthead char(10));
Query OK, 0 rows affected (0.11 sec)

mysql> desc department;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| deptno   | int(10)  | YES  |     | NULL    |       |
| deptname | char(10) | YES  |     | NULL    |       |
| depthead | char(10) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

4)

```
mysql> create table course(courseno int(3),coursedesc char(15),coursetype char(1),semno int(10),hallno int(10),Facno int(10));
Query OK, 0 rows affected (0.09 sec)

mysql> desc course;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| courseno   | int(3)   | YES  |     | NULL    |       |
| coursedesc | char(15) | YES  |     | NULL    |       |
| coursetype | char(1)  | YES  |     | NULL    |       |
| semno      | int(10)  | YES  |     | NULL    |       |
| hallno     | int(10)  | YES  |     | NULL    |       |
| Facno      | int(10)  | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

5)

```
mysql> alter table faculty add dept char(10);
Query OK, 2 rows affected (0.09 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> desc faculty;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Facno    | int(3)   | NO   | PRI | NULL    |       |
| FacName  | char(15) | YES  |     | NULL    |       |
| gender   | char(1)  | YES  |     | NULL    |       |
| Dob      | int(10)  | YES  |     | NULL    |       |
| mobileno | int(10)  | YES  |     | NULL    |       |
| DOJ      | int(10)  | YES  |     | NULL    |       |
| dept     | char(10) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

RESULT:

Tables are created, altered and modified using DDL commands.

DDL Commands with Constraints – PRIMARY, FOREIGN KEY, UNIQUE, CHECK

AIM:

To add the constraints like primary key, foreign key, unique key and check using DDL commands.

Description:

PRIMARY KEY:

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only one primary key, which may consist of single or multiple fields.

FOREIGN KEY:

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

UNIQUE Constraint:

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

CHECK Constraint:

- The CHECK constraint is used to limit the value range that can be placed in a column
- If you define a CHECK constraint on a single column it allows only certain values for this column.
- If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

PRIMARY:

```
ALTER TABLE table_name
ADD PRIMARY KEY(primary_key_column);
```

FOREIGN KEY:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
FOREIGN KEY foreign_key_name (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
```

UNIQUE:

```
CREATE TABLE table_1(
...
column_name_1 data_type,
...
UNIQUE(column_name_1)
);
```

CHECK

```
CREATE TABLE IF NOT EXISTS parts (
part_no VARCHAR(18) PRIMARY KEY,
description VARCHAR(40),
cost DECIMAL(10 , 2 ) NOT NULL CHECK(cost > 0), price DECIMAL (10,2) NOT
NULL
);
```

Questions:

1) Alter the table STUDENT2 with following structure.

#	Column Name	Constraints
1	RegNo	PRIMARY KEY

2) Alter the table name FACULTY with following structure.

#	Column Name	Constraints
1	FacNo	PRIMARY KEY
2	Gender	CHECK ‘M’ or ‘F’

3) Alter the table name DEPARTMENT with following structure.

#	Column Name	Constraints
1	DeptNo	PRIMARY KEY

4) Alter the table name COURSE with following structure.

#	Column Name	Constraints
1	CourseNo	PRIMARY KEY
2	SemNo	1 to 6

5) Alter the table name STUDENT2 with following structure.

#	Column Name	Constraints
1	S1Name	UNIQUE KEY

6) After the STUDENT2 & STUDENT3 tables are successfully created, test if you can add a constraint FOREIGN KEY to the RegNo of this table.

OUTPUTS:

1)

```
mysql> Alter table student2 ADD PRIMARY KEY(RegNo);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student2;
+----------+-------------+------+-----+---------+-------
| Field    | Type        | Null | Key | Default | Extra
+----------+-------------+------+-----+---------+-------
| RegNo    | int         | NO   | PRI | NULL    |
| S1Name   | varchar(15) | YES  |     | NULL    |
| Age      | char(2)     | YES  |     | NULL    |
| MobileNo | int         | YES  |     | NULL    |
| address  | varchar(15) | YES  |     | NULL    |
+----------+-------------+------+-----+---------+-------
5 rows in set (0.01 sec)
```

2)

```
mysql> alter table faculty add primary key(facno);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table faculty add check(gender='M'or'F');
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc faculty;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| Facno    | int(3)   | NO   | PRI | NULL    |       |
| FacName  | char(15) | YES  |     | NULL    |       |
| gender   | char(1)  | YES  |     | NULL    |       |
| Dob      | int(10)  | YES  |     | NULL    |       |
| mobileno | int(10)  | YES  |     | NULL    |       |
| DOJ      | int(10)  | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

3)

```
mysql> alter table department add primary key(deptno);
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> desc department;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| deptno   | int(10)  | NO   | PRI | NULL    |       |
| deptname | char(10) | YES  |     | NULL    |       |
| depthead | char(10) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

4)

```
mysql> alter table course add primary key(courseno);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table course add check(semno>=1&&semno<=6);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc course;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| courseno   | int(3)   | NO   | PRI | NULL    |       |
| coursedesc | char(15) | YES  |     | NULL    |       |
| coursetype | char(1)  | YES  |     | NULL    |       |
| semno      | int(10)  | YES  |     | NULL    |       |
| hallno     | int(10)  | YES  |     | NULL    |       |
| Facno      | int(10)  | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

5)

```
mysql> Alter table student2 ADD UNIQUE KEY(S1Name);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student2;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| RegNo    | int         | NO   | PRI | NULL    |       |
| S1Name   | varchar(15) | YES  | UNI | NULL    |       |
| Age      | char(2)     | YES  |     | NULL    |       |
| MobileNo | int         | YES  |     | NULL    |       |
| address  | varchar(15) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

6)

```
mysql> ALTER TABLE student1 modify address int not null;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc student1;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| RegNo   | int         | NO   | PRI | NULL    |       |
| S1Name  | varchar(15) | YES  |     | NULL    |       |
| Age     | char(2)     | YES  | UNI | NULL    |       |
| DOB     | int         | YES  |     | NULL    |       |
| address | int         | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Result:

DDL Commands with Primary, Foreign, Unique, Check constraints are updated and verified.

Ex.No. : 3

Date:

DML Commands – INSERT, SELECT

Aim:

To perform Data Manipulation Language (DML) Commands such as INSERT, SELECT in the table.

Description:

Data Manipulation Language (DML) statements are used for managing data within schema objects. DML Commands: Insert, Select

- INSERT - insert data into a table
- SELECT - retrieve data from the a database

INSERT:

INSERT INTO table_name

VALUES (value1, value2, value3,...);

(or)

INSERT INTO table_name (column1, column2, column3,...)

VALUES (value1, value2, value3,...);

SELECT:

SELECT column_name(s)

FROM table_name;

Questions:

1. Populate all the five tables with your own data.
2. View all the records from the five tables.

OUTPUTS:

1)

```
mysql> insert into faculty values('1191151','mohan','m','2004-01-12','2010-05-23',990826973,'mtr','y'),(
'1f1922112','raju','m','2005-02-13','2009-06-25',90865894,'mgr','y'),('f1922113','ramesh','m','2005-03-1
4','2014-12-20',998263548,'mnr','y'),('f1922154','ramu','m','2006-04-14','2001-12-25',908269279,'mgr','n
'),('f1922115','san','m','2007-05-17','2016-09-20',998269245,'hyt','y');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select* from faculty;
+-----------+---------+--------+------------+------------+-----------+--------+---------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged |
+-----------+---------+--------+------------+------------+-----------+--------+---------+
| f1922113  | ramesh  | m      | 2005-03-14 | 2014-12-20 | 998263548 | mnr    | y       |
| f1922115  | san     | m      | 2007-05-17 | 2016-09-20 | 998269245 | hyt    | y       |
| f1922154  | ramu    | m      | 2006-04-14 | 2001-12-25 | 908269279 | mgr    | n       |
| 1191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |
| 1f1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |
+-----------+---------+--------+------------+------------+-----------+--------+---------+
5 rows in set (0.00 sec)

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

mysql> _
```

```
mysql> insert into student values(19221151,'mohan','m','2004-01-12',990826973,'mtr'),(19221152,'raju','m
','2005-02-13',90865894,'mgr'),(19221153,'ramesh','m','2005-03-14',998263548,'mnr'),(19221154,'ramu','m'
,'2006-04-14',908269279,'mgr'),(19221155,'san','m','2007-05-17',998269245,'hyt');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select* from student;
+-----------+--------+--------+------------+-----------+----------+
| Reg_no    | Name   | Gender | DOB        | Mobile_no | city     |
+-----------+--------+--------+------------+-----------+----------+
|         0 | NULL   | NULL   | NULL       |         0 | NULL     |
|  19221151 | mohan  | m      | 2004-01-12 | 990826973 | mtr      |
|  19221152 | raju   | m      | 2005-02-13 |  90865894 | mgr      |
|  19221153 | ramesh | m      | 2005-03-14 | 998263548 | mnr      |
|  19221154 | ramu   | m      | 2006-04-14 | 908269279 | mgr      |
|  19221155 | san    | m      | 2007-05-17 | 998269245 | hyt      |
| 192219184 | siri   | f      | 2004-01-12 | 891938534 | chittoor |
+-----------+--------+--------+------------+-----------+----------+
7 rows in set (0.00 sec)
```

2)

```
mysql> select*from course;
+----------+----------------------+------------+-------+--------+-------+
| courseno | coursedesc           | coursetype | semno | hallno | facno |
+----------+----------------------+------------+-------+--------+-------+
| CS101    | Computer Science 101 | manadtory  | 1     | H001   | F001  |
| CS201    | Computer Science 201 | manadtory  | 2     | H002   | F001  |
| MA101    | Mathematics 101      | manadtory  | 1     | H003   | F002  |
| MA201    | Mathematics 201      | elective   | 2     | H004   | F002  |
| PH101    | Physics 101          | L          | 1     | H005   | F003  |
+----------+----------------------+------------+-------+--------+-------+
5 rows in set (0.02 sec)
```

```
mysql> select* from department;
+--------+----------+----------+
| deptno | deptname | depthead |
+--------+----------+----------+
|      1 | wind     | raj      |
|      2 | war      | eir      |
|      3 | cooper   | lut      |
|      4 | fanna    | fayaz    |
+--------+----------+----------+
4 rows in set (0.00 sec)
```

RESULT:

Data Manipulation Language (DML) Commands such as INSERT, SELECT are performed in the five tables.

Ex.No. : 4

Date:

DML Commands with Constraints –UPDATE, DELETE

Aim:

To perform Data Manipulation Language (DML) Commands such as UPDATE, DELETE in the table.

Description:

Data Manipulation Language (DML) statements are used for managing data within schema objects. DML Commands: Update, Delete

- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain

UPDATE:

UPDATE table_name

SET column1=value, column2=value2,...

WHERE some_column=some_value;

DELETE:

DELETE FROM table_name

WHERE some_column=some_value;

Questions:

1. Update the value of student name whose register number is ‘191711342’
2. Delete the record in the table FACULTY, who resigned her job.
3. Modify the age for the faculty whose name is 'mohan' with a value ‘59’.
4. Remove all faculty who are having over 65 years

OUTPUTS:

1)

```
mysql> update student set Name='ram' where Reg_no=19221154;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select* from student;
+-----------+--------+--------+------------+-----------+----------+
| Reg_no    | Name   | Gender | DOB        | Mobile_no | city     |
+-----------+--------+--------+------------+-----------+----------+
|  19221151 | mohan  | m      | 2004-01-12 | 990826973 | mtr      |
|  19221152 | raju   | m      | 2005-02-13 |  90865894 | mgr      |
|  19221153 | ramesh | m      | 2005-03-14 | 998263548 | mnr      |
|  19221154 | ram    | m      | 2006-04-14 | 908269279 | mgr      |
|  19221155 | san    | m      | 2007-05-17 | 998269245 | hyt      |
| 192219184 | siri   | f      | 2004-01-12 | 891938534 | chittoor |
+-----------+--------+--------+------------+-----------+----------+
6 rows in set (0.00 sec)
```

2)

```
mysql> delete from faculty where resiged='n';
Query OK, 1 row affected (0.02 sec)

mysql> select* from faculty;
+-----------+---------+--------+------------+------------+-----------+--------+---------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged |
+-----------+---------+--------+------------+------------+-----------+--------+---------+
| f1922113  | ramesh  | m      | 2005-03-14 | 2014-12-20 | 998263548 | mnr    | y       |
| f1922115  | san     | m      | 2007-05-17 | 2016-09-20 | 998269245 | hyt    | y       |
| l191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |
| lf1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |
+-----------+---------+--------+------------+------------+-----------+--------+---------+
4 rows in set (0.00 sec)

mysql>
```

3)

```
mysql> update faculty set age=59 where facname='mohan';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select* from faculty;
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged | age  |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| f1922113  | ramesh  | m      | 2005-03-14 | 2014-12-20 | 998263548 | mnr    | y       |   45 |
| f1922115  | san     | m      | 2007-05-17 | 2016-09-20 | 998269245 | hyt    | y       |   68 |
| l191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |   59 |
| lf1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |   45 |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
4 rows in set (0.00 sec)
```

4)

```
mysql> delete from faculty where age>=65;
Query OK, 1 row affected (0.02 sec)

mysql> select* from faculty;
+-----------+---------+--------+------------+------------+-----------+--------+---------+-------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged | age   |
+-----------+---------+--------+------------+------------+-----------+--------+---------+-------+
| f1922113  | ramesh  | m      | 2005-03-14 | 2014-12-20 | 998263548 | mnr    | y       |    45 |
| l191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |    59 |
| lf1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |    45 |
+-----------+---------+--------+------------+------------+-----------+--------+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

RESULT:

Data Manipulation Language (DML) Commands such as UPDATE, DELETE are performed in the five tables.

Ex. No.: 5

Date:

SELECT with various clause – WHERE, pattern matching

AIM:

To view the records from the tables using SELECT commands with WHERE Clause and Pattern matching.

DESCRIPTION:

The SELECT statement allows you to get the data from tables. A table consists of rows and columns like a spreadsheet. Often, you want to see a subset rows, a subset of columns, or a combination of two. The result of the SELECT statement is called a result set that is a list of rows, each consisting of the same number of columns.

SELECT:

```
SELECTcolumn_1, column_2, ...
FROMtable_1
[INNER | LEFT |RIGHT] JOIN table_2 ON conditions
WHEREconditions

GROUP BY column_1

HAVING group_conditions

ORDER BY column_1
LIMIT offset, length;
```

The SELECT statement consists of several clauses as explained in the following list:

- SELECT followed by a list of comma-separated columns or an asterisk (*) to indicate that you want to return all columns.
- FROM specifies the table or view where you want to query the data.
- JOIN gets related data from other tables based on specific join conditions.
- WHERE clause filters row in the result set.
- GROUP BY clause groups a set of rows into groups and applies aggregate functions on each group.
- HAVING clause filters group based on groups defined by GROUP BY clause.
- ORDER BY clause specifies a list of columns for sorting.
- LIMIT constrains the number of returned rows.

LIKE:

- The LIKE operator is commonly used to select data based on patterns. Using the LIKE operator in the right way is essential to increase the query performance.
- The LIKE operator allows you to select data from a table based on a specified pattern. Therefore, the LIKE operator is often used in the WHERE clause of the SELECT statement.
- MySQL provides two wildcard characters for using with the LIKE operator, the percentage % and underscore _ .
- The percentage (%) wildcard allows you to match any string of zero or more characters.
- The underscore (_) wildcard allows you to match any single character.

Questions:

WHERE:

1. The student counselor wanted to display the registration number, student name and date of birth for all the students.
2. The controller of examinations wanted to list all the female students
3. List the Students who registered for the “C001” course.
4. Display all faculty details joined before “November 2014”
5. Display all the courses not allotted to hall ‘H001’

LIKE:

6. List the students whose name ends with the substring “sh”
7. Display all students whose name contains the substring “sh”
8. Find all the students who are located in cities having “Sal” as substring
9. Display the students whose names do not contain six letters.
10. Find all the students whose names contains “am”.

OUTPUTS:

1)

```
mysql> select Reg_no,Name,DOB from student ;
+-----------+--------+------------+
| Reg_no    | Name   | DOB        |
+-----------+--------+------------+
|  19221151 | mohan  | 2004-01-12 |
|  19221152 | raju   | 2005-02-13 |
|  19221153 | ramesh | 2005-03-14 |
|  19221154 | ram    | 2006-04-14 |
|  19221155 | san    | 2007-05-17 |
| 192219184 | siri   | 2004-01-12 |
+-----------+--------+------------+
6 rows in set (0.00 sec)
```

2)

```
mysql> select* from student where Gender='f';
+-----------+------+--------+------------+-----------+----------+
| Reg_no    | Name | Gender | DOB        | Mobile_no | city     |
+-----------+------+--------+------------+-----------+----------+
| 192219184 | siri | f      | 2004-01-12 | 891938534 | chittoor |
+-----------+------+--------+------------+-----------+----------+
1 row in set (0.00 sec)

mysql> _
```

3)

```
mysql>
mysql> select * from student where courseno="C001";
+-----------+----------+--------+------------+-----------+----------+----------+
| RegNo     | Name     | Gender | DOB        | MobileNo  | City     | courseno |
+-----------+----------+--------+------------+-----------+----------+----------+
| 1922211123 | RAKESH  | M      | 2004-12-15 | 987654329 | TIRUPATHI | C001    |
| 1922211125 | ROSY    | F      | 2004-08-24 | 987654323 | NELLORE   | C001    |
| 1922211198 | SRINIVAS | M     | 2004-06-17 | 986534256 | VIZAG     | C001    |
+-----------+----------+--------+------------+-----------+----------+----------+
3 rows in set (0.00 sec)

mysql> select * from faculty;
```

4)

```
mysql> select* from faculty where DOJ<'2014-11-01';
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged | age  |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| l191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |   59 |
| lf1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |   45 |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
2 rows in set (0.02 sec)
```

5)

```
mysql> select* from course where hallno!='H001';
+----------+----------------------+------------+-------+--------+-------+
| courseno | coursedesc           | coursetype | semno | hallno | facno |
+----------+----------------------+------------+-------+--------+-------+
| CS201    | Computer Science 201 | L          | 2     | H002   | F001  |
| MA101    | Mathematics 101      | L          | 1     | H003   | F002  |
| MA201    | Mathematics 201      | L          | 2     | H004   | F002  |
| PH101    | Physics 101          | L          | 1     | H005   | F003  |
+----------+----------------------+------------+-------+--------+-------+
4 rows in set (0.00 sec)
```

6)

```
mysql> select* from student where Name like '%sh';
+----------+--------+--------+------------+-----------+------+
| Reg_no   | Name   | Gender | DOB        | Mobile_no | city |
+----------+--------+--------+------------+-----------+------+
| 19221153 | ramesh | m      | 2005-03-14 | 998263548 | mnr  |
+----------+--------+--------+------------+-----------+------+
1 row in set (0.00 sec)
```

7)

```
ysql> select* from student where Name like '%sh%';
----------+--------+--------+------------+-----------+------+
 Reg_no   | Name   | Gender | DOB        | Mobile_no | city |
----------+--------+--------+------------+-----------+------+
 19221153 | ramesh | m      | 2005-03-14 | 998263548 | mnr  |
----------+--------+--------+------------+-----------+------+
 row in set (0.00 sec)
```

8)

```
mysql> SELECT * from student where City like "%SAL%";
+------------+------+--------+------------+-----------+-------+----------+
| RegNo      | Name | Gender | DOB        | MobileNo  | City  | courseno |
+------------+------+--------+------------+-----------+-------+----------+
| 1922211123 | PUMA | M      | 2004-12-15 | 987654329 | SALT  | C001     |
| 1922211125 | SUMA | F      | 2004-08-24 | 987654323 | BASAL | C001     |
+------------+------+--------+------------+-----------+-------+----------+
2 rows in set (0.00 sec)
```

9)

```
mysql> SELECT * from student where Name not like "_____";
+------------+---------+--------+------------+-----------+--------+----------+
| RegNo      | Name    | Gender | DOB        | MobileNo  | City   | courseno |
+------------+---------+--------+------------+-----------+--------+----------+
| 1922211123 | PUMA    | M      | 2004-12-15 | 987654329 | SALT   | C001     |
| 1922211125 | SUMA    | F      | 2004-08-24 | 987654323 | BASAL  | C001     |
| 1922211156 | RAMA    | F      | 2004-02-14 | 876543297 | KADAPA | C002     |
| 1922211198 | HEMANTH | M      | 2004-06-17 | 986534256 | VIZAG  | C001     |
+------------+---------+--------+------------+-----------+--------+----------+
4 rows in set (0.01 sec)
```

10)

```
mysql> select* from student where Name like '%am%';
+----------+--------+--------+------------+-----------+------+
| Reg_no   | Name   | Gender | DOB        | Mobile_no | city |
+----------+--------+--------+------------+-----------+------+
| 19221153 | ramesh | m      | 2005-03-14 | 998263548 | mnr  |
| 19221154 | ram    | m      | 2006-04-14 | 908269279 | mgr  |
+----------+--------+--------+------------+-----------+------+
2 rows in set (0.00 sec)
```

RESULT:

The records from the tables are displayed using SELECT commands with WHERE Clause and Pattern matching.

Ex. No. : 6

Date:

SELECT with various clause – BETWEEN, IN, Aggregate function

AIM:

To view the records from the tables using SELECT commands with BETWEEN, IN, Aggregate functions.

DESCRIPTION:

- The BETWEEN operator allows you to specify a range to test. We often use the BETWEEN operator in the WHERE clause of the SELECT, INSERT, UPDATE, and DELETE statements.
- The IN operator allows you to determine if a specified value matches any one of a list or a sub query.
- MySQL provides many aggregate functions that include AVG, COUNT, SUM, MIN, MAX, etc. An aggregate function ignores NULL values when it performs calculation except for the COUNT function.

BETWEEN operator:

```
SELECTcolumn1,column2,...
FROMtable_name
WHERE expr [NOT] BETWEEN begin_expr AND end_expr;
```

The *expr* is the expression to test in the range that is defined by *begin_expr* and *end_expr*.

IN operator:

```
SELECTcolumn1,column2,...
FROMtable_name
WHERE (expr|column_1) IN ('value1','value2',...);
```

Questions:

IN & BETWEEN

1. List the type of the courses “Statistics” and “Programming”
2. The instructor wants to know the CourseNos whose scores are in the range 50 to 80

AGGREGATE

1. Find the average mark of “C002”.
2. List the maximum, minimum mark for “C002”
3. List the maximum, minimum, average mark for each course
4. List the name of the courses and average mark of each course.
5. Calculate the sum of all the scores.
6. How many students are registered for each course? Display the course description and the number of students registered in each course.
7. How many courses did each student register for?

OUTPUTS:

IN & BETWEEN

1)

```
mysql> SELECT coursetype FROM Course WHERE CourseDesc IN ('Computer Science 101', 'Mathematics 201');
+------------+
| coursetype |
+------------+
| manadtory  |
| elective   |
+------------+
2 rows in set (0.00 sec)

mysql>
```

2)

```
mysql> select CourseNo from StudentScores where score between 50 and 80;
+----------+
| CourseNo |
+----------+
| C001     |
| C001     |
| C002     |
| C002     |
+----------+
4 rows in set (0.00 sec)
```

AGGREGATE

1)

```
mysql> select avg(Score) from StudentScores where CourseNo='C002';
+------------+
| avg(Score) |
+------------+
|    75.0000 |
+------------+
1 row in set (0.00 sec)
```

2)

```
mysql> select max(Score),MIN(Score) from StudentScores where CourseNo='C002';
+------------+------------+
| max(Score) | MIN(Score) |
+------------+------------+
|         85 |         65 |
+------------+------------+
1 row in set (0.02 sec)
```

3)

```
mysql> select max(Score),min(Score),avg(Score) from StudentScores group by coursename ;
+------------+------------+------------+
| max(Score) | min(Score) | avg(Score) |
+------------+------------+------------+
|         90 |         70 |    80.0000 |
|         85 |         65 |    75.0000 |
+------------+------------+------------+
2 rows in set (0.02 sec)
```

4)

```
mysql> select coursename,avg(score) from StudentScores group by coursename;
+-------------+------------+
| coursename  | avg(score) |
+-------------+------------+
| computersci |    80.0000 |
| probability |    75.0000 |
+-------------+------------+
2 rows in set (0.02 sec)
```

5)

```
mysql> select sum(score) from StudentScores;
+------------+
| sum(score) |
+------------+
|        465 |
+------------+
1 row in set (0.00 sec)
```

6)

```
mysql> select coursedesc,count(StudentNo) from StudentScores group by coursedesc;
+------------+------------------+
| coursedesc | count(StudentNo) |
+------------+------------------+
| Cse        |                3 |
| mat        |                3 |
+------------+------------------+
2 rows in set (0.00 sec)

mysql> |
```

7)

```
mysql> select StudentNo,count(coursename) from StudentScores group by StudentNo;
+-----------+-------------------+
| StudentNo | count(coursename) |
+-----------+-------------------+
| S001      |                 2 |
| S002      |                 2 |
| S003      |                 2 |
+-----------+-------------------+
3 rows in set (0.00 sec)
```

RESULT:

The records from the tables are displayed using SELECT commands with WHERE Clause and Pattern matching.

Ex. No.: 7

Date:

SELECT with various clause – GROUP BY, HAVING, ORDER BY

AIM:

To view the records from the tables using SELECT commands with Group By, Having, Order By

DESCRIPTION:

GROUP BY – HAVING:

- The GROUP BY clause groups a set of rows into a set of summary rows by values of columns or expressions. The GROUP BY clause returns one row for each group. In other words, it reduces the number of rows in the result set.
- The GROUP BY clause is used with aggregate functions such as SUM, AVG, MAX, MIN, and COUNT. The aggregate function that appears in the SELECT clause provides the information about each group.
- The GROUP BY clause is an optional clause of the SELECT statement.
- To filter the groups returned by GROUP BY clause, you use a HAVING clause.

ORDER BY:

When you use the SELECT statement to query data from a table, the result set is not sorted in any orders. To sort the result set, you use the ORDER BY clause. The ORDER BY clause allows you to:

- Sort a result set by a single column or multiple columns.
- Sort a result set by different columns in ascending or descending order.

SYNTAX:

GROUP BY – HAVING:

SELECTc1, c2,...,cn, aggregate_function(ci)
FROMtable
WHEREwhere_conditions
GROUP BYc1 , c2,...,cn
HAVINGconditionS

ORDER BY:

SELECT column1, column2,...
FROMtbl
ORDER BY column1 [**ASC**|**DESC**], column2 [**ASC**|**DESC**],...

ASC stands for ascending and the DESC stands for descending. By default, the ORDER BY clause sorts the result set in ascending order if you don't specify ASC or DESC explicitly.

Questions:

GROUP BY - HAVING

1. How many students are registered for each course? Display the course description and the number of students registered in each course.
2. How many courses did each student register for?

ORDER BY

1. Retrieve Name, Gender, Mobile No of all the students in ascending order of Reg No.
2. List the faculty members in the order of older faculty first.

OUTPUTS:

1)

```
mysql> select coursedesc,count(StudentNo) from StudentScores group by coursedesc;
+------------+------------------+
| coursedesc | count(StudentNo) |
+------------+------------------+
| Cse        |                3 |
| mat        |                3 |
+------------+------------------+
2 rows in set (0.00 sec)
```

2)

```
mysql> select StudentNo,count(coursename) from StudentScores group by StudentNo;
+-----------+------------------+
| StudentNo | count(coursename) |
+-----------+------------------+
| S001      |                2 |
| S002      |                2 |
| S003      |                2 |
+-----------+------------------+
3 rows in set (0.00 sec)
```

3)

```
mysql> select Name,Gender,Mobile_no from student order by Reg_no;
+--------+--------+-----------+
| Name   | Gender | Mobile_no |
+--------+--------+-----------+
| mohan  | m      | 990826973 |
| raju   | m      |  90865894 |
| ramesh | m      | 998263548 |
| ram    | m      | 908269279 |
| san    | m      | 998269245 |
| siri   | f      | 891938534 |
+--------+--------+-----------+
6 rows in set (0.00 sec)
```

4)

```
mysql> select* from faculty order by DOJ;
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| facno     | facname | Gender | DOB        | DOJ        | Mobileno  | deptno | resiged | age  |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
| lf1922112 | raju    | m      | 2005-02-13 | 2009-06-25 |  90865894 | mgr    | y       |   45 |
| l191151   | mohan   | m      | 2004-01-12 | 2010-05-23 | 990826973 | mtr    | y       |   59 |
| f1922113  | ramesh  | m      | 2005-03-14 | 2014-12-20 | 998263548 | mnr    | y       |   45 |
+-----------+---------+--------+------------+------------+-----------+--------+---------+------+
3 rows in set (0.00 sec)
```

RESULT:

The records from the tables are displayed using SELECT commands with GROUP BY, HAVING and ORDER BY.

Ex. No.: 8

Date:

Query with Sub Query& Correlated Query

AIM:

To perform subquery and correlated query on the given relations.

DESCRIPTION:

SUBQUERY

A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. In addition, a MySQL subquery can be nested inside another subquery.

A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query. A subquery can be used anywhere that expression is used and must be closed in parentheses.

CORRELATED QUERY:

A correlated subquery is a subquery that uses the data from the outer query. In other words, a correlated subquery depends on the outer query. A correlated subquery is evaluated once for each row in the outer query.

SYNTAX:

SUBQUERY:

```
SELECTc1, c2,...,cn
FROMtable
WHEREc1 IN (SELECTc1, c2,...,cn
            FROMtable
            WHEREwhere_conditions);
```

CORRELATED QUERY:

```
SELECT*
FROMtable_name
WHEREEXISTS (subquery );
```

Questions:

Sub-Query and Correlated Sub-Query:

1. Which of the student's score is greater than the average score?
2. Which of the students' have written more than one assessment test?
3. Which faculty has joined recently and when?
4. List the course and score of assessments that have the value more than the average score each Course

OUTPUTS:

1) Which of the student's score is greater than the avg score?

```
mysql> select * from stud2 where marks>(select avg(marks) from stud2);
+-------+------+----------+-------+------------+-----------+
| regno | name | courseno | marks | facultydoj | assements |
+-------+------+----------+-------+------------+-----------+
|  1234 | nani | coo1     |    50 |       2001 |         5 |
|  1254 | ramu | coo1     |    50 |       2002 |         4 |
+-------+------+----------+-------+------------+-----------+
2 rows in set (0.00 sec)
```

2) Which of the students' have written more than one assessment test?

```
mysql> select name from stud2 where assements>1;
+------+
| name |
+------+
| nani |
| ramu |
| ravi |
+------+
3 rows in set (0.00 sec)
```

3) Which faculty has joined recently and when?

```
mysql> select * from faculty order by doj limit 1;
+-------+------------+--------+------------+------------+------------+--------+
| FacNo | FacultyName | gender | Dob        | Doj        | Mobileno   | DeptNo |
+-------+------------+--------+------------+------------+------------+--------+
| fo1   | chaithu    | m      | 0000-00-00 | 0000-00-00 | 2147483647 | cse    |
+-------+------------+--------+------------+------------+------------+--------+
1 row in set (0.00 sec)
```

4) List the course and score of assessments that have the value more than the average score each Course

```
mysql> select courseno,marks from stud2 where marks>(select avg(marks) from stud2) order by courseno;
+----------+-------+
| courseno | marks |
+----------+-------+
| coo1     |    50 |
| coo1     |    50 |
+----------+-------+
2 rows in set (0.00 sec)
```

RESULT:

The records from the tables are displayed using Sub-Query and Correlated Sub-Query.

Ex. No.: 8

Date:

Query with Joins – EquiJoin, InnerJoin, OuterJoin

AIM: To perform JOIN using Equi Join, Inner Join, Outer Join on the given relation.

DESCRIPTION:

JOIN

A MySQL join is a method of linking data from one or more table based on values of the common column between tables.

MySQL supports the following types of joins:

1. Cross join
2. Inner join
3. Left join
4. Right join

CROSS JOIN

The CROSS JOIN makes a Cartesian product of rows from multiple tables. Suppose, you join t1 and t2 tables using the CROSS JOIN, the result set will include the combinations of rows from the t1 table with the rows in the t2 table.

INNER JOIN

To join two tables, the INNER JOIN compares each row in the first table with each row in the second table to find pairs of rows that satisfy the join-predicate. Whenever the join-predicate is satisfied by matching non-NULL values, column values for each matched pair of rows of the two tables are included in the result set.

LEFT JOIN

Unlike an INNER JOIN, a LEFT JOIN returns all rows in the left table including rows that satisfy join-predicate and rows do not. For the rows that do not match the join-predicate, NULLs appear in the columns of the right table in the result set.

RIGHT JOIN

A RIGHT JOIN is similar to the LEFT JOIN except that the treatment of tables is reversed. With a RIGHT JOIN, every row from the right table (t2) will appear in the result set. For the rows in the right table that do not have the matching rows in the left table (t1), NULLs appear for columns in the left table (t1).

SYNTAX:

CROSS JOIN:

SELECTt1.id, t2.id

FROMt1**CROSS JOIN** t2;

INNER JOIN:

SELECTt1.id, t2.id

FROMt1**INNER JOIN**t2 **ON** t1.pattern **=** t2.pattern;

LEFT JOIN:

SELECTt1.id, t2.id

FROMt1**LEFT JOIN**t2 **ON** t1.pattern = t2.pattern

ORDER BY t1.id;

RIGHT JOIN:

SELECTt1.id, t2.id

FROMt1**RIGHT JOIN**t2 **ON** t1.pattern = t2.pattern

ORDER BYt2.id;

Questions:

1. List the departments where the faculty members are working.
2. Find the student who has no score in any of the courses. List student name and course number.
3. The office clerk needs the names of the courses taken by the faculty belonging to ‘ECE department’ whose name is ‘Kamal’

OUTPUTS:

1) List the departments where the faculty members are working.

```
mysql> select faculty.facno,faculty.facname,department.deptno,department.deptname from faculty cross join department;
+-------+---------+--------+----------+
| facno | facname | deptno | deptname |
+-------+---------+--------+----------+
| 802   | Ratnam  | 11     | Sales    |
+-------+---------+--------+----------+
1 row in set (0.00 sec)
```

2) Find the student who has no score in any of the courses. List student name and course number.

```
mysql> select student.name,student.marks,course.courseno from student inner join course on student.course=course.courseno;
+--------+-------+----------+
| name   | marks | courseno |
+--------+-------+----------+
| Ramu   |     0 | C00      |
| Geetha |     0 | C00      |
| Pooja  |     0 | C00      |
+--------+-------+----------+
3 rows in set (0.00 sec)
```

3) The office clerk needs the names of the courses taken by the faculty belonging to ‘Sales’ whose name is ‘Ratnam’

```
mysql> select faculty.facno,faculty.facname,department.deptno,department.deptname from faculty cross join department;
+-------+---------+--------+----------+
| facno | facname | deptno | deptname |
+-------+---------+--------+----------+
| 802   | Ratnam  | 11     | Sales    |
+-------+---------+--------+----------+
1 row in set (0.00 sec)
```

RESULT: The records from the tables are displayed using JOIN using EquiJoin, InnerJoin, OuterJoin.

Ex. No.: 10

Date:

Database with VIEW and INDEX

AIM:

To create view and index on the given relation.

DESCRIPTION:

VIEW:

MySQL has supported database views since version 5+. In MySQL, almost features of views conform to the SQL: 2003 standard. MySQL processes query against the views in two ways:

1. In a first way, MySQL creates a temporary table based on the view definition
2. Statement and executes the incoming query on this temporary table.
3. In a second way, MySQL combines the incoming query with the query defined the view into one query and executes the combined query.

SYNTAX- VIEW:

CREATE [ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}]
VIEW [**database_name**].[view_name]
AS [SELECT statement]

INDEX:

A database index, or just index, helps **speed up the retrieval of data from tables**. When you query data from a table, first MySQL **checks if the indexes exist**, then MySQL uses the indexes to select exact physical corresponding rows of the table instead of scanning the whole table..

SYNTAX- INDEX:

CREATE [UNIQUE|FULLTEXT|SPATIAL] **INDEX**index_name
USING [BTREE | HASH | RTREE]
ON table_name (column_name [(length)] [ASC | DESC],...)

Questions on View:

1. Create a view with name ‘v1’ using employees1 table which holds the value of employee_id and salary of employee.
2. Do the insert and delete records from v1 table.

```
mysql> select * from employees1;
+-------------+-------------+-----------+---------------+--------+
| employee_id | first_name  | last_name | device_serial | salary |
+-------------+-------------+-----------+---------------+--------+
|           1 | John        | Smith     | ABC123        |  60000 |
|           2 | Jane        | Doe       | DEF456        |  65000 |
|           3 | Bob         | Johnson   | GHI789        |  70000 |
|           4 | Sally       | Fields    | JKL012        |  75000 |
|           5 | Michael     | Smith     | MNO345        |  80000 |
|           6 | Emily       | Jones     | PQR678        |  85000 |
|           7 | David       | Williams  | STU901        |  90000 |
|           8 | Sarah       | Johnson   | VWX234        |  95000 |
|           9 | James       | Brown     | YZA567        | 100000 |
|          10 | Emma        | Miller    | BCD890        | 105000 |
|          11 | William     | Davis     | EFG123        | 110000 |
|          12 | Olivia      | Garcia    | HIJ456        | 115000 |
|          13 | Christopher | Rodriguez | KLM789        | 120000 |
|          14 | Isabella    | Wilson    | NOP012        | 125000 |
|          15 | Matthew     | Martinez  | QRS345        | 130000 |
|          16 | Sophia      | Anderson  | TUV678        | 135000 |
|          17 | Daniel      | Smith     | WXY901        | 140000 |
|          18 | Mia         | Thomas    | ZAB234        | 145000 |
|          19 | Joseph      | Hernandez | CDE567        | 150000 |
|          20 | Abigail     | Smith     | FGH890        | 155000 |
+-------------+-------------+-----------+---------------+--------+
20 rows in set (0.00 sec)
```

```
mysql> desc employees1;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| employee_id   | int         | YES  | MUL | NULL    |       |
| first_name    | varchar(50) | YES  |     | NULL    |       |
| last_name     | varchar(50) | YES  |     | NULL    |       |
| device_serial | varchar(15) | YES  |     | NULL    |       |
| salary        | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> create view v1 as select employee_id,salary from employees1;
Query OK, 0 rows affected (0.02 sec)

mysql> desc v1;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| employee_id | int  | YES  |     | NULL    |       |
| salary      | int  | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> insert into v1 values(101,100001);
Query OK, 1 row affected (0.01 sec)

mysql> select * from v1;
+-------------+--------+
| employee_id | salary |
+-------------+--------+
|           1 |  60000 |
|           2 |  65000 |
|           3 |  70000 |
|           4 |  75000 |
|           5 |  80000 |
|           6 |  85000 |
|           7 |  90000 |
|           8 |  95000 |
|           9 | 100000 |
|          10 | 105000 |
|          11 | 110000 |
|          12 | 115000 |
|          13 | 120000 |
|          14 | 125000 |
|          15 | 130000 |
|          16 | 135000 |
|          17 | 140000 |
|          18 | 145000 |
|          19 | 150000 |
|          20 | 155000 |
|         101 | 100001 |
+-------------+--------+
21 rows in set (0.00 sec)
```

```
mysql> select * from employees1;
+-------------+-------------+-----------+---------------+--------+
| employee_id | first_name  | last_name | device_serial | salary |
+-------------+-------------+-----------+---------------+--------+
|           1 | John        | Smith     | ABC123        |  60000 |
|           2 | Jane        | Doe       | DEF456        |  65000 |
|           3 | Bob         | Johnson   | GHI789        |  70000 |
|           4 | Sally       | Fields    | JKL012        |  75000 |
|           5 | Michael     | Smith     | MNO345        |  80000 |
|           6 | Emily       | Jones     | PQR678        |  85000 |
|           7 | David       | Williams  | STU901        |  90000 |
|           8 | Sarah       | Johnson   | VWX234        |  95000 |
|           9 | James       | Brown     | YZA567        | 100000 |
|          10 | Emma        | Miller    | BCD890        | 105000 |
|          11 | William     | Davis     | EFG123        | 110000 |
|          12 | Olivia      | Garcia    | HIJ456        | 115000 |
|          13 | Christopher | Rodriguez | KLM789        | 120000 |
|          14 | Isabella    | Wilson    | NOP012        | 125000 |
|          15 | Matthew     | Martinez  | QRS345        | 130000 |
|          16 | Sophia      | Anderson  | TUV678        | 135000 |
|          17 | Daniel      | Smith     | WXY901        | 140000 |
|          18 | Mia         | Thomas    | ZAB234        | 145000 |
|          19 | Joseph      | Hernandez | CDE567        | 150000 |
|          20 | Abigail     | Smith     | FGH890        | 155000 |
|         101 | NULL        | NULL      | NULL          | 100001 |
+-------------+-------------+-----------+---------------+--------+
21 rows in set (0.00 sec)
```

```
mysql> delete from v1 where employee_id=10;
Query OK, 1 row affected (0.01 sec)

mysql> select * from v1;
+-------------+--------+
| employee_id | salary |
+-------------+--------+
|           1 |  60000 |
|           2 |  65000 |
|           3 |  70000 |
|           4 |  75000 |
|           5 |  80000 |
|           6 |  85000 |
|           7 |  90000 |
|           8 |  95000 |
|           9 | 100000 |
|          11 | 110000 |
|          12 | 115000 |
|          13 | 120000 |
|          14 | 125000 |
|          15 | 130000 |
|          16 | 135000 |
|          17 | 140000 |
|          18 | 145000 |
|          19 | 150000 |
|          20 | 155000 |
|         101 | 100001 |
+-------------+--------+
20 rows in set (0.00 sec)
```

```
mysql> select * from employees1;
+-------------+-------------+-----------+---------------+--------+
| employee_id | first_name  | last_name | device_serial | salary |
+-------------+-------------+-----------+---------------+--------+
|           1 | John        | Smith     | ABC123        |  60000 |
|           2 | Jane        | Doe       | DEF456        |  65000 |
|           3 | Bob         | Johnson   | GHI789        |  70000 |
|           4 | Sally       | Fields    | JKL012        |  75000 |
|           5 | Michael     | Smith     | MNO345        |  80000 |
|           6 | Emily       | Jones     | PQR678        |  85000 |
|           7 | David       | Williams  | STU901        |  90000 |
|           8 | Sarah       | Johnson   | VWX234        |  95000 |
|           9 | James       | Brown     | YZA567        | 100000 |
|          11 | William     | Davis     | EFG123        | 110000 |
|          12 | Olivia      | Garcia    | HIJ456        | 115000 |
|          13 | Christopher | Rodriguez | KLM789        | 120000 |
|          14 | Isabella    | Wilson    | NOP012        | 125000 |
|          15 | Matthew     | Martinez  | QRS345        | 130000 |
|          16 | Sophia      | Anderson  | TUV678        | 135000 |
|          17 | Daniel      | Smith     | WXY901        | 140000 |
|          18 | Mia         | Thomas    | ZAB234        | 145000 |
|          19 | Joseph      | Hernandez | CDE567        | 150000 |
|          20 | Abigail     | Smith     | FGH890        | 155000 |
|         101 | NULL        | NULL      | NULL          | 100001 |
+-------------+-------------+-----------+---------------+--------+
20 rows in set (0.00 sec)
```

Questions on INDEX:

1. Create index1 for ‘salary’ attribute from employees1 relation and list the first name of the employees whose salary is above 145000 and explain the working principle of indexing and then drop the index1.
2. Create index1 for ‘employee_id’ attribute and display the first name of an employee whose employee id is 10 and explain the working principle of index1.

```
mysql> select * from employees1;
+-------------+-------------+-----------+---------------+--------+
| employee_id | first_name  | last_name | device_serial | salary |
+-------------+-------------+-----------+---------------+--------+
|           1 | John        | Smith     | ABC123        |  60000 |
|           2 | Jane        | Doe       | DEF456        |  65000 |
|           3 | Bob         | Johnson   | GHI789        |  70000 |
|           4 | Sally       | Fields    | JKL012        |  75000 |
|           5 | Michael     | Smith     | MNO345        |  80000 |
|           6 | Emily       | Jones     | PQR678        |  85000 |
|           7 | David       | Williams  | STU901        |  90000 |
|           8 | Sarah       | Johnson   | VWX234        |  95000 |
|           9 | James       | Brown     | YZA567        | 100000 |
|          10 | Emma        | Miller    | BCD890        | 105000 |
|          11 | William     | Davis     | EFG123        | 110000 |
|          12 | Olivia      | Garcia    | HIJ456        | 115000 |
|          13 | Christopher | Rodriguez | KLM789        | 120000 |
|          14 | Isabella    | Wilson    | NOP012        | 125000 |
|          15 | Matthew     | Martinez  | QRS345        | 130000 |
|          16 | Sophia      | Anderson  | TUV678        | 135000 |
|          17 | Daniel      | Smith     | WXY901        | 140000 |
|          18 | Mia         | Thomas    | ZAB234        | 145000 |
|          19 | Joseph      | Hernandez | CDE567        | 150000 |
|          20 | Abigail     | Smith     | FGH890        | 155000 |
+-------------+-------------+-----------+---------------+--------+
20 rows in set (0.00 sec)
```

```
mysql> select first_name from employees1 where salary>145000;
+------------+
| first_name |
+------------+
| Joseph     |
| Abigail    |
+------------+
2 rows in set (0.00 sec)

mysql> explain select first_name from employees1 where salary>145000;
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | employees1 | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   20 |    33.33 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> create index index1 on employees1(salary);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select first_name from employees1 where salary>145000;
+----+-------------+-----------+------------+-------+---------------+--------+---------+------+------+----------+-----------------------+
| id | select_type | table     | partitions | type  | possible_keys | key    | key_len | ref  | rows | filtered | Extra                 |
+----+-------------+-----------+------------+-------+---------------+--------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | employees1 | NULL       | range | index1        | index1 | 5       | NULL |    2 |   100.00 | Using index condition |
+----+-------------+-----------+------------+-------+---------------+--------+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> drop index index1 on employees1;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> select first_name from employees1 where employee_id=10;
+------------+
| first_name |
+------------+
| Emma       |
+------------+
1 row in set (0.00 sec)

mysql> explain select first_name from employees1 where employee_id=10;
+----+-------------+------------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table      | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+------------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | employees1 | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   20 |    10.00 | Using where |
+----+-------------+------------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> create index index1 on employees1(employee_id);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select first_name from employees1 where employee_id=10;
+----+-------------+------------+------------+------+---------------+--------+---------+-------+------+----------+-------+
| id | select_type | table      | partitions | type | possible_keys | key    | key_len | ref   | rows | filtered | Extra |
+----+-------------+------------+------------+------+---------------+--------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | employees1 | NULL       | ref  | index1        | index1 | 5       | const |    1 |   100.00 | NULL  |
+----+-------------+------------+------------+------+---------------+--------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> show index from employees1;
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table     | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| employees1 |          1 | index1   |            1 | employee_id | A         |          20 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | NULL       |
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.00 sec)
```

RESULT:

The records from the tables are displayed using view and index on the given relation.

Ex. No.: 11

Date:

DATABASE WITH AUTO_INCREMENT SEQUENCES

AIM:

To create Auto Increment sequence on the given relation.

DESCRIPTION:

SEQUENCE:

In MySQL, a sequence is a **list of integers generated in the ascending order i.e., 1,2,3…** Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee numbers in HR, equipment numbers in services management system, etc.

To create a sequence in MySQL automatically, you set the AUTO_INCREMENT attribute to a column, which typically is a primary key column.

SYNTAX:

CREATE TABLEtable_name(

col_name1**AUTO_INCREMENT PRIMARYKEY,** col_name2, col_name3, ….);

Questions:

1. Populate register number using auto increment in DBMS_Stud table.
2. Manually populate register number
3. Drop the auto increment.

```
mysql> CREATE TABLE DBMS_Stud(
    -> Reg_No INT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> PRIMARY KEY (Reg_No),
    -> Name VARCHAR(30) NOT NULL,
    -> Department VARCHAR(30) NOT NULL,
    -> Mark INT(30) NOT NULL
    -> );
Query OK, 0 rows affected, 1 warning (0.04 sec)

mysql> desc DBMS_Stud;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| Reg_No     | int unsigned | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(30)  | NO   |     | NULL    |                |
| Department | varchar(30)  | NO   |     | NULL    |                |
| Mark       | int          | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

```
mysql> INSERT INTO DBMS_Stud(Name, Department, Mark) VALUES
    ->     ('Raj','CSE', 89),
    ->     ('Rajesh','CSE', 88),
    ->     ('Ramesh','ECE', 90),
    ->     ('Rajan','ECE', 85);
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+--------+------------+------+
| Reg_No | Name   | Department | Mark |
+--------+--------+------------+------+
|      1 | Raj    | CSE        |   89 |
|      2 | Rajesh | CSE        |   88 |
|      3 | Ramesh | ECE        |   90 |
|      4 | Rajan  | ECE        |   85 |
+--------+--------+------------+------+
4 rows in set (0.00 sec)
```

```
mysql> INSERT INTO DBMS_Stud(Reg_No,Name, Department, Mark) VALUES
    ->     (10,'Aarthi','CSE', 89),
    ->     (12,'Anu','CSE', 88),
    ->     (11,'Anbu','ECE', 90);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+--------+------------+------+
| Reg_No | Name   | Department | Mark |
+--------+--------+------------+------+
|      1 | Raj    | CSE        |   89 |
|      2 | Rajesh | CSE        |   88 |
|      3 | Ramesh | ECE        |   90 |
|      4 | Rajan  | ECE        |   85 |
|     10 | Aarthi | CSE        |   89 |
|     11 | Anbu   | ECE        |   90 |
|     12 | Anu    | CSE        |   88 |
+--------+--------+------------+------+
7 rows in set (0.00 sec)
```

```
mysql> INSERT INTO DBMS_Stud(Name, Department, Mark) VALUES
    ->     ('abc','CSE', 89),
    ->     ('xyz','CSE', 88);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+--------+------------+------+
| Reg_No | Name   | Department | Mark |
+--------+--------+------------+------+
|      1 | Raj    | CSE        |   89 |
|      2 | Rajesh | CSE        |   88 |
|      3 | Ramesh | ECE        |   90 |
|      4 | Rajan  | ECE        |   85 |
|     10 | Aarthi | CSE        |   89 |
|     11 | Anbu   | ECE        |   90 |
|     12 | Anu    | CSE        |   88 |
|     13 | abc    | CSE        |   89 |
|     14 | xyz    | CSE        |   88 |
+--------+--------+------------+------+
9 rows in set (0.00 sec)
```

```
mysql> INSERT INTO DBMS_Stud(Name, Department, Mark) VALUES
    ->     ('AAAAA','CSE', 89),
    ->     ('BBBBB','ECE', 88),
    ->     ('CCCCC','ECE', 88);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+--------+------------+------+
| Reg_No | Name   | Department | Mark |
+--------+--------+------------+------+
|      1 | Raj    | CSE        |   89 |
|      2 | Rajesh | CSE        |   88 |
|      3 | Ramesh | ECE        |   90 |
|      4 | Rajan  | ECE        |   85 |
|     10 | Aarthi | CSE        |   89 |
|     11 | Anbu   | ECE        |   90 |
|     12 | Anu    | CSE        |   88 |
|     13 | abc    | CSE        |   89 |
|     14 | xyz    | CSE        |   88 |
|     15 | AAAAA  | CSE        |   89 |
|     16 | BBBBB  | ECE        |   88 |
|     17 | CCCCC  | ECE        |   88 |
+--------+--------+------------+------+
12 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE DBMS_Stud AUTO_INCREMENT =50;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> INSERT INTO DBMS_Stud(Name, Department, Mark) VALUES
    ->     ('DD','CSE', 89),
    ->     ('EE','ECE', 88),
    ->     ('FF','ECE', 88);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+--------+------------+------+
| Reg_No | Name   | Department | Mark |
+--------+--------+------------+------+
|      1 | Raj    | CSE        |   89 |
|      2 | Rajesh | CSE        |   88 |
|      3 | Ramesh | ECE        |   90 |
|      4 | Rajan  | ECE        |   85 |
|     10 | Aarthi | CSE        |   89 |
|     11 | Anbu   | ECE        |   90 |
|     12 | Anu    | CSE        |   88 |
|     13 | abc    | CSE        |   89 |
|     14 | xyz    | CSE        |   88 |
|     15 | AAAAA  | CSE        |   89 |
|     16 | BBBBB  | ECE        |   88 |
|     17 | CCCCC  | ECE        |   88 |
|     50 | DD     | CSE        |   89 |
|     51 | EE     | ECE        |   88 |
|     52 | FF     | ECE        |   88 |
+--------+--------+------------+------+
15 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE DBMS_Stud DROP Reg_No;
Query OK, 15 rows affected (0.07 sec)
Records: 15  Duplicates: 0  Warnings: 0

mysql> INSERT INTO DBMS_Stud(Name, Department, Mark) VALUES
    ->     ('gggg','CSE', 89),
    ->     ('hhhh','ECE', 88);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from DBMS_Stud;
+--------+------------+------+
| Name   | Department | Mark |
+--------+------------+------+
| Raj    | CSE        |   89 |
| Rajesh | CSE        |   88 |
| Ramesh | ECE        |   90 |
| Rajan  | ECE        |   85 |
| Aarthi | CSE        |   89 |
| Anbu   | ECE        |   90 |
| Anu    | CSE        |   88 |
| abc    | CSE        |   89 |
| xyz    | CSE        |   88 |
| AAAAA  | CSE        |   89 |
| BBBBB  | ECE        |   88 |
| CCCCC  | ECE        |   88 |
| DD     | CSE        |   89 |
| EE     | ECE        |   88 |
| FF     | ECE        |   88 |
| gggg   | CSE        |   89 |
| hhhh   | ECE        |   88 |
+--------+------------+------+
17 rows in set (0.00 sec)
```

```
mysql> INSERT INTO DBMS_Stud(Reg_No, Name, Department, Mark) VALUES
    ->     (111,'yy','CSE', 89),
    ->     (222,'zz','ECE', 88);
ERROR 1054 (42S22): Unknown column 'Reg_No' in 'field list'
mysql> desc DBMS_Stud;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Name       | varchar(30) | NO   |     | NULL    |       |
| Department | varchar(30) | NO   |     | NULL    |       |
| Mark       | int         | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

RESULT:

The records from the tables are displayed using auto_incrementsequence on the given relation.

Ex:No: 12

Date:

Simple Programming using REPEAT, WHILE

Aim:

To learn how to use various MySQL loop statements including while, repeat to run a block of code repeatedly based on a condition.

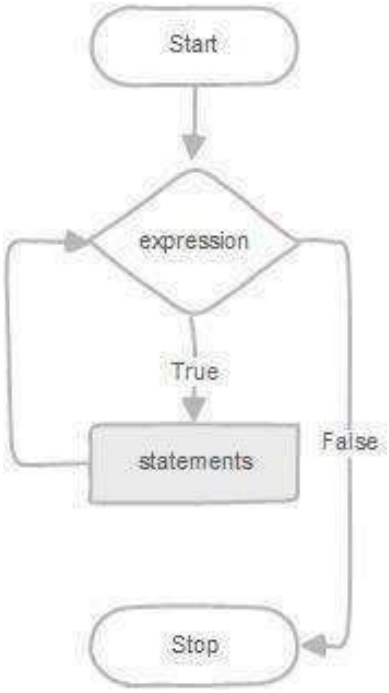
WHILE loop Syntax:

WHILE expression
DOstatements
END WHILE;

Procedure:

1. The WHILE loop checks the expression at the beginning of each iteration.
2. If the expression valuates to TRUE, MySQL will execute statements between WHILE and END WHILE until the expression evaluates to FALSE.
3. The WHILE loop is called pretest loop because it checks the expression before the statements execute.

The following flowchart illustrates the WHILE loop statement:



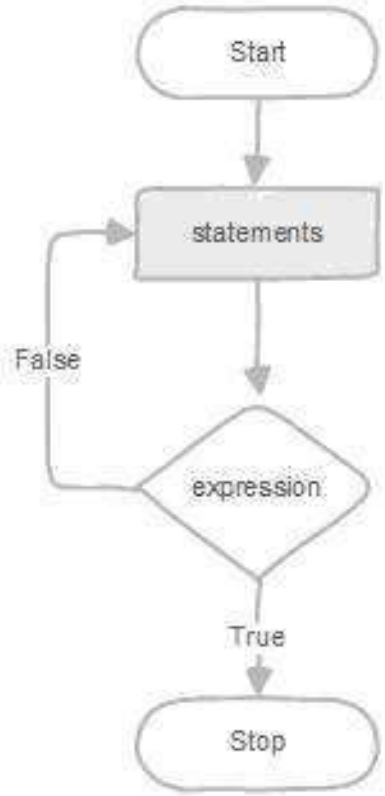
REPEAT loop Syntax:

REPEATstatements
UNTIL expression
END REPEAT;

Procedure:

1. First, MySQL executes the statements, and then it evaluates the expression.
2. Ifthe expression evaluates to FALSE, MySQL executes the statements repeatedly untilthe expression evaluates to TRUE.
3. Because the REPEAT loop statement checks the expression after the execution of statements, the REPEAT loop statement is also known as the post-test loop.

The following flowchart illustrates the REPEAT loop statement:



Program-1:

Write a function that uses WHILE statement to build a string repeatedly until the value of the variable becomes x greater than 5. Then, we display the final string using a SELECT statement.

Program-2:

Write a function that uses REPEAT statement which would repeat the loop until income is greater than or equal to 4000, at which point the REPEAT loop would be terminated.

Program-1:

```
mysql> CREATE PROCEDURE test_mysql_while_loop()
    ->  BEGIN
    ->  DECLARE x  INT;
    ->  DECLARE str  VARCHAR(255);
    ->
    ->  SET x = 1;
    ->  SET str =  '';
    ->
    ->  WHILE x  <= 5 DO
    ->  SET  str = CONCAT(str,x,',');
    ->  SET  x = x + 1;
    ->  END WHILE;
    ->
    ->  SELECT str;
    ->  END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> CALL test_mysql_while_loop() //
+------------+
| str        |
+------------+
| 1,2,3,4,5, |
+------------+
1 row in set (0.00 sec)

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

Program-2:

```
mysql> CREATE PROCEDURE dorepeat(p1 INT) BEGIN SET @x=0; REPEAT SET @x=@x+1; UNT
IL @x>p1 END REPEAT; END//
Query OK, 0 rows affected (0.01 sec)

mysql> CALL dorepeat(4001) //
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT @x;
    -> //
+------+
| @x   |
+------+
| 4002 |
+------+
1 row in set (0.00 sec)
```

RESULT: Thus the Simple programming exercise using (REPEAT, WHILE)executed successfully.

Ex:No: 13

Date:

Simple programming usingCASE and LOOP

Aim:

To learn how to use various MySQL loop statements including case and loop to run a block of code repeatedly based on a condition.

Procedure:

In MySQL, the CASE statement has the functionality of an IF-THEN-ELSE statement and has 2 syntaxes that we will explore.

CASE Syntax

```
CASE case_value
WHEN when_value THEN statement_list
[WHEN when_value THEN statement_list] ...
[ELSE statement_list]
END CASE
```

Procedure:

LOOP Syntax

```
[begin_label:] LOOP
statement_list
END LOOP [end_label]
```

LOOP implements a simple loop construct, enabling repeated execution of the statement list, which consists of one or more statements, each terminated by a semicolon (;) statement delimiter. The statements within the loop are repeated until the loop is terminated. Usually, this is accomplished with a LEAVE statement. Within a stored function, RETURN can also be used, which exits the function entirely.

Program 1:

Write a function that uses CASE statement where if monthly_value is equal to or less than 4000, then income_level will be set to 'Low Income'. If monthly_value is equal to or less than 5000, then income_level will be set to 'Avg Income'. Otherwise, income_level will be set to 'High Income'.

Program 2:

Write a function that will use ITERATE statement which would cause the loop to repeat while income is less than 4000. Once income is greater than or equal to 4000, would terminate the LOOP.

Program -1

```
mysql> CREATE FUNCTION IncomeLevel ( monthly_value INT )
    -> RETURNS varchar(20)
    ->
    -> BEGIN
    ->
    ->    DECLARE income_level varchar(20);
    ->
    ->    CASE monthly_value
    ->       WHEN 4000 THEN
    ->         SET income_level = 'Low Income';
    ->
    ->       WHEN 5000 THEN
    ->         SET income_level = 'Avg Income';
    ->
    ->       ELSE
    ->         SET income_level = 'High Income';
    ->    END CASE;
    ->
    ->    RETURN income_level;
    ->
    -> END; //
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT INCOMELEVEL(5300); //
+-------------------+
| INCOMELEVEL(5300) |
+-------------------+
| High Income       |
+-------------------+
1 row in set (0.00 sec)
```

Program -2

```
mysql> CREATE FUNCTION CALCINCOME2 ( starting_value INT )
    -> RETURNS INT
    ->
    -> BEGIN
    ->
    ->    DECLARE income INT;
    ->
    ->    SET income = 0;
    ->
    ->    label1: LOOP
    ->      SET income = income + starting_value;
    ->      IF income < 4000 THEN
    ->        ITERATE label1;
    ->      END IF;
    ->      LEAVE label1;
    ->    END LOOP label1;
    ->
    ->    RETURN income;
    ->
    -> END; //
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT CALCINCOME2(2100);
    -> //
+-------------------+
| CALCINCOME2(2100) |
+-------------------+
|              4200 |
+-------------------+
1 row in set (0.00 sec)

mysql>
```

RESULT: Thus the Simple programming exercise using CASE and LOOP executed successfully.

Ex:No: 14

Date:

TCL COMMANDS – COMMIT, SAVEPOINT, ROLLBACK

Aim:

To learn how to use various TCL commands Commit, Savepoint and Rollback SQL commands

Procedure and Syntax:

Transaction Control Language (TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements. It also allows statements to be grouped together into logical transactions.

COMMIT:

- COMMIT command is used to permanently save any transaction into the database.
- When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.
- To avoid that, we use the COMMIT command to mark the changes as permanent

Syntax:

COMMIT;

ROLLBACK:

- This command restores the database to last committed state.
- It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.
- If we have used the UPDATE command to make some changes into the database, and realize that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command.

Syntax:

ROLLBACK;

ROLLBACK TO savepoint_name;

SAVEPOINT:

- SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Syntax:

SAVEPOINT savepoint_name;

Problem 1:

Create a following table Class and insert values into it in the following order and create savepoints in between them. Try to rollback the save point and check your output by giving select commands.

Let us use some SQL queries on the above table and see the results.

UPDATE class SET name ='bravo' WHERE id ='5';

SAVEPOINT A;

INSERT INTO class VALUES('uppal', 6);

SAVEPOINT B;

INSERTINTO class VALUES(‘balu’, 7);

SAVEPOINT C;

Now let's use the ROLLBACK command to roll back the state of data to the savepoint.

```
mysql> create table clss(name varchar(10),id int(5));
Query OK, 0 rows affected (0.19 sec)

mysql> insert into clss values("dj",5);
Query OK, 1 row affected (0.00 sec)

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

mysql> update class set name="bravo" where id="5";
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

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

mysql> insert into clss values("uppal",6);
Query OK, 1 row affected (0.00 sec)

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

mysql> insert into clss values("balu",7);
Query OK, 1 row affected (0.00 sec)

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

mysql> select * from clss;
+-------+------+
| name  | id   |
+-------+------+
| dj    |    5 |
| uppal |    6 |
| balu  |    7 |
+-------+------+
3 rows in set (0.00 sec)

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

mysql> select * from clss;
+-------+------+
| name  | id   |
+-------+------+
| dj    |    5 |
| uppal |    6 |
+-------+------+
2 rows in set (0.00 sec)

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

mysql> select * from clss;
+------+------+
| name | id   |
+------+------+
| dj   |    5 |
+------+------+
1 row in set (0.00 sec)
```

Result:

The commands SQL COMMIT, ROLLBACK and SAVEPOINT are implemented and result is verified.

Ex:No: 15

Date:

DCL COMMANADS– GRANT, REVOKE

Aim:

To learn how to use various DCL commands such as GRANT and REVOKE.

Procedure and Syntax:

Data Control Language (DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

System: This includes permissions for creating session, table, etc and all types ofother system privileges.

Object: This includes permissions for any command or query to perform any operation on the database tables.

In DCL we have two commands,

GRANT: Used to provide any user access privileges or other privileges forthe database.

REVOKE: Used to take back permissions from any user.

- Allow a User to create session
- When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/privileges are granted to the user.
- Following command can be used to grant the session creating privileges.

GRANT CREATE SESSION TO username;

Allow a User to create table

To allow a user to create tables in the database, we can use the below command,

GRANT CREATE TABLE TO username;

Provide user with space on tablespace to store table

Allowing a user to create table is not enough to start storing data in that table. We also must provide the user with privileges to use the available tablespace for their table and data.

ALTER USER username QUOTA UNLIMITED ON SYSTEM;

The above command will alter the user details and will provide it access to unlimited tablespace on system.

NOTE: Generally unlimited quota is provided to Admin users.

Grant all privilege to a User

Sysdba is a set of privileges which has all the permissions in it. So if we want toprovide all the privileges to any user, we can simply grant them the sysdba permission.

GRANT sysdba TO username

Grant permission to create any table

Sometimes user is restricted from creating come tables with names which are reserved for system tables. But we can grant privileges to a user to create any table using the below command,

GRANT CREATE ANY TABLE TO username

Grant permission to drop any table

As the title suggests, if you want to allow user to drop any table from the database, then grant this privilege to the user,

GRANT DROP ANY TABLE TO username

To take back Permissions

And, if you want to take back the privileges from any user, use the REVOKE command.

REVOKE CREATE TABLE FROM username

RESULT: Thus the DCL commands GRANT and REVOKE SQL executed successfully.

Ex:No: 16

Date:

HIGH LEVEL PROGRAMMING EXTENSIONS - PROCEDURES

Aim:

To implement procedures using program in MySQL.

PROCEDURES:

A procedure is a subprogram that performs a specific action.

Creating a procedure

We use the CREATE PROCEDURE statement to create a new stored procedure. We specify the name of stored procedure after the CREATE PROCEDURE statement. The DELIMITER command is used to change thestandard delimiter of MySQLcommands (i.e. ;). As the statements within the routines (functions, stored procedures or triggers) end with a semi-colon (;), to treat them as a compound statement we use DELIMITER.

Calling stored procedures(Executing a procedure)

In order to call a stored procedure, you use the following SQL command:

CALLstored_procedure_name();

Problem 1:

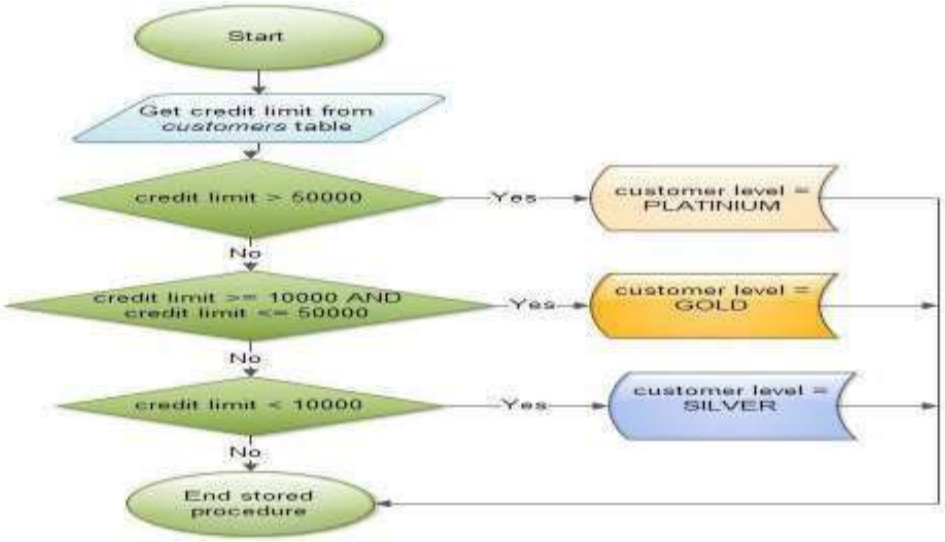
Create a simple procedure to get all the records from the table ‘student_info’.

Problem 2:

Create a stored procedure GetCustomerLevel() that accepts two parameters customer number and customer level.

- First, it gets the credit limit from the customers table.
- Then, based on the credit limit, it determines the customer level: PLATINUM, GOLD, and SILVER.
- The parameter p_customerlevel stores the level of the customer and is used by the calling program.

The following flowchart demonstrates the logic of determining customer level.



Program-1:

```
mysql> CREATE PROCEDURE student_info()
    -> select * from student_info;
Query OK, 0 rows affected (0.00 sec)

mysql> call student_info();
+-------+-------+-----------+---------+
| stuid | name  | area      | subject |
+-------+-------+-----------+---------+
|   201 | raj   | chennai   | dbms    |
|   202 | rahul | hyderabad | ooad    |
|   203 | rahim | mumbai    | java    |
|   204 | vikas | kochi     | python  |
+-------+-------+-----------+---------+
4 rows in set (0.00 sec)

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

Program-2:

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE GetCustomerLevel(
    ->     in  p_customerNumber int(11),
    ->     out p_customerLevel  varchar(10))
    -> BEGIN
    ->     DECLARE creditlim double;
    ->
    ->     SELECT creditlimit INTO creditlim
    ->     FROM customers
    ->     WHERE customerNumber = p_customerNumber;
    ->
    ->     IF creditlim > 50000 THEN
    ->  SET p_customerLevel = 'PLATINUM';
    ->     ELSEIF (creditlim <= 50000 AND creditlim >= 10000) THEN
    ->         SET p_customerLevel = 'GOLD';
    ->     ELSEIF creditlim < 10000 THEN
    ->         SET p_customerLevel = 'SILVER';
    ->     END IF;
    ->
    -> END$$
Query OK, 0 rows affected (0.00 sec)
```

RESULT: Thus the high level procedures programsare executed successfully.

Ex:No: 17

Date:

HIGH LEVEL PROGRAMMING EXTENSIONS-FUNCTIONS

Aim:

To implement Functions using program in MySQL.

Functions:

A function is a subprogram that computes a value.

Creating a function

The CREATE FUNCTION statement is also used in MySQL to support UDFs (user-defined functions). A UDF can be regarded as an external stored function.

MySQL stored function syntax

CREATE FUNCTION function_name(param1,param2,…)

RETURNS datatype

[NOT] DETERMINISTIC

…..

statements

….

Problem 1:

Create a function that returns the level of a customer based on credit limit.

(Use the IF statement to determine the credit limit).

If credit limit > 50000 then customer_level = PLATINUM

If credit limit >= 10000 AND credit limit <= 50000 then customer_level = GOLD

If credit limit credit limit < 10000 then customer_level = SILVER

<u>RECURSION in Mysql Procedures</u>

Mysql version should be >= 5.

Have to set system parameters. This means putting the recursion count limit.

SET @@GLOBAL.max_sp_recursion_depth = 255;

SET @@session.max_sp_recursion_depth = 255;

Problem 2

Write a recursive MySQL procedure compute the factorial of a number .

Program- 1:

```
mysql> DELIMITER //
mysql> CREATE FUNCTION CustomerLevel(p_CREDITLIMIT INT) RETURNS VARCHAR(10)
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE lvl VARCHAR(10);
    -> IF p_CREDITLIMIT > 50000 THEN
    -> SET lvl = 'PLATINUM';
    -> ELSEIF (p_CREDITLIMIT <= 50000 AND p_CREDITLIMIT >= 10000) THEN
    -> SET lvl = 'GOLD';
    -> ELSEIF p_CREDITLIMIT < 10000 THEN
    -> SET lvl = 'SILVER';
    -> END IF;
    -> RETURN (lvl);
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT CNAME,CustomerLevel(CREDITLIMIT) FROM CUSTOMER ORDER BY CNAME//
+----------+----------------------------+
| CNAME    | CustomerLevel(CREDITLIMIT) |
+----------+----------------------------+
| DINESH   | GOLD                       |
| NAGENDRA | PLATINUM                   |
| RAJA     | GOLD                       |
| RAMU     | SILVER                     |
+----------+----------------------------+
4 rows in set (0.00 sec)
```

Program- 2:

```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE factorial(IN n INT,OUT fact INT)
   -> BEGIN
   -> IF n=1 THEN
   ->     SET fact:=1;
   -> ELSE
   ->    CALL factorial(n-1,fact);
   ->    SET fact:=n*fact;
   ->   END IF;
   -> END
   -> $$
uery OK, 0 rows affected (0.01 sec)

ysql> CALL find_fact(5);
   -> $$
-------+
 @fact |
-------+
   120 |
-------+
 row in set (0.00 sec)

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

RESULT: Thus the Functions using program in MySQL executed successfully .

Ex.No: 18

Date:

HIGH LEVEL LANGUAGE EXTENSION WITH CURSORS

Aim:

To implement Cursors using program in MySQL.

Description:

Cursor is a **Temporary Memory** or **Temporary Work Station**. It is allocated by Database Server at the Time of Performing DML(Data Manipulation Language)operations on Table by User. Cursors are used to store Database Tables.

1. **Implicit Cursors:**

   Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

2. **Explicit Cursors:**

   Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for **Fetching data from Table in Row-By-Row Manner**.

How to create Explicit Cursor:

1. **Declare Cursor Object.**

   Syntax : DECLARE cursor_name CURSOR FOR SELECT * FROM table_name

2. **Open Cursor Connection.**

   Syntax : OPEN cursor_connection

3. **Fetch Data from cursor.**

   There are total 6 methods to access data from cursor. They are as follows :

   FIRST is used to fetch only the first row from cursor table.

   LAST is used to fetch only last row from cursor table.

   NEXT is used to fetch data in forward direction from cursor table.

   PRIOR is used to fetch data in backward direction from cursor table.

   ABSOLUTE n is used to fetch the exact n^{th} row from cursor table.

   RELATIVE n is used to fetch the data in incremental way as well as decremental way.

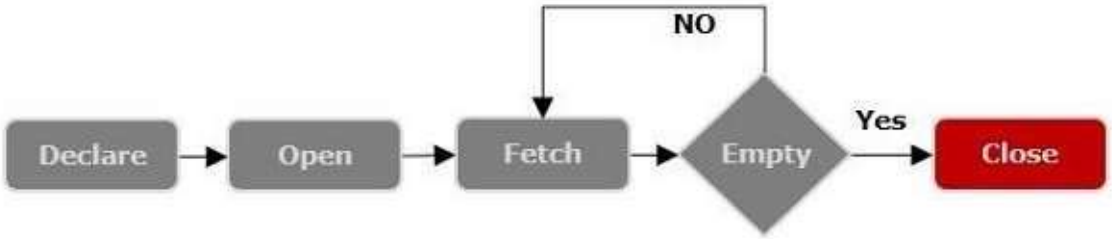
Syntax : FETCH NEXT/FIRST/LAST/PRIOR/ABSOLUTE n/RELATIVE n FROM cursor_name

4. **Close cursor connection.**

   Syntax : CLOSE cursor_name

5. **Deallocate cursor memory.**

   Syntax : DEALLOCATE cursor_name



Problem- 1

Write a Cursor program using MySQL to retrieve the email-ids(build an email list) of employees from employees table.

SOLUTION :

```
create table employees(id integer, Name varchar(100), email varchar(100));
insert into employees(id, Name, email) values(1, "Harry Potter", "pharry@warnerbros.com");
insert into employees(id, Name, email) values(2, "Clark Kent","kclark@dccomics.com");
insert into employees(id, Name, email) values(3, "Tony Stark", "stony@marvel.com");

DELIMITER $$
CREATE PROCEDURE build_email_list (INOUT email_listvarchar(4000))

BEGIN

DECLARE v_finished INTEGER DEFAULT 0;
DECLARE v_emailvarchar(100) DEFAULT "";

-- declare cursor for employee email

DECLAREemail_cursor CURSOR FOR
```

```
SELECT email FROM employees;

-- declare NOT FOUND handler

DECLARE CONTINUE HANDLER FOR
NOT FOUND SET v_finished = 1;

OPEN email_cursor;
get_email: LOOP

FETCH email_cursor INTO v_email;
	IF v_finished = 1 THEN
		LEAVE get_email;

	END IF;
-- build email list
	SET email_list = CONCAT(v_email,";",email_list);

	END LOOP get_email;
CLOSE email_cursor;

END$$
DELIMITER ;

-- Calling the procedure and getting the email list

SET @email_list = "";
CALL build_email_list(@email_list);

SELECT @email_list;
```

Program :-1

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE build_email_list (INOUT email_list varchar(4000))
    -> BEGIN
    -> DECLARE v_finished INTEGER DEFAULT 0;
    ->     DECLARE v_email varchar(100) DEFAULT "";
    -> DECLARE email_cursor CURSOR FOR
    -> SELECT email FROM employees;
    -> DECLARE CONTINUE HANDLER FOR
    ->     NOT FOUND SET v_finished = 1;
    -> OPEN email_cursor;
    -> get_email:LOOP
    -> FETCH email_cursor INTO v_email;
    -> IF v_finished = 1 THEN
    -> LEAVE get_email;
    -> END IF;
    -> SET email_list = CONCAT(v_email,";",email_list);
    -> END LOOP get_email;
    -> CLOSE email_cursor;
    -> END $$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> SET @email_list = "";
Query OK, 0 rows affected (0.00 sec)

mysql> CALL build_email_list(@email_list);
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select @email_list;
+-----------------------------------------------------------+
| @email_list                                               |
+-----------------------------------------------------------+
| stony@marvel.com;kclark@decomics.com;pharry@warnerbros.com; |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

RESULT:

Thus the Cursor program using MySQL is executed successfully.

Ex:No:19

Date:

TRIGGER

Aim:

To implement trigger in MySQL.

A trigger or database trigger is a stored program **executed automatically** to respond to a specific event e.g., insert, update or delete occurred in a table.

Syntax-Create trigger

```
CREATE TRIGGER trigger_nametrigger_timetrigger_event
ON table_name
FOR EACH ROW
BEGIN
...
END;
```

Program 1 :

Create a trigger in MySQL to log the changes of the STUDENT table with fields ID, Name and Email. Also create a new table named EMPLOYEES_AUDIT to keep the changes of the employee table. Create a **BEFORE UPDATE trigger** that is invoked before a change is made to the employees table.

Program:-

```
mysql> DELIMITER //
mysql> CREATE TRIGGER before_student_update
    -> BEFORE UPDATE ON student
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO student_audit
    -> SET action ='update',
    -> student_id =OLD.id,
    -> lastname=OLD.Name,
    -> changedat = NOW();
    -> END //
Query OK, 0 rows affected (0.06 sec)

mysql> DELIMITER;
mysql> update student set name ='tony stark_c'Where id=3;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * fron student_audit;
+----+------------+------------+---------------------+--------+
| id | student_id | lastname   | changedat           | action |
+----+------------+------------+---------------------+--------+
|  1 |          3 | tony stark | 2019-08-12 13:07:44 | update |
+----+------------+------------+---------------------+--------+
1 row in set (0.00 sec)
```

RESULT: Thus the trigger is executed successfully.

Exp. No.:20

Date:

MYSQL STRING FUNCTIONS- REPLACE, REPEAT,REVERSE, RIGHT, LEFT, RPAD,LPAD

AIM:

To implement Replace, Repeat, Reverse, Right, Left,Rpad and LpadString Functionsusing MySQL.

REPLACE()

MySQL REPLACE() replaces all the occurrences of a substring within a string.

Syntax : REPLACE(str,from_str,to_str)

```
mysql> SELECT Replace("Database Management Systems","Management","");
+-------------------------------------------------------+
| Replace("Database Management Systems","Management","") |
+-------------------------------------------------------+
| Database  Systems                                     |
+-------------------------------------------------------+
1 row in set (0.01 sec)
```

REPEAT()

MySQL REPEAT() repeats a string for a specified number of times.

The function returns NULL either any either of the arguments are NULL.

Syntax : REPEAT(str,count)

```
mysql> SELECT REPEAT('DBMS',10);
+-----------------------------------------+
| REPEAT('DBMS',10)                       |
+-----------------------------------------+
| DBMSDBMSDBMSDBMSDBMSDBMSDBMSDBMSDBMSDBMS |
+-----------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT REPEAT(' DBMS ',10);
+-------------------------------------------------------------+
| REPEAT(' DBMS ',10)                                         |
+-------------------------------------------------------------+
|  DBMS  DBMS  DBMS  DBMS  DBMS  DBMS  DBMS  DBMS  DBMS  DBMS  |
+-------------------------------------------------------------+
1 row in set (0.00 sec)
```

REVERSE()

Returns a given string with the order of the characters reversed.

Syntax : REVERSE(str)

```
mysql> SELECT Reverse("Database");
+---------------------+
| Reverse("Database") |
+---------------------+
| esabataD            |
+---------------------+
1 row in set (0.00 sec)
```

RIGHT()

MySQL RIGHT() extracts a specified number of characters from the right side of a given string.

Syntax : RIGHT(str,len)

```
mysql> SELECT Right("Database",4);
+---------------------+
| Right("Database",4) |
+---------------------+
| base                |
+---------------------+
1 row in set (0.00 sec)
```

LEFT()

MySQL LEFT() returns a specified number of characters from the left of a given string. Both the number and the string are supplied in the arguments as str and len of the function.

Syntax : LEFT(str,len)

```
mysql> SELECT LEFT('Database', 5);
+---------------------+
| LEFT('Database', 5) |
+---------------------+
| Datab               |
+---------------------+
1 row in set (0.00 sec)
```

RPAD()

MySQL RPAD() function pads strings from right. The actual string which is to be padded as str, length of the string returned after padding as len and string which is used for padding as padstr is used as a parameters within the argument.

Syntax : RPAD(str,len,padstr)

```
mysql> SELECT Rpad("Database",14,'#');
+-------------------------+
| Rpad("Database",14,'#') |
+-------------------------+
| Database######          |
+-------------------------+
1 row in set (0.00 sec)
```

LPAD()

MySQL LPAD() left pads a string with another string. The actual string, a number indicating the length of the padding in characters (optional) and the string to be used for left padding - all are passed as arguments.

Syntax : LPAD(str,len,padstr)

```
mysql> SELECT LPAD('Database',20,'$');
+-------------------------+
| LPAD('Database',20,'$') |
+-------------------------+
| $$$$$$$$$$$$Database    |
+-------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT LPAD('Database',4,'$');
+------------------------+
| LPAD('Database',4,'$') |
+------------------------+
| Data                   |
+------------------------+
1 row in set (0.00 sec)
```

ASCII()

This function returns the numeric value of the leftmost character of the string str. Returns 0 if str is the empty string. Returns NULL if str is NULL

Syntax : ASCII(str)

```
mysql> SELECT ASCII(12);
+-----------+
| ASCII(12) |
+-----------+
|        49 |
+-----------+
1 row in set (0.00 sec)

mysql> SELECT ASCII('12');
+-------------+
| ASCII('12') |
+-------------+
|          49 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT ASCII('a');
+------------+
| ASCII('a') |
+------------+
|         97 |
+------------+
1 row in set (0.00 sec)
```

BIN()

Returns a string representation of the binary value of N, where N is a longlong (BIGINT) number. Returns NULL if N is NULL.

Syntax : BIN(N)

```
mysql> SELECT BIN(02);
+---------+
| BIN(02) |
+---------+
| 10      |
+---------+
1 row in set (0.00 sec)

mysql> SELECT BIN(11);
+---------+
| BIN(11) |
+---------+
| 1011    |
+---------+
1 row in set (0.00 sec)
```

OCT()

Returns a string representation of the octal value of N, where N is a longlong (BIGINT) number. Returns NULL if N is NULL.

Syntax : OCT(N)

```
mysql> SELECT OCT(8);
+--------+
| OCT(8) |
+--------+
| 10     |
+--------+
1 row in set (0.00 sec)
```

RESULT: Thus the String Functionssuch as Replace, Repeat, Reverse, Right, Left,Rpad and Lpadare executed successfully.

Exp. No.: 21

Date:

MYSQL STRING FUNCTIONS- SPACE, SUBSTR, UPPER, LOWER, TRIM, LENGTH

AIM:

To implement Space, Substr, Upper, Lower, Trim, LengthString Functionsusing MySQL.

SPACE()

MySQL SPACE() returns the string containing a number of spaces as specified in the argument.

Syntax : SPACE(N)

```
mysql> SELECT 'start', SPACE(20),'end';
+-------+----------------------+-----+
| start | SPACE(20)            | end |
+-------+----------------------+-----+
| start |                      | end |
+-------+----------------------+-----+
1 row in set (0.00 sec)
```

SUBSTRING() / SUBSTR()

MySQL SUBSTRING() returns a specified number of characters from a particular position of a given string.

Syntax : SUBSTRING(str,pos,len)

MySQL SUBSTR() returns the specified number of characters from a particular position of a given string.

SUBSTR() is a synonym for SUBSTRING().

Syntax : SUBSTR(str,pos,len)

```
mysql> SELECT substr("Database",5,4);
+------------------------+
| substr("Database",5,4) |
+------------------------+
| base                   |
+------------------------+
1 row in set (0.00 sec)

mysql> SELECT substr("Database",5);
+----------------------+
| substr("Database",5) |
+----------------------+
| base                 |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT substr("Database",-5);
+-----------------------+
| substr("Database",-5) |
+-----------------------+
| abase                 |
+-----------------------+
1 row in set (0.00 sec)
```

UPPER()

MySQL UPPER() converts all the characters in a string to uppercase characters.

Syntax : UPPER(str)

```
mysql> SELECT UPPER('database');
+-------------------+
| UPPER('database') |
+-------------------+
| DATABASE          |
+-------------------+
```

LOWER() /LCASE()

MySQL **LCASE()** converts the characters of a string to lower case characters.

Syntax : LCASE(str)

MySQL **LOWER**() **converts all** the characters in a string to lowercase characters.

Syntax: **LOWER** (str);

```
mysql> SELECT lower('DATABASE');
+-------------------+
| lower('DATABASE') |
+-------------------+
| database          |
+-------------------+
```

TRIM()

MySQL TRIM() function returns a string after removing all prefixes or suffixes from the given string.

Syntax : TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)

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

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

mysql> SELECT TRIM(LEADING "DATABASE" FROM "DATABASE MANAGEMENT");
+-----------------------------------------------------+
| TRIM(LEADING "DATABASE" FROM "DATABASE MANAGEMENT") |
+-----------------------------------------------------+
|  MANAGEMENT                                         |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

RTRIM()

MySQL RTRIM() removes the trailing spaces from a given string.

Syntax : RTRIM(str)

```
mysql> SELECT RTRIM('Database        ');
+---------------------------+
| RTRIM('Database        ') |
+---------------------------+
| Database                  |
+---------------------------+
1 row in set (0.00 sec)
```

LTRIM(str)

MySQL LTRIM() removes the leading space characters of a string passed as argument.

Syntax : LTRIM(str)

```
mysql> SELECT LTRIM('          Database');
+-----------------------------+
| LTRIM('          Database') |
+-----------------------------+
| Database                    |
+-----------------------------+
1 row in set (0.00 sec)
```

LENGTH()

MySQL LENGTH() returns the length of a given string.

Syntax : LENGTH(str)

```
mysql> SELECT LENGTH('abc');
+---------------+
| LENGTH('abc') |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)

mysql> SELECT LENGTH(123);
+-------------+
| LENGTH(123) |
+-------------+
|           3 |
+-------------+
1 row in set (0.00 sec)
```

BIT_LENGTH()

Returns the length of the string str in bits.

Syntax : BIT_LENGTH(str)

```
mysql> SELECT BIT_LENGTH('abc');
+-------------------+
| BIT_LENGTH('abc') |
+-------------------+
|                24 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT BIT_LENGTH(2);
+---------------+
| BIT_LENGTH(2) |
+---------------+
|             8 |
+---------------+
1 row in set (0.00 sec)
```

CHAR_LENGTH()

Returns the length of the string str, measured in characters. A multi-byte character counts as a single character. This means that for a string containing five 2-byte characters, LENGTH() returns 10, whereas CHAR_LENGTH() returns 5.

Syntax : CHAR_LENGTH(str)

```
mysql> SELECT CHAR_LENGTH('abc');
+--------------------+
| CHAR_LENGTH('abc') |
+--------------------+
|                  3 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT CHAR_LENGTH(123);
+------------------+
| CHAR_LENGTH(123) |
+------------------+
|                3 |
+------------------+
1 row in set (0.00 sec)
```

CONCAT()

Returns the string that results from concatenating one or more arguments. If all arguments are nonbinary strings, the result is a nonbinary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent nonbinary string form.

Syntax : CONCAT(str1,str2,...)

```
mysql> SELECT CONCAT('data','base','system');
+--------------------------------+
| CONCAT('data','base','system') |
+--------------------------------+
| databasesystem                 |
+--------------------------------+
1 row in set (0.00 sec)
```

INSTR()

MySQL INSTR() takes a string and a substring of it as arguments, and returns an integer which indicates the position of the first occurrence of the substring within the string

Syntax : INSTR(str,substr)

```
mysql> SELECT INSTR('database','ab');
+------------------------+
| INSTR('database','ab') |
+------------------------+
|                      4 |
+------------------------+
1 row in set (0.00 sec)

mysql> SELECT INSTR('database','a');
+-----------------------+
| INSTR('database','a') |
+-----------------------+
|                     2 |
+-----------------------+
1 row in set (0.00 sec)
```

LOCATE()

MySQL LOCATE() returns the position of the first occurrence of a string within a string. Both of these strings are passed as arguments. An optional argument may be used to specify from which position of the string (i.e. string to be searched) searching will start. If this position is not mentioned, searching starts from the beginning.Syntax : LOCATE(substr,str,pos)

```
mysql> SELECT LOCATE('ab','database');
+-------------------------+
| LOCATE('ab','database') |
+-------------------------+
|                       4 |
+-------------------------+
1 row in set (0.00 sec)
```

MID()

MySQL MID() extracts a substring from a string. The actual string, position to start extraction and length of the extracted string - all are specified as arguments.

Syntax : MID(str,pos,len)

```
mysql> SELECT MID('Database',5,4);
+---------------------+
| MID('Database',5,4) |
+---------------------+
| base                |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT MID('Database',1,4);
+---------------------+
| MID('Database',1,4) |
+---------------------+
| Data                |
+---------------------+
1 row in set (0.00 sec)
```

POSITION()

MySQL POSITION() returns the position of a substring within a string..

Syntax : POSITION(substr IN str)

```
mysql> SELECT POSITION("ta" IN "Database");
+-----------------------------+
| POSITION("ta" IN "Database") |
+-----------------------------+
|                           3 |
+-----------------------------+
1 row in set (0.00 sec)
```

RESULT: Thus the String Functionssuch as space, substr, upper, lower, trim, length are executed successfully.

Exp No:22

Date:

DATABASE CONNECTIVITY USING PHP AND MYSQL

AIM:

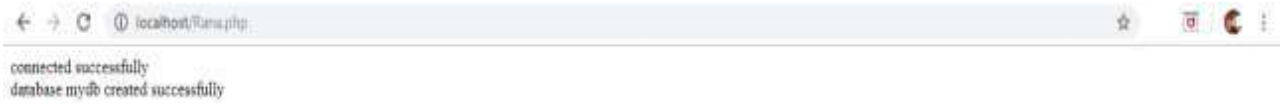
To connect he PHP and MYSQL and to execute the CREATE, INSERT, SELECT command in MySQL.

PROGRAM:

```
<?php
$host=
$password="";
$conn=mysqli_connect($host,$user,$password);
if(!$conn)
{
Die('couldnot connect:',mysql_connect_error());
}
echo"connectsuccessfully('br/>')";
$sql='Create database mydb';
$sql="create table emp(id int,namevarchar(10) NOT NULL,empsalary INT NOT NULL,primary key(id))";
$sql="insert into emp(id,name,empsalary) values(312,RANA,200000)";
$sql="delete from emp where id=1";
$sql="updateemp set empsalary=9000000 where id=312";
if(mysqli_query($conn,$sql))
{echo "operations failed failed",mysqli_error($conn);
}
mysqli_close($conn);
?>
```

OUTPUT:

After creation of database and emp table



After insertion:

After deletion:

After updation:

RESULT: ThustheMysqlconnectedusingPHP and MYSQLandexecutedtheCREATE,INSERT,SELECTcommandinMySQL.

Exp No: 23

Date:

TRAIN TICKET RESERVATION SYSTEM TO RECEIVE TICKETS THROUGH SOCIAL NETWORK.

AIM: To create an online train ticket booking system that generates the ticket and sends it to the WhatsApp number of the customer.

ABSTRACT

This project is developed for making the online train ticket booking more easier and hassle free. A copy of the ticket is sent to the customer's whatsapp number to ensure an easy journey. The system is developed using python in the front end and sqllite for the database management. The system gets the user id and password and validates it. Once it is validated it takes the details like from and to destinations, the journey date, and the class of the ticket. Once the details of the journey are taken, the user enters his personal details like his name, age, gender.
Then these details are stored in the database. Then a ticket copy is displayed on the screen and at the same time a copy of the ticket is mailed to the whatsapp number of the user. This ensures the availability of the ticket copy with the user at any time.

EXISTING SYSTEM:
The existing system does have some disadvantages. Since the ticket copy is only available on the website, it becomes difficult to access it during network disturbances. Hence the customer is forced to have a physical copy of the ticket. Eradication of this problem was the main aim of this project.

PROPOSED SYSTEM:
The new system has some upgradations from the existing system. The user interface has been simplified .Most importantly, the copy of the ticket, apart from just being displayed, it is also sent to the whatsapp number of the customer.

SOFTWARE AND HARDWARE REQUIREMENTS:

- 64bit laptop/desktop
- Latest version of Python
- Sqllite

Modules Required :

- Tkinter
- Pywahtkit
- Time
- Random

PROCEDURE:

- Make sure that the latest version of python and sqllite are installed.
- Make sure that all the required modules are installed – pip install (module name).
- Create the required database and tables.
- Generate a connection between python and sql.
- Import all the required modules.
- Classify the functions based on the program.
- Generate the functions.
- Assemble the functions.
- Call the functions and run the program.

SOURCE CODE:

```
fromtkinter import *
importtkinter.messagebox
importtkinter.font as tkFont
import time
import random
import sqlite3
global  x1,x2,x3,x4
importtkinter.ttk as ttk
#import pywhatkit

global conn, cursor
conn = sqlite3.connect('Railway.db')
c = conn.cursor()

globalLoginId,count
global Password
global  Source
global Destination
global Date
global Name
globalAge,Gender,IdProof
global variable,variable1,variable2,v2,var
globalDepartureTime, TrainNumber, Number
root = Tk()
root.title("Railway reservation")

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
width = 400
height = 400
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry('%dx%d+%d+%d' % (width, height, x, y))
root.resizable(0, 0)
```

```
w = 400
h = 400
canvas = Canvas(root, width=w, height=h)
canvas.config(bg='light blue')
canvas.pack()

my_image = PhotoImage(file='Train.gif')
my_img = canvas.create_image(0, 0, anchor=NW, image=my_image)

Label(root, text="BHAARATH RAILWAYS",font=('Slab Serif',17),bg='Orange').place(x=100,y=150)
Label(root, text="LoginId",font=('Slab Serif',15),bg='green').place(x=60,y=200)
Label(root, text="Password",font=('Slab Serif',15), bg='green').place(x=60,y=240)

entry_1 = Entry(root, font=('Slab Serif',10))
entry_1.place(x=160,y=200,height=30)

entry_2 = Entry(root, font=('Slab Serif',10),show="*")
entry_2.place(x=160,y=240,height=30)

defprintMsg():
    if((entry_1.get()=='dhoni' and entry_2.get()=='844') or (entry_1.get()=='jainam' and entry_2.get()=='844') or
(entry_1.get()=='ayush' and entry_2.get()=='844') ):
tkinter.messagebox.showinfo('login result', 'CONGRATULATIONS!! LOGIN SUCCESSFUL')
createWindow()
else:
tkinter.messagebox.showinfo('login result', 'LOGIN FAILED!:( TRY AGAIN')

defcreateWindow():
root.destroy()
window = Tk()
window.title("Login frame")
customFont = tkFont.Font(family="Helvetica", size=14)
screen_width = window.winfo_screenwidth()
screen_height = window.winfo_screenheight()
width = 410
height = 400
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
window.geometry('%dx%d+%d+%d' % (width, height, x, y))
window.resizable(0, 0)

window.config(bg='purple')

    entry1 =Entry(window,justify='center',font=('Slab Serif',3))
    entry1.place(x=210,y=70)

    entry2 = Entry(window, justify='center',font=('Slab Serif', 3))
    entry2.place(x=210, y=115)
```

```
    entry3 = Entry(window, justify='center',font=('Slab Serif', 3))
    entry3.place(x=210, y=205)


def fun_1(*args):
entry1.insert(10,variable.get())
def fun_2(*args):
entry2.insert(10,variable1.get())
def fun_3(*args):
entry3.insert(10,variable2.get())

variable = StringVar(window)
choices = {'Howrah', 'Ajmer'}
variable.set('Select')
variable.trace("w", fun_1)
popupMenu = OptionMenu(window, variable, *choices)
popupMenu.place(x=210, y=70,width=100)
popupMenu.config(font=('Slab Serif',15),bg="yellow")
Label(window, text="Source",font=('Slab Serif',14), bg="cyan").place(x=110,y=70,width=80)

    variable1 = StringVar(window)
trains = {'New Delhi','Chandigarh'}
variable1.set('Select')
variable1.trace("w", fun_2)

    popupMenu1 = OptionMenu(window, variable1, *trains)
Label(window, text="Destination",font=('Slab Serif',15), bg="cyan").place(x=100,y=115,width=100)
popupMenu1.config(font=('Slab Serif',14),bg="yellow")
    popupMenu1.place(x=210,y=115,width=130)

    variable2 = StringVar(window)
classes = {'1A','2A','3A'}
variable2.set('Select')
variable2.trace("w", fun_3)

    popupMenu1 = OptionMenu(window, variable2, *classes)
Label(window, text="class", font=('Slab Serif', 15), bg="cyan").place(x=100, y=205, width=100)
popupMenu1.config(font=('Slab Serif', 14), bg="yellow")
    popupMenu1.place(x=210, y=205, width=130)

Label(window,text="Date",font=('Slab Serif',15), bg="cyan").place(x=110,y=160,width=80)
    Date=StringVar()
    e1=Entry(window,textvariable=Date)


def Check():
        if len(e1.get()) == 0 or len(entry1.get()) == 0 or len(entry2.get()) == 0 or len(entry3.get()) == 0:
tkinter.messagebox.showinfo('Error','enter all required fields!')
else:
Check1()
def Check1():
```

```
if (entry1.get() == "Ajmer" and entry2.get() == "Chandigarh" and entry3.get() == "3A") or (entry1.get() ==
"Ajmer" and entry2.get() == "New Delhi" and entry3.get() == "3A"):
tkinter.messagebox.showinfo('Error','Sorry Train Unavailable!')
else:
ops = e1.get()

        len1 = len(ops)
if len1==10:
            date1=ops[0]+ops[1]
            month1=ops[3]+ops[4]
            year11=ops[6]+ops[7]+ops[8]+ops[9]


if(len1==10):
if(int(date1)<=31 and int(date1)>=1):
if(int(month1)>=1 and int(month1)<=12):
if(int(year11)>=2017 and int(year11)<=2018):
print('')
else:
tkinter.messagebox.showinfo('Error', 'Enter Year Correctly!')
else:
tkinter.messagebox.showinfo('Error', 'Enter Month Correctly!')
else:
tkinter.messagebox.showinfo('Error', 'Enter date Correctly!')
window.destroy()
Search()
else:
tkinter.messagebox.showinfo('Error', 'Enter date Correctly!')

        #print('list1 print',list1)
        #window.destroy()

        #Search()
```

DATABASE:

TABLE 1 : TRAIN DETAILS

	Trainnumber	Source	Destination	Departure	Arrival	Trainname
	Filter	Filter	Filter	Filter	Filter	Filter
1	12235	Howrah	New Delhi	14:30	7:55	Rajdhani Express
2	12236	Howrah	New Delhi	16:30	5:50	Howrah Juntion
3	12237	Howrah	New Delhi	8:35	15:50	New Delhi Duronto
4	12238	Howrah	New Delhi	12:30	7:20	Anand Vihar
5	12239	Howrah	Chandigarh	6:30	18:20	Howrah Amritsar Express
6	12240	Howrah	Chandigarh	14:30	23:30	Kalka Mail
7	12241	Howrah	Chandigarh	12:30	8:40	JallianwalaBagh Express
8	12242	Howrah	Chandigarh	8:30	16:20	Durgiana Express
9	12243	Ajmer	Chandigarh	9:30	14:40	Garibrath
10	12244	Ajmer	Chandigarh	1:00	00:45	Mumbai Bandra (T)
11	12245	Ajmer	Chandigarh	11:05	3:00	Pooja SF Express
12	12246	Ajmer	Chandigarh	14:05	6:45	Gandhidham
13	12247	Ajmer	New Delhi	1:40	10:40	Uttaranchal Express
14	12248	Ajmer	New Delhi	6:30	10:45	Rajkot Express
15	12249	Ajmer	New Delhi	11:05	20:00	Corbet park link Express
16	12250	Ajmer	New Delhi	12:10	21:10	Ranikhet Express
17	12251	Pune	Nagpur	06:00	21:30	Nagpur Garibrath
18	12252	Mumbai	Ahmedabad	12:00	08:30	Duronto Express
19	12253	Dehradun	Indore	05:50	12:30	Dehradun Express
20	12254	Mysore	Bengaluru	03:00	22:00	Bangalore PASSR

TABLE 2 : PASSENGER DETAILS

	pnr	Name	Gender	Age	class
	Filter	Filter	Filter	Filter	Filter
1		priti	8	Male	NULL
2		ooo	9	Male	NULL
3		p	9	Male	NULL
4	500602	p	9	Female	NULL
5	719889	pp	88	Male	NULL
6	538807	l	0	Male	NULL
7	581794	k	1	Female	NULL
8	894560	ll	0	Female	NULL
9	197302	ll	0	Female	NULL
10	662954	p	9	Male	NULL
11	391778	pp	99	Male	NULL
12	180644	kk	7	Male	NULL
13	133909	prt	9	Male	NULL
14	288960	p	8	Female	NULL
15	246671	p	9	Female	NULL
16	179982	o	1	Female	NULL
17	820319	p	98	Male	NULL
18	816111	pp	9	Male	NULL
19	437391	uu	9	Male	NULL
20	482881	mi	8	Female	NULL
21	324453	kk	99	Female	NULL
22	586951	i	9	Male	NULL
23	148731	ji	0	Female	NULL

OUTPUT:

IMAGE 1 : LOGIN PAGE

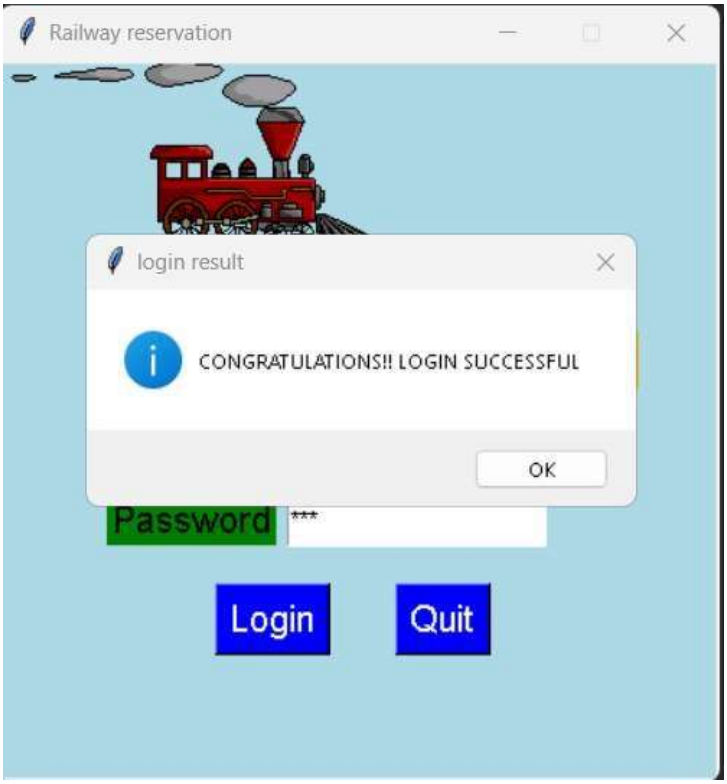


IMAGE 3 : DETAILS OF THE JOURNEY

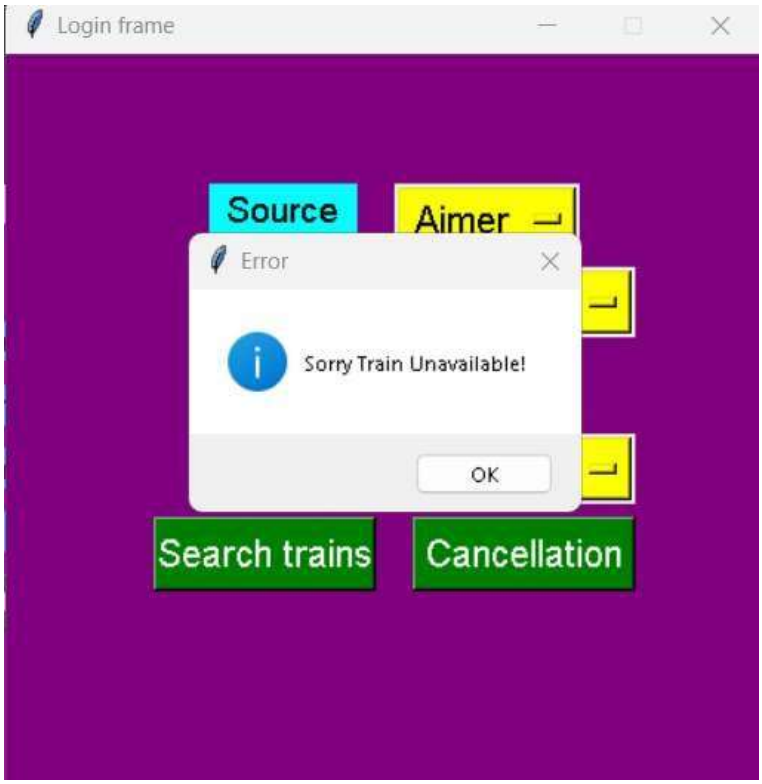


IMAGE 4 : DISPLAYING UNAVAILABILITY OF TRAINS

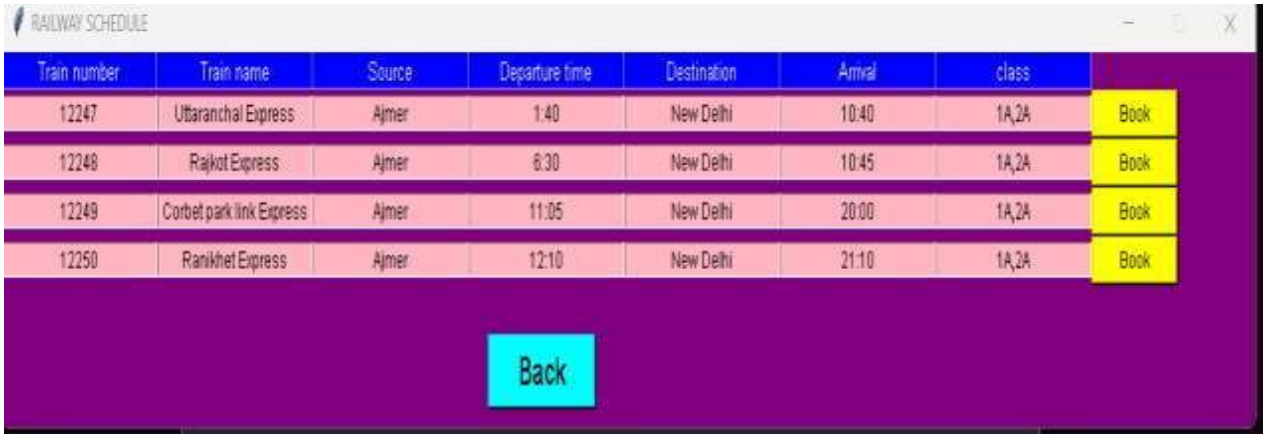


IMAGE 5 : DISPLAYING AVAILABLE TRAINS

IMAGE 6 : PASSENGER DETAILS

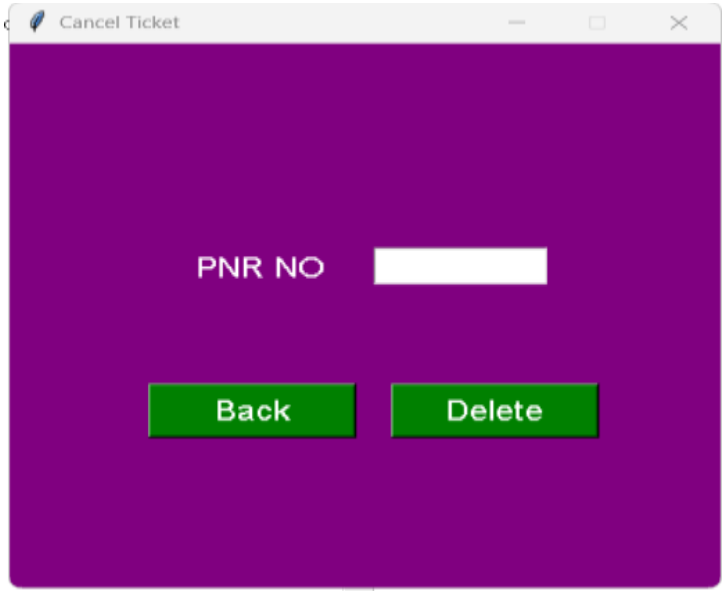


IMAGE 8 : CANCELLATION

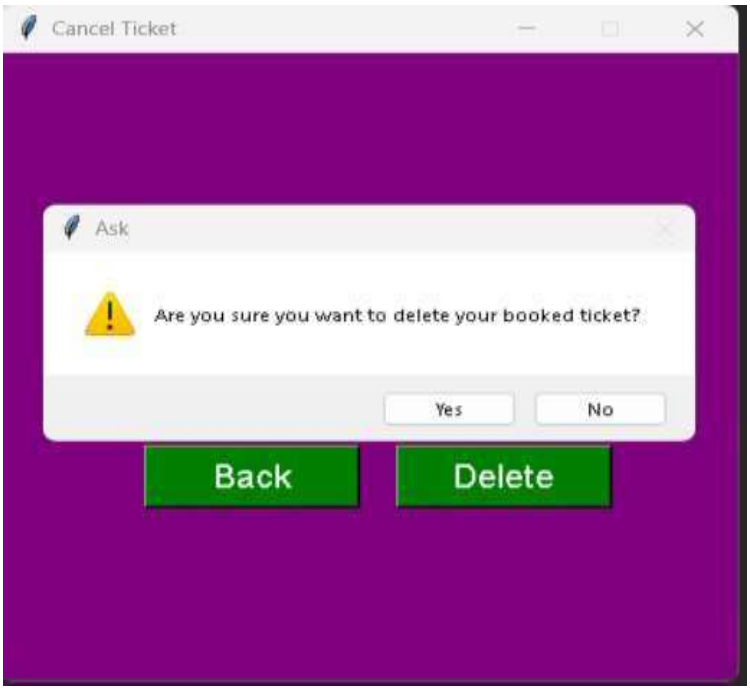


IMAGE 9: APPROVAL FOR

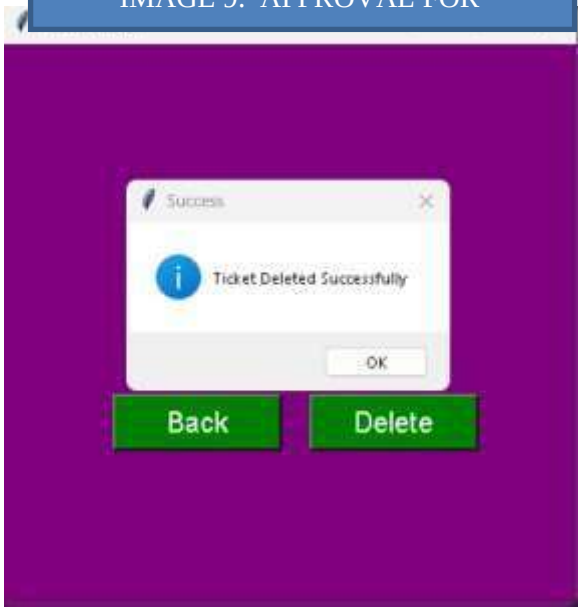


IMAGE 10 : DELETION SUCCESSFUL

Exp No: 24

Date:

COLLEGE ADMISSION FORM

AIM:

To create an online college admission form that generates an admission number for the candidate.

ABSTRACT:

This form is developed to create a simple college admission form where the candidate can enter the details and edit it afterwards using his id number. The system is developed using python in the front end and mysql for the database management. Once the user goes into the app, he has two options

1. To create a new login and get a new id number by entering the required details, and
2. To make changes to the already entered details by giving his id number. Then these details are stored in the database.

EXISTING SYSTEM:

The existing system does have some disadvantages. The id number is not displayed to the user immediately.

PROPOSED SYSTEM:

The new system has some upgradations from the existing system. The user interface has been simplified.The unique user id is displayed once the user enters his details and logs in.

SOFTWARE AND HARDWARE REQUIREMENTS:

- 64bit laptop/desktop
- Latest version of Python
- mysql

MODULES REQUIRED :

- Tkinter
- Time
- Random

PROCEDURE:

- Make sure that the latest version of python and sqllite are installed.
- Make sure that all the required modules are installed – pip install (module name).
- Create the required database and tables.

- Generate a connection between python and sql.
- Import all the required modules.
- Classify the functions based on the program.
- Generate the functions.
- Assemble the functions.
- Call the functions and run the program.

SOURCE CODE:

```
importtkinter as tk
importtkinter.messagebox as mb
import random
importtkinter.ttk
fromtkinter import *
importmysql.connector as mysql


def signup():
root.destroy()
entrypage = Tk()
entrypage.geometry('1000x1000')
entrypage.title("ADMISSION FORM")
admission=Frame(entrypage,bd=20, bg='black', relief=SOLID,padx=10,pady=10)


  label_0 = Label(entrypage, text="NEW ADMISSION",width=20,font=("bold", 20))
  label_0.place(x=350,y=50)

  label_1 = Label(entrypage, text="First Name",width=20,font=("bold", 10))
  label_1.place(x=300,y=150)
firstname = Entry(entrypage,font=f)
firstname.place(x=500,y=150)

  label_2 = Label(entrypage, text="lastname",width=20,font=f)
  label_2.place(x=300,y=200)
lastname = Entry(entrypage,font=f)
lastname.place(x=500,y=200)

  label_3 = Label(entrypage, text="Age",width=20,font=f)
  label_3.place(x=300,y=250)
age = Entry(entrypage,font=f)
age.place(x=500,y=250)
#Radiobutton(root, text="Male",padx = 5, variable=var, value=1).place(x=235,y=230)
#Radiobutton(root, text="Female",padx = 20, variable=var, value=2).place(x=290,y=230)

  label_4 = Label(entrypage, text="Gender",width=20,font=("bold", 10))
  label_4.place(x=300,y=300)
  gender=' '
Radiobutton(entrypage, text="Male",padx = 5, variable=gender, value='M').place(x=485,y=300)
Radiobutton(entrypage, text="Female",padx = 20, variable=gender, value='F').place(x=550,y=300)

Label(entrypage, text="Date of Birth",width=20,font=("bold", 10)).place(x=300,y=350)
```

```
dob=Entry(entrypage,font=f).place(x=500,y=350)

Label(entrypage, text="Community",width=20,font=("bold", 10)).place(x=300,y=400)
   list1 = ['OC','BC','MBC','SC','ST'];
community=StringVar()
droplist=OptionMenu(entrypage,community,  *list1)
droplist.config(width=20)
community.set('select your community')
droplist.place(x=500,y=400)

Label(entrypage, text="Select your state",width=20,font=("bold", 10)).place(x=300,y=450)
   States=["Andhra Pradesh","Arunachal Pradesh ","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana","Himachal
Pradesh","Jammu and Kashmir","Jharkhand","Karnataka","Kerala","Madhya
Pradesh","Maharashtra","Manipur","Meghalaya","Mizoram","Nagaland","Odisha","Punjab","Rajasthan","Sikkim","Tamil
Nadu","Telangana","Tripura","Uttar Pradesh","Uttarakhand","West Bengal","Andaman and Nicobar Islands","Chandigarh","Dadra and
Nagar Haveli","Daman and Diu","Lakshadweep","National Capital Territory of Delhi","Puducherry"]
state=StringVar()
droplist=OptionMenu(entrypage,state, *States)
droplist.config(width=20)
community.set('select your State')
droplist.place(x=500,y=450)

   label_4 = Label(entrypage, text="Subject Studied in class 12",width=20,font=("bold", 10))

   label_4.place(x=300,y=500)
subjects= [ ]
Checkbutton(entrypage, text="Maths", variable=subjects).place(x=500,y=525)
Checkbutton(entrypage, text="Physics", variable=subjects).place(x=500,y=550)
Checkbutton(entrypage, text="Chemistry", variable=subjects).place(x=500,y=575)
Checkbutton(entrypage, text="Biology", variable=subjects).place(x=500,y=600)
Checkbutton(entrypage, text="Computer Science", variable=subjects).place(x=500,y=625)
Checkbutton(entrypage, text="English", variable=subjects).place(x=500,y=650)

Label(entrypage, text="Total Marks Scored in 12th",width=20,font=("bold", 10)).place(x=700,y=600)
marks=Entry(entrypage,font=f).place(x=900,y=600)
```

DATABASE:

TABLE 1 :The Student table

Field	Type	Null	Key	Default	Extra
adno	decimal(10,0)	NO	PRI	NULL	
fullname	varchar(30)	YES		NULL	
gender	varchar(4)	YES		NULL	
dob	date	YES		NULL	
community	varchar(5)	YES		NULL	
state	varchar(25)	YES		NULL	
subjects	varchar(100)	YES		NULL	
marks	int	YES		NULL	

OUTPUT:

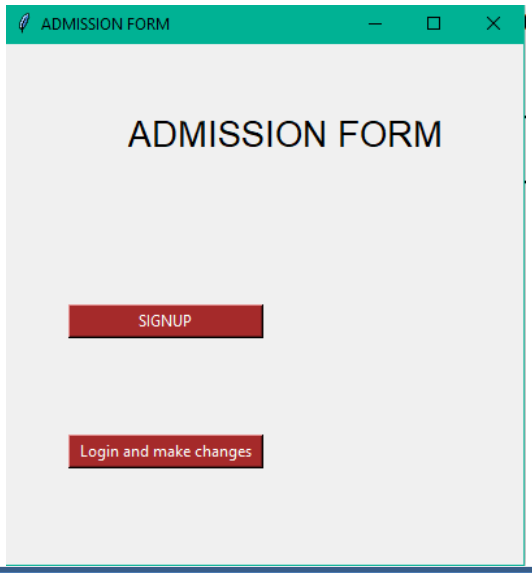


IMAGE 1 :ENTRY PAGE

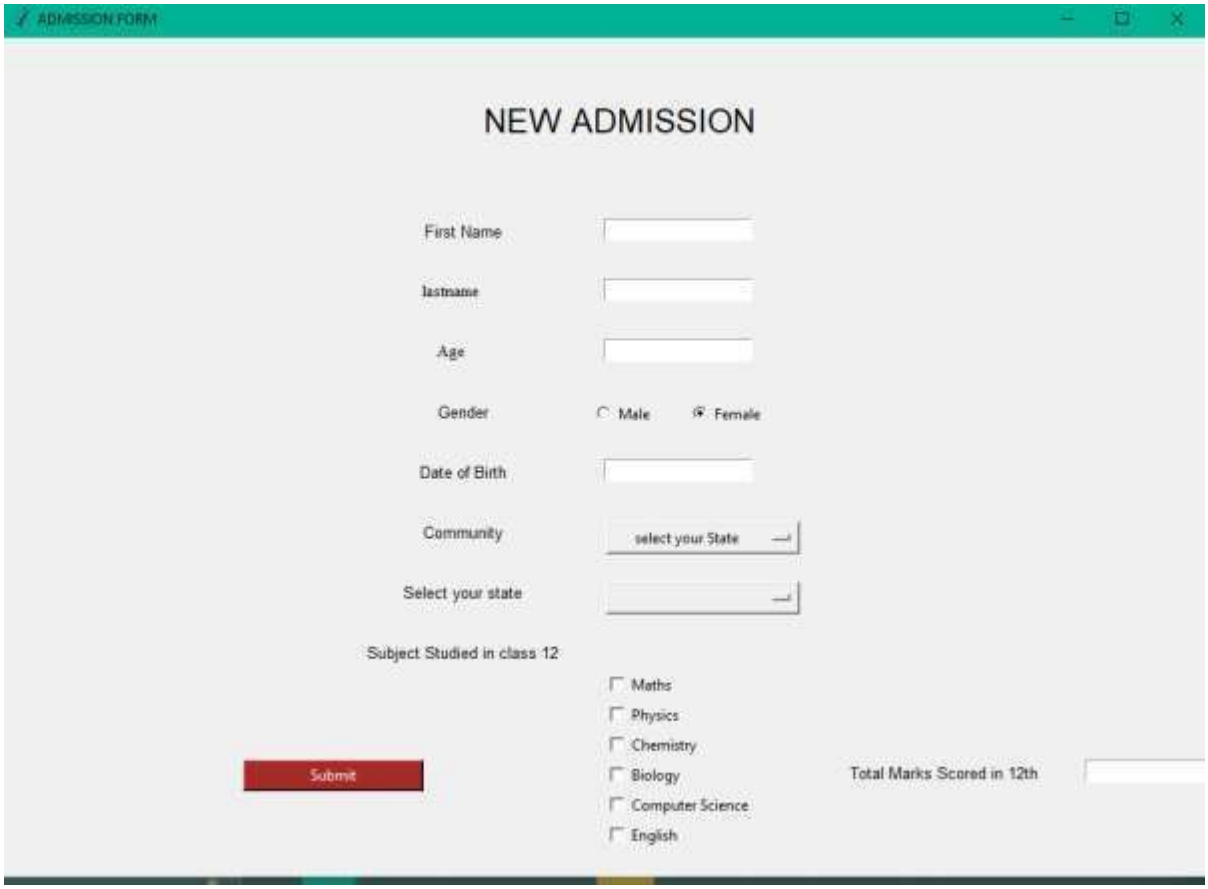


IMAGE 2 :SIGNUP PAGE

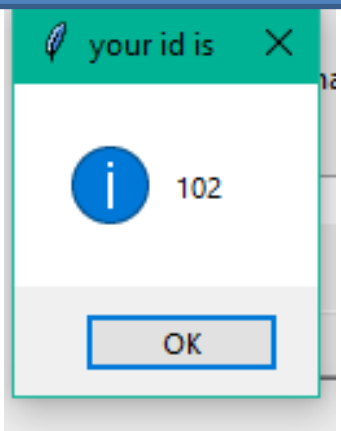


IMAGE 3 :UNIQUE ID GENERATION

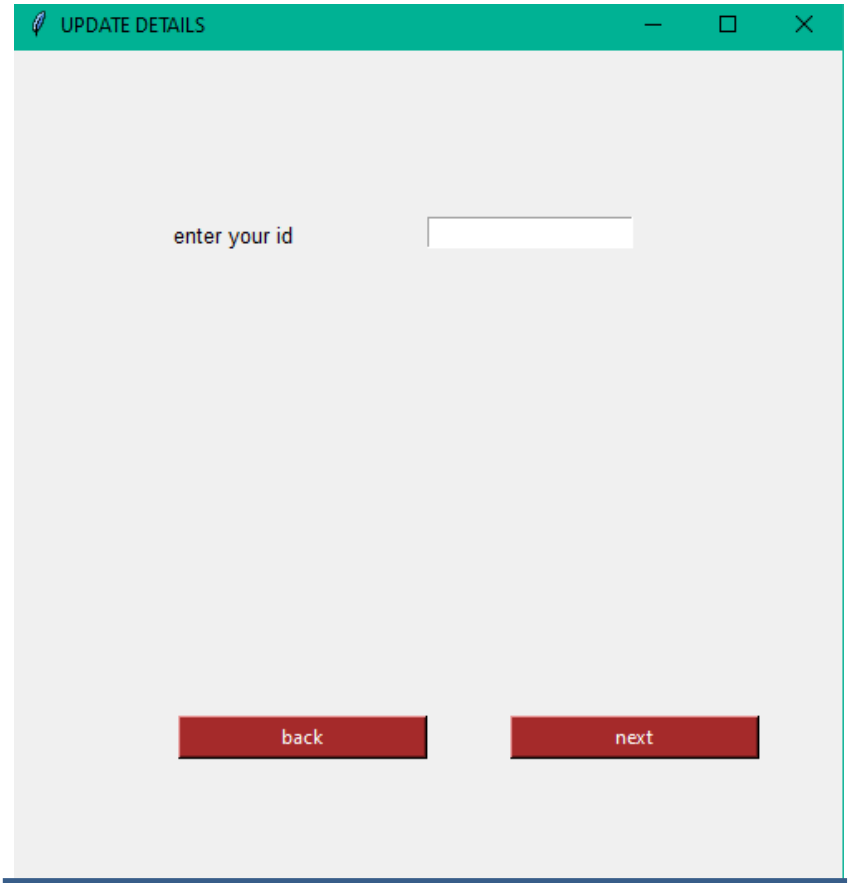


IMAGE 4 : ENTER ID TO UPDATE

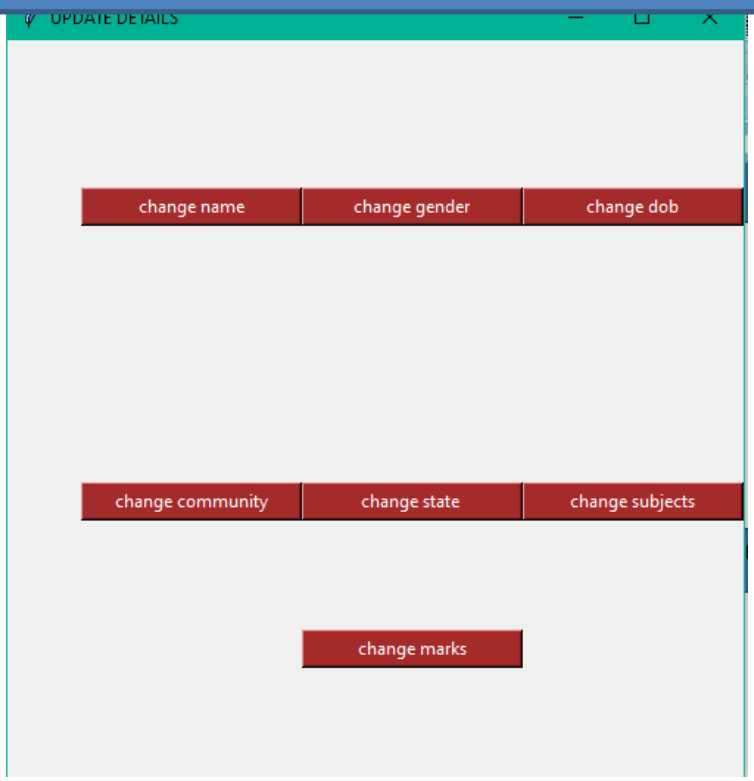


IMAGE 5 :DATA UPDATION OPTIONS

Exp No: 24

Date:

QR ENABLED AUTOMATIC BUS TICKET BOOKING SYSTEM

S. VIJAY SARANGESHWAR

192224073

ARTFICIAL INTELLIGENCE AND DATA SCIENCE

G. MUKESH

19211309

COMPUTER SCIENCE AND ENGINEERING

IMPLEMENTATION USING PYTHON AND DATABASE MANAGEMNET SYSTEM EQUIPPED WITH SQL (STRUCTURED QUERY LANGUAGE)

QR ENABLED AUTOMATIC BUS TICKET BOOKING SYSTEM

AIM:

The main objective of this project is to enable the passengers to travel with ease and make their journey memorable. This project can be used to book tickets with ease and hassle-free and also generates a QR-code that makes it easier for the passengers to board the vehicle without much difficulty.

ABSTRACT:
This project is the implementation of QR code scanning on the regular bus system such that the passengers have a hassle-free journey. The structure of the project is designed in such a way that it suits the very kind of people who have not used highly complex interfaces and also suits elderly people for whom the interface needs to be as simple as possible. This online application is modified in such a way that anyone can access it at any point of time. Thereby, making it very useful for people who have very less time.
The user can enter their name, age and password that they have assigned and select the very operation that they would want to perform. Since, the interface is easy to understand , it makes it easier for the user to select the options.
The application as a response would generate a QR code that contains all the information about the ticket and can be used a digital ticket by the user. The user also has other options cancel the ticket , book the ticket , select the mode of transaction etc.

HARDWARE AND SOFTARE REQUIRED:
Hardware:32 or 64-bit laptop equipped with at least 4 GB RAM and 512 GB internal storage.
Software: Latest python version(3.11.2) and MYSQL software of latest version
Modules required: “Tkinter”, qrcode ,PIL,random , mysql-connector
.

EXISTING SYSTEM:
The existing system has a lot of loopholes in it . The transaction is mostly offline therefore , making it a hassle to look for change if they need to, as compared to the online , where you can just enter the amount and the process is done. Finding a seat can also be a hassle for the passengers especially elders . The timing of the bus is not scheduled properly, therefore making it possible to miss the bus or wait for a very long time.

PROPOSED SYSTEM:
The updated system eradicates almost all the problems faced in the existing system , thereby making it more relevant. As the transaction online the passengers do not need to worry about having change

all the time while travelling. Also since the seat is being booked beforehand the passenger need not worry about searching for a seat. And finally, since the bus is scheduled the passengers need not worry about missing the bus or waiting for a long time.

PROCEDURE:

- Make sure that the system is running the latest version by using the following command in the command prompt ”python --version” and ”mysql –version” before starting the project make sure that the project has all the required software.
- Once the installation is done, open pycharm (IDE for python) and create a new project named “bus ticket booking system” and create a python file and name it ”bus ticket
- book”.
- Establish connection between mysql and python interface by creating an instance
- of the name ”conn” and giving it all the required data’s like the :username,
- password, port number.
- Import all the modules that are in the required section and make sure that all the modules are imported error-free.
- Classify the functions based on the program needed and make sure to implement all the functions need fully according to the functionality of the program.
- Assemble the functions.
- Call the functions.
- Make sure the program is error free and all the necessary functions all called.
- Run the program.

SOURCE CODE

```
import  pyqrcode
import tkinter as tk
import mysql.connector
from tkinter import messagebox
from PIL import Image
from PIL import ImageTk
def perform_operations():

def cancel_operation():
pass

def generate_qr_code():
pass

def display_ticket():
pass

root = tk.Tk()
def insert_passenger_data():
def submit():
first_name = first_name_entry.get()
last_name = last_name_entry.get()
password = password_entry.get()
confirm_password = confirm_password_entry.get()

if password != confirm_password:
error_label.config(text="Passwords do not match!")
return


conn = mysql.connector.connect(
host="localhost",
user="root",
password="Kutta@123",
database="vijay"
)


cursor = conn.cursor()
```

```
create_table_query = """
        CREATE TABLE IF NOT EXISTS passenger1 (
            id INT AUTO_INCREMENT PRIMARY KEY,
first_nameVARCHAR(255),
last_nameVARCHAR(255),
            password VARCHAR(255)
        )
        """
cursor.execute(create_table_query)


insert_query = """
        INSERT INTO passenger2 (first_name, last_name, password)
        VALUES (%s, %s, %s)
        """
        data = (first_name, last_name, password)
cursor.execute(insert_query, data)


conn.commit()
conn.close()


first_name_entry.delete(0, tk.END)
last_name_entry.delete(0, tk.END)
password_entry.delete(0, tk.END)
confirm_password_entry.delete(0, tk.END)

error_label.config(text="Data saved successfully!")

login_choice = tk.messagebox.askyesno("Login", "Do you want to login?")
        if login_choice:
login()

    def cancel_operation():
        def submit():
entered_first_name = first_name_entry.get()
entered_password = password_entry.get()


            conn = mysql.connector.connect(
                host="localhost",
```

```
            user="root",
            password="Kutta@123",
            database="vijay"
        )

        cursor = conn.cursor()


select_query = """
        SELECT * FROM passenger2
        WHERE first_name = %s AND password = %s
        """
        data = (entered_first_name, entered_password)
cursor.execute(select_query, data)
        result = cursor.fetchone()

        if result:

delete_query = """
            DELETE FROM passenger2
            WHERE first_name = %s
            """
cursor.execute(delete_query, (entered_first_name,))
conn.commit()
messagebox.showinfo("Cancel", "Operation Cancelled Successfully")
        else:
messagebox.showinfo("Cancel", "Invalid Credentials")


conn.close()


      window = tk.Tk()
window.title("Cancel Operation")
window.geometry("300x200")


tk.Label(window, text="First Name:").pack()
first_name_entry = tk.Entry(window)
first_name_entry.pack()

tk.Label(window, text="Password:").pack()
password_entry = tk.Entry(window, show="*")
```

```
password_entry.pack()

submit_button = tk.Button(window, text="Submit", command=submit)
submit_button.pack()

window.mainloop()

    def generate_qr_code():
        def open_image():
image.show()

        def generate():
first_name = first_name_entry.get()
last_name = last_name_entry.get()


            data = f"First Name: {first_name}\nLast Name: {last_name}"

qr = pyqrcode.create(data)


qr_path = "qr_code.png"
qr.png(qr_path, scale=10)


            global photo
            global image
            image = Image.open(qr_path)
            photo = ImageTk.PhotoImage(image)

qr_label.configure(image=photo)
qr_label.image = photo


messagebox.showinfo("Success", "QR code generated successfully!")


        window = tk.Tk()
window.title("Generate QR Code")


tk.Label(window, text="First Name:").pack()
first_name_entry = tk.Entry(window)
```

```
first_name_entry.pack()

tk.Label(window, text="Last Name:").pack()
last_name_entry = tk.Entry(window)
last_name_entry.pack()


generate_button = tk.Button(window, text="Generate QR Code", command=generate)
generate_button.pack()


qr_label = tk.Label(window)
qr_label.pack()


        open_button = tk.Button(window, text="Open Image", command=open_image)
        open_button.pack()


        window.mainloop()


        generate_qr_code()

            def display_ticket():
        print("Display Ticket operation selected")


            def perform_operations():
                root = tk.Tk()

                def on_button_click(operation):
        root.destroy()
        operation()

                label = tk.Label(root, text="Available Operations:")
        label.pack()

        cancel_button = tk.Button(root, text="Cancel", command=lambda:
        on_button_click(cancel_operation))
        cancel_button.pack()
```

```
qr_button = tk.Button(root, text="Generate QR Code", command=lambda:
on_button_click(generate_qr_code))
qr_button.pack()

ticket_button = tk.Button(root, text="Display Ticket", command=lambda:
on_button_click(display_ticket))
ticket_button.pack()

root.mainloop()
   def login():
      def submit():
entered_first_name = first_name_entry.get()
entered_last_name = last_name_entry.get()
entered_password = password_entry.get()

         # Connect to the MySQL database
         conn = mysql.connector.connect(
            host="localhost",
     user="root",
     password="Kutta@123",
     database="vijay"
  )


  cursor = conn.cursor()


select_query = """
  SELECT * FROM passenger2
  WHERE first_name = %s AND last_name = %s AND password = %s
  """
  data = (entered_first_name, entered_last_name, entered_password)
cursor.execute(select_query, data)
  result = cursor.fetchone()

  if result:
messagebox.showinfo("Login", "Login Successful")
perform_operations()
  else:
messagebox.showinfo("Login", "Login Unsuccessful")


conn.close()
```

```
        window = tk.Tk()
window.title("Login")
window.geometry("300x200")

tk.Label(window, text="First Name:").pack()
first_name_entry = tk.Entry(window)
first_name_entry.pack()

tk.Label(window, text="Last Name:").pack()
last_name_entry = tk.Entry(window)
last_name_entry.pack()


tk.Label(window, text="Password:").pack()
password_entry = tk.Entry(window, show="*")
password_entry.pack()

submit_button = tk.Button(window, text="Submit", command=submit)
submit_button.pack()


window.mainloop()


    window = tk.Tk()
window.title("Passenger Details")
window.geometry("300x250")


tk.Label(window, text="First Name:").pack()
first_name_entry = tk.Entry(window)
first_name_entry.pack()

tk.Label(window, text="Last Name:").pack()
last_name_entry = tk.Entry(window)
last_name_entry.pack()

tk.Label(window, text="Password:").pack()
password_entry = tk.Entry(window, show="*")
password_entry.pack()
```

```
tk.Label(window, text="Confirm Password:").pack()
confirm_password_entry = tk.Entry(window, show="*")
confirm_password_entry.pack()

    # Create the submit button
submit_button = tk.Button(window, text="Submit", command=submit)
submit_button.pack()

    # Create error label
error_label = tk.Label(window, text="")
error_label.pack()

    # Run the Tkinter event loop
window.mainloop()



    def generate_qr_code():
print("Generate QR Code operation selected")
        # Perform QR code generation operation

    def display_ticket():
print("Display Ticket operation selected")
        # Perform ticket display operation

    def perform_operations():
                root = tk.Tk()

                def on_button_click(operation):
            root.destroy()
            operation()

                label = tk.Label(root, text="Available Operations:")
            label.pack()

            cancel_button = tk.Button(root, text="Cancel", command=lambda:
            on_button_click(cancel_operation))
            cancel_button.pack()

            qr_button = tk.Button(root, text="Generate QR Code", command=lambda:
            on_button_click(generate_qr_code))
            qr_button.pack()
```

```
ticket_button = tk.Button(root, text="Display Ticket", command=lambda:
on_button_click(display_ticket))
ticket_button.pack()

root.mainloop()

    def generate_qr_code():
print("Generate QR Code operation selected")


    def display_ticket():
print("Display Ticket operation selected")


    def perform_operations():
        root = tk.Tk()

        def on_button_click(operation):
root.destroy() # Close the GUI window
operation()  # Call the selected operation function

        label = tk.Label(root, text="Available Operations:")
label.pack()

cancel_button = tk.Button(root, text="Cancel", command=lambda:
on_button_click(cancel_operation))
cancel_button.pack()

qr_button = tk.Button(root, text="Generate QR Code", command=lambda:
on_button_click(generate_qr_code))
qr_button.pack()

ticket_button = tk.Button(root, text="Display Ticket", command=lambda:
on_button_click(display_ticket))
ticket_button.pack()

root.mainloop()


insert_passenger_data()
```

OUTPUTS

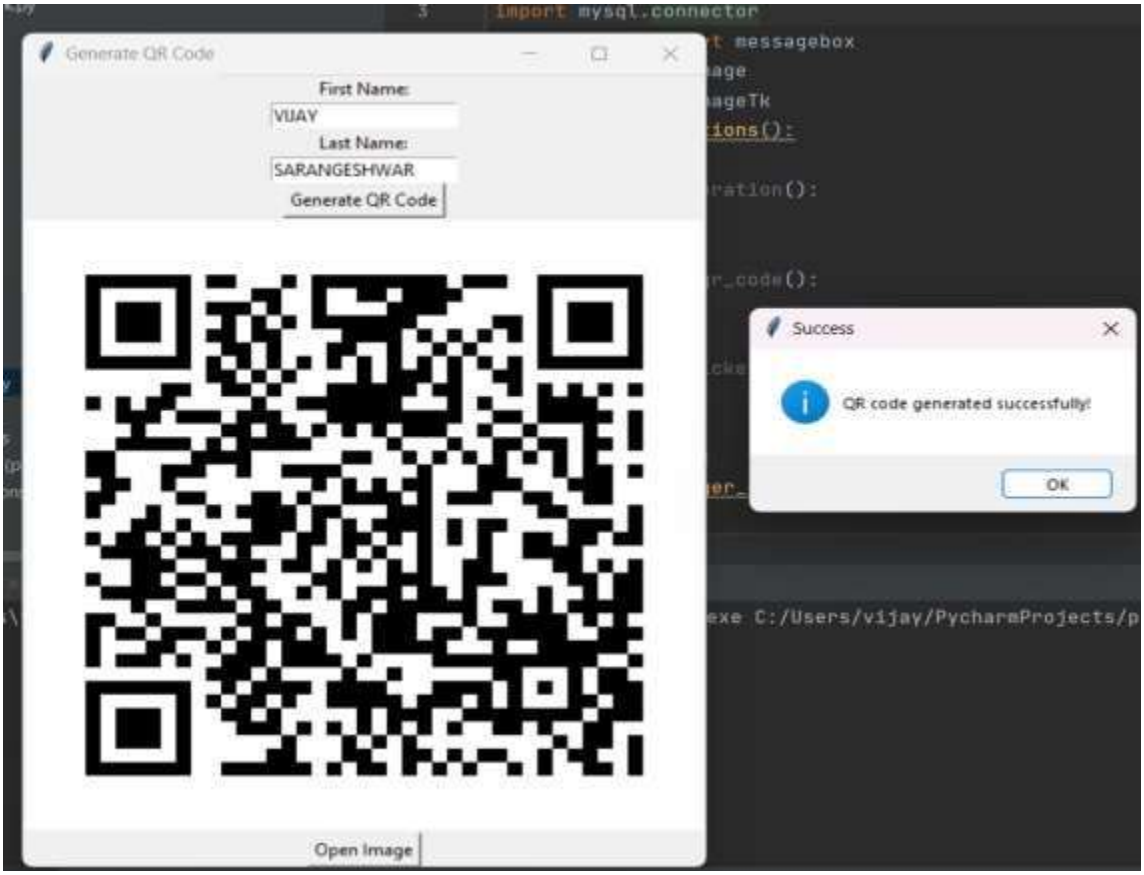


Fig 1-QR-CODE GENERATED

Fig 2-QR-CODE GENERATION PAGE

Fig 3- SIGN UP CONFIRMATION PAGE

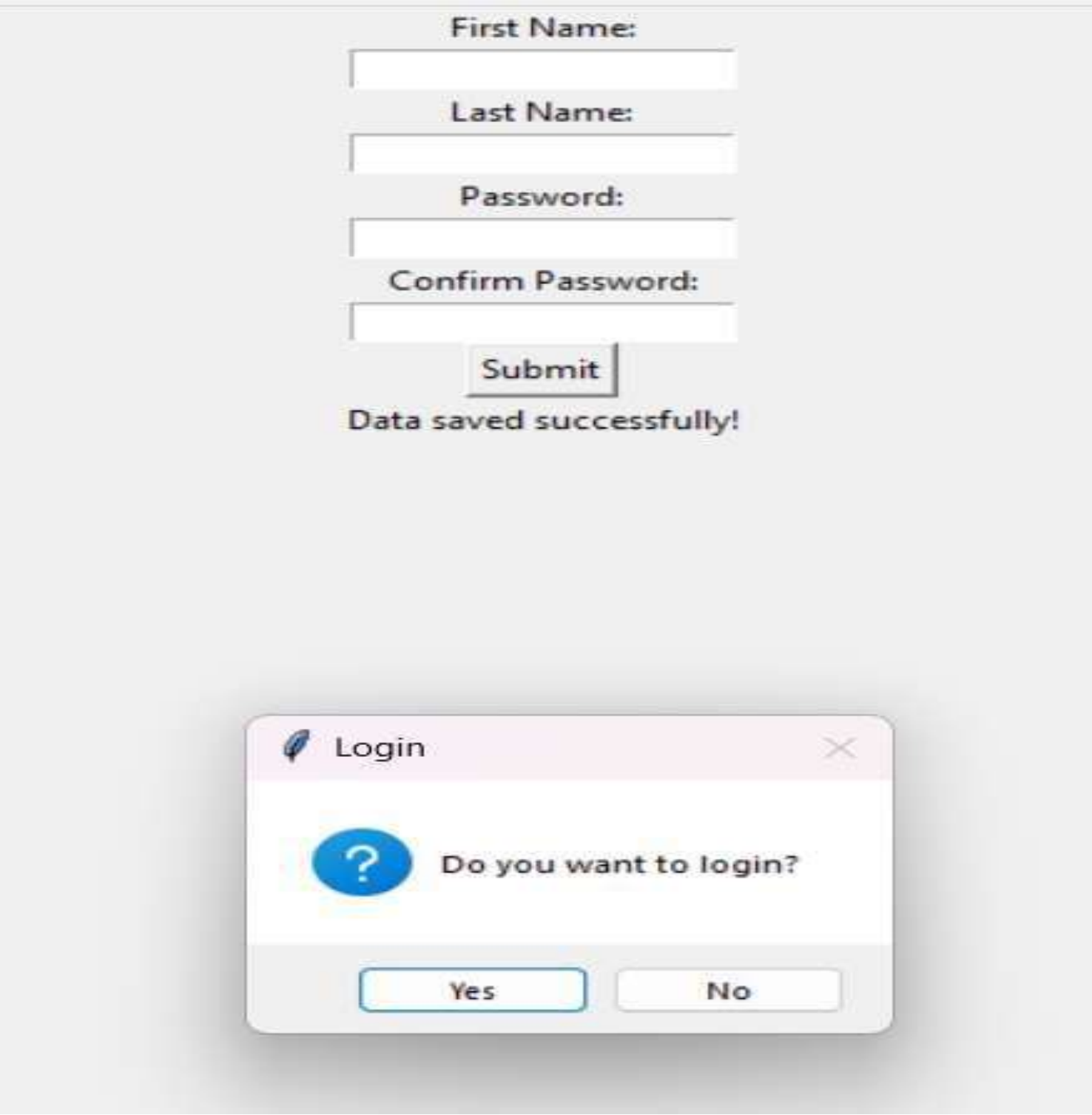


Fig 4-LOGIN PAGE

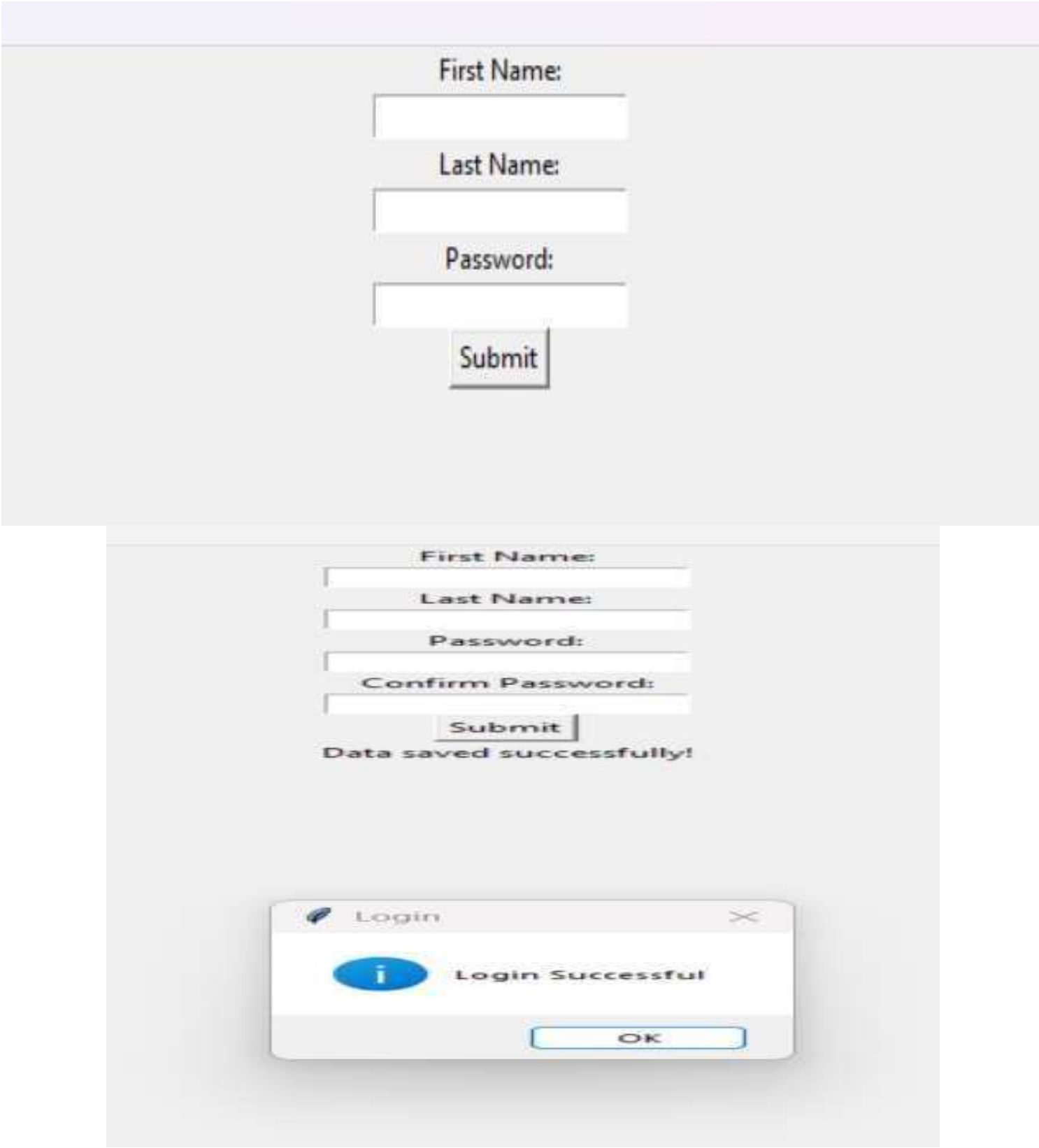


Fig 6-SIGNUP PAGE

Fig 7 OPERATIONS AVAILABLE PAGE

Fig 8- TICKET CANCELLATION PAGE

SQL TABLE NAME -PASSENGER2

id	first_name	last_name	password
1	vijay	sara	oil
2	hjeb	dhjd	oil
3	kln	skjk	oil
4	dhkjf	skjhf	oil
5	hjhk	ghgh	oil
6	kji	jijij	oil
7	uiui	rtrt	oil
8	jiji	okok	oil
9	vjvj	hjhj	oil
10	jij	kkk	oil
11	vijay	jiji	oil
12	vjjv	hjhj	oil
13	ghgh	hghg	oil
14	vjvj	hghg	ilo
15	hghgh	gfgf	121
16	vijay	sarangeshwar	iol
17	vijay	erer	oil
18	vij	wewe	oil
19	vijay	sae	leo
20	vijy	erer	oil
21	viju	huju	oil

Fig. 8 SQL TABLE OF PASSENGER2 THSAT CONTAINS INFO ABOUT THE PASSENGER