Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING

Plot No. 46/46A, Sector No. 5, Behind MSEB Sub Station,
Kharghar, Navi Mumbai - 410 210.

Certificate

Certified that Mr./Ms. Khan Armaan Tufail

Class SE *Roll No.* 18 *Course* DBMS

Exam No. DS4028 *has completed the required number of Practical / Term work / Sessional in the subject in the Department of* Computer Science And Engineering (Data Science) *during the academic Year* 2021-2022

Prof. Vijay Kapure
Lecturer Incharge

Prof. Shraddha Subhedar
Head of the Department

Date : 28/04/2022

Principal

Experiment List:

SR. NO.	Name of Experiment	Date
1	Definition and Draw ER/EER diagram	17/1/22
2	Design a Relational model with case study	9/2/22
3	To Perform basic DDL (Data Definition Language) commands on SQL	7/2/22
4	To Perform basic DML (Data Manipulation Language)commands on SQL	14/2/22
5	To Perform basic DCL (Data control Language) and TCL (Transaction control Language) commands on SQL	7/3/22
6	To Perform basic TCL (Transaction control Language) commands on SQL	21/3/22
7	To Perform DRL (data retrieval commands) ,clauses and aggregate functions	4/4/22
8	To Perform SET operations in SQL	11/4/22
9	To Perform join operations in SQL	18/4/22
10	Mini-project	18/4/22

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

Computer science and Engineering

Data Science Department

Name of Faculty : Prof. Vijay R. Kapure

Class : S. E. 4th Semester

Subject : Data Base Management Systems

Year & Semester : Even Semester 2021-22

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

Computer science and Engineering

Data Science Department

Lab Manual

Subject: Data Base Management Systems

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

CSE-AI and DS Department

Experiment No. 1

Aim: Write a Problem Definition and Draw ER/EER diagram.

Resources Required: H/W :- P4 machine

S/W :- Oracle 10g

Theory:

An entity-relationship diagram is a data modelling technique that creates a graphical

representation of the entities, and the relationships between entities, within an information

system.

The three main components of an ERD are:

- The entity is a person, object, place or event for which data is collected. For example, if

you consider the information system for a business, entities would include not only

customers, but the customer's address, and orders as well

- The relationship is the interaction between the entities. In the example above, the

customer places an order, so the word "places" defines the relationship between that

instance of a customer and the order or orders that they place. A relationship may be

represented by a diamond shape, or more simply, by the line connecting the entities. In

either case, verbs are used to label the relationships.

- The cardinality defines the relationship between the entities in terms of numbers. An

entity may be optional: for example, a sales rep could have no customers or could have

one or many customers; or mandatory: for example, there must be at least one product

listed in an order. There are several different types of cardinality notation; crow's foot

notation, used here, is a common one. In crow's foot notation, a single bar indicates one, a

double bar indicates one and only one (for example, a single instance of a product can

only be stored in one warehouse), a circle indicates zero, and a crow's foot indicates many. The three main cardinal relationships are: one-to-one, expressed as 1:1;

one-to-many, expressed as 1:M; and many-to-many, expressed as M:N.

Attribute: describes one aspect of an entity type; usually [and best when] single valued and

indivisible (atomic).

- Represented by oval on E-R diagram
- Ex: name, maximum enrolment
- May be multi-valued - use double oval on E-R diagram
- May be composite - attribute has further structure; also use oval for composite attribute,

with ovals for components connected to it by lines

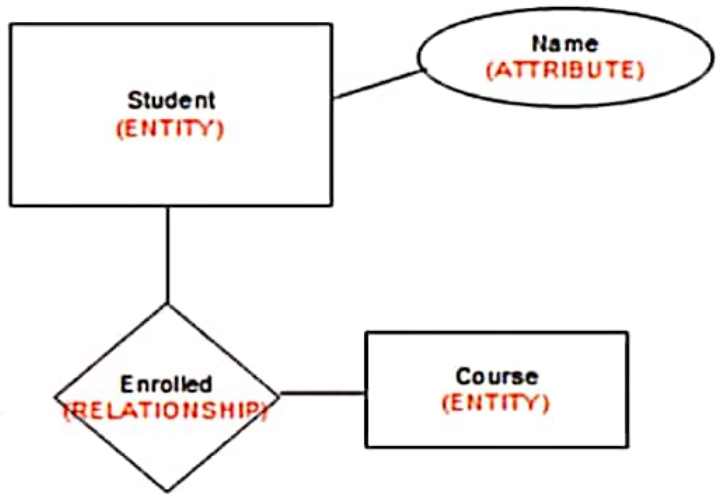
- May be derived - a virtual attribute, one that is computable from existing data in the

database, use dashed oval. This helps reduce redundancy

Symbols used in E-R Diagram

- Entity - rectangle

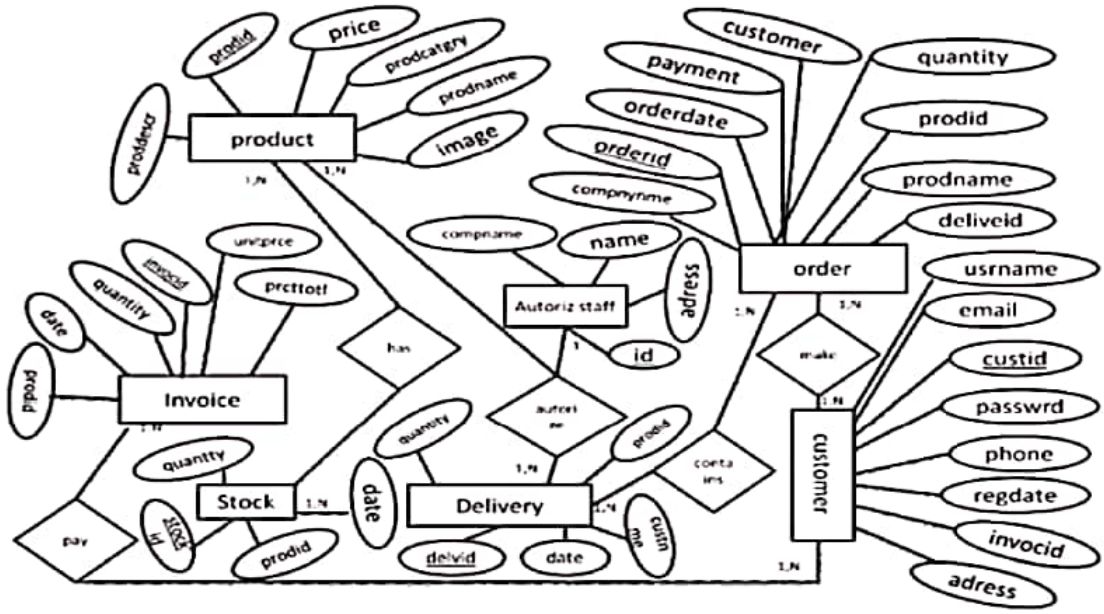
- Attribute - oval
- Relationship - diamond
- Link - line



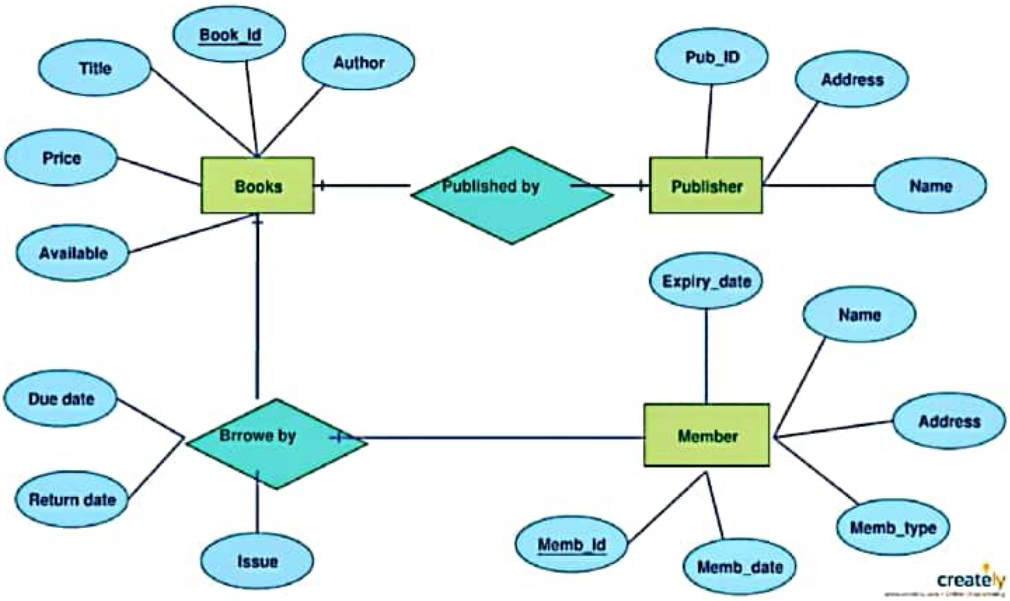
The steps involved in creating an ERD are:

- Identify the entities.
- Determine all significant interactions.
- Analyse the nature of the interactions.
- Draw the ERD

ER diagram of online ordering product:



E-R Diagram for Library Management System



Experiment No. 2:

Aim: To design a relational model.

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

In relational model data and relationships among the data is arranged in the form of rows and

columns.

The relational model for database management is an approach to logically represent and manage

the data stored in a database. In this model, the data is organized into a collection of

two-dimensional inter-related tables, also known as relations. Each relation is a collection of

columns and rows, where the column represents the attributes of an entity and the rows (or

tuples) represents the records. The use of tables to store the data provided a straightforward,

efficient, and flexible way to store and access structured information. Because of this simplicity,

this data model provides easy data sorting and data access. Hence, it is used widely around the

world for data storage and processing.

- Relation : Two-dimensional table used to store a collection of data elements.
- Tuple : Row of the relation, depicting a real-world entity.

• Attribute/Field : Column of the relation, depicting properties that define the relation.

• Attribute Domain : Set of pre-defined atomic values that an attribute can take i.e., it

describes the legal values that an attribute can take.

• Degree : It is the total number of attributes present in the relation.

• Cardinality : It specifies the number of entities involved in the relation i.e., it is the total

number of rows present in the relation.

• Relational Schema : It is the logical blueprint of the relation i.e., it describes the design

and the structure of the relation. It contains the table name, its attributes, and their types:

• Relation Key : It is an attribute or a group of attributes that can be used to uniquely

identify an entity in a table or to determine the relationship between two tables. Relation

keys can be of 6 different types:

- Candidate Key
  Candidate keys are those attributes that uniquely identify rows of a table. The Primary Key of a table is selected from one of the candidate keys. So, candidate keys have the same properties as the primary keys explained above.
- Super Key
  Super Key is the set of all the keys which help to identify rows in a table uniquely. This means that all those columns of a table than capable of identifying the other columns of that table uniquely will all be considered super keys.
- Composite Key
  Composite Key is a set of two or more attributes that help identify each tuple in a table uniquely. The attributes in the set may not be unique when considered separately. However, when taken all together, they will ensure uniqueness.
- Primary Key

A primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely. There can be only one primary Key in a table.

- Alternate Key
  As stated above, a table can have multiple choices for a primary key; however, it can choose only one. So, all the keys which did not become the primary Key are called alternate keys.
- Foreign Key
  Foreign Key is used to establish relationships between two tables. A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table. Foreign keys help to maintain data and referential integrity.

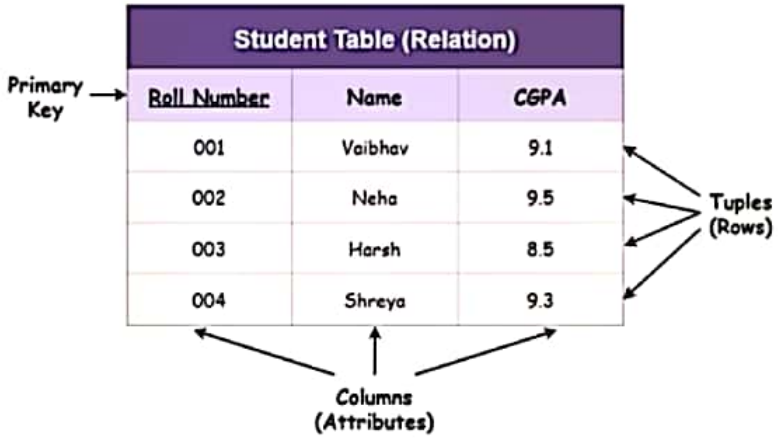
Advantages of using the relational model

1. The advantages and reasons due to which the relational model in DBMS is widely accepted as a standard are:
2. Simple and Easy To Use - Storing data in tables is much easier to understand and implement as compared to other storage techniques.
3. Manageability - Because of the independent nature of each relation in a relational database, it is easy to manipulate and manage. This improves the performance of the database.
4. Query capability - With the introduction of relational algebra, relational databases provide easy access to data via high-level query language like SQL.
5. Data integrity - With the introduction and implementation of relational constraints, the relational model can maintain data integrity in the database.

Disadvantages of using the relational model

1. The main disadvantages of relational model in DBMS occur while dealing with a huge amount of data as:
2. The performance of the relational model depends upon the number of relations present in the database.
3. Hence, as the number of tables increases, the requirement of physical memory increases.
4. The structure becomes complex and there is a decrease in the response time for the queries.
5. Because of all these factors, the cost of implementing a relational database increase.

Relational Model in DBMS



Name	Dry/Wet Food	Good Boy (Y/N)
Fido	Dry	Y
Rex	Wet	N
Bubbles	Dry	Y
Cujo	Wet	N

Tag #	Height (in)	Weight (lbs)
1573	15	21
2684	9	7
3795	27	130
4806	6	5

Tag #	Name	Breed	Color	Age
1573	Fido	Beagle	Brown/White	1.5
2684	Rex	Pekingese	White	9
3795	Bubbles	Rottweiler	Black	5
4806	Cujo	Chihuahua	Gold	4

Experiment No 3:

Aim: To Perform basic DDL (Data Definition Language) commands on SQL

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

SQL is structured Query Language which is a computer language for storing, manipulating and

retrieving data stored in relational database.

SQL Commands:

The standard SQL commands to interact with relational databases are CREATE, SELECT,

INSERT, UPDATE, DELETE, and DROP. These commands can be classified into groups based

on their nature:

- 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

RENAME - rename an object

Create Table:

It is used to create a table

Syntax: Create table tablename (column_name1 data_ type constraints, column_name2

data_ type constraints ...)

Example:

1)Create table stud (sname varchar2(20) not null, rollno number(10) not null,dob date not null);

2) Create table Emp (EmpNo number(5), EName VarChar(15), Job Char(10));

Alter Table:

Alter command is used to:

1. Add a new column.

3. Modify the existing column definition.

4. To include or drop integrity constraint.

Syntax: alter table tablename add/modify (attribute datatype(size));

Example:

1. Alter table emp add (phone_no char (20));

2. Alter table emp modify(phone_no number (10));

3. ALTER TABLE EMP ADD CONSTRAINT Pkey1 PRIMARY KEY (EmpNo);

Drop Table:

It will delete the table structure provided the table should be empty.

Example: drop table table_name;

Eg: drop table employee; Here employee is table name

Truncate Table:

If there is no further use of records stored in a table and the structure has to be retained

then the records alone can be deleted.

Syntax: TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

Rename Table:

It is used to change the name of table created earlier. Rename changes only the table name,

contents within the table are not altered.

Syntax: Rename TABLE <TABLE NAME> to TABLE<NEW_TABLE_NAME>;

DESC:

This is used to view the structure of the table.

Example: desc emp;

Name	Null?	Type
EmpNo	NOT NULL	number(5)
EName		VarChar(15)
Job	NOT NULL	Char(10)
DeptNo	NOT NULL	number(3)
PHONE_NO		number (10)

EXERCISE

1) Create a table and perfom all the operations of DDL on the table?

```
MariaDB [practicals]> CREATE TABLE studentjk(roll   INT,fNAME VARCHAR (10),lNAME VARCHAR (10),divison VARCHAR (10));
Query OK, 0 rows affected (0.025 sec)

MariaDB [practicals]> desc studentj
    -> ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(10) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.026 sec)

MariaDB [practicals]> desc studentjk;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll    | int(11)     | YES  |     | NULL    |       |
| fNAME   | varchar(10) | YES  |     | NULL    |       |
| lNAME   | varchar(10) | YES  |     | NULL    |       |
| divison | varchar(10) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.017 sec)



MariaDB [practicals]> alter table studentjk add(phno char(20));
Query OK, 0 rows affected (0.028 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [practicals]> desc studentjk;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll    | int(11)     | YES  |     | NULL    |       |
| fNAME   | varchar(10) | YES  |     | NULL    |       |
| lNAME   | varchar(10) | YES  |     | NULL    |       |
| divison | varchar(10) | YES  |     | NULL    |       |
| phno    | char(20)    | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.015 sec)
```

```
MariaDB [practicals]> alter table studentjk modify phno INT;
Query OK, 0 rows affected (0.069 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [practicals]> desc studentjk;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll    | int(11)     | YES  |     | NULL    |       |
| fNAME   | varchar(10) | YES  |     | NULL    |       |
| lNAME   | varchar(10) | YES  |     | NULL    |       |
| divison | varchar(10) | YES  |     | NULL    |       |
| phno    | int(11)     | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.017 sec)

MariaDB [practicals]> truncate table studentjk;
Query OK, 0 rows affected (0.030 sec)



MariaDB [practicals]> alter table studentjk rename to studentk;
Query OK, 0 rows affected (0.022 sec)

MariaDB [practicals]> desc studentk;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll    | int(11)     | YES  |     | NULL    |       |
| fNAME   | varchar(10) | YES  |     | NULL    |       |
| lNAME   | varchar(10) | YES  |     | NULL    |       |
| divison | varchar(10) | YES  |     | NULL    |       |
| phno    | int(11)     | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

Conclusion:

Data definition language commands such as CREATE, ALTER,TRUNCATE,RENAME,DROP and DELETE are studied and performed operations on table in SQL.

Experiment No 4:

Aim: To Perform basic DML (Data Manipulation Language) commands on SQL

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

DML is abbreviation of Data Manipulation Language. It is used to retrieve, store, modify, delete, insert and update data in database.SQL is structured Query Language which is a computer language for storing, manipulating and retrieving data stored in relational database.

SQL Commands:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table.
- UPDATE - updates existing data within a table.
- DELETE - deletes all records from a table, the space for the records remain

SELECT Statement: The SELECT statement is used to select data from a database.

The result is stored in a result table, called the result-set.

Syntax:

SELECT column_name(s) FROM table_name

OR

SELECT * FROM table_name

INSERT INTO Statement: The INSERT INTO statement is used to insert a new row in a table.

This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined

Syntax:

INSERT INTO table_name VALUES (value1,value2,value3,...)

OR

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

DELETE Statement: The DELETE statement is used to delete rows in a table.

After inserting row in a table we can also delete them if required. The delete command consists

of a from clause followed by an optional where clause.

Syntax:

DELETE FROM table_name WHERE some_column=some_value

UPDATE Statement: The UPDATE statement is used to update existing records in a table.. A single column may be updated or more than one column could be updated.

Syntax:

UPDATE table_name

SETcolumn1=value,column2=value2,...

WHERE some_column=some_value

EXERCISE

2) Create a table and perfom all the operations of DML on the table?

```
MariaDB [(none)]> use database practicals;
ERROR 1049 (42000): Unknown database 'database'
MariaDB [(none)]> use practicals;
Database changed
MariaDB [practicals]> desc studentk;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll    | int(11)     | YES  |     | NULL    |       |
| fNAME   | varchar(10) | YES  |     | NULL    |       |
| lNAME   | varchar(10) | YES  |     | NULL    |       |
| divison | varchar(10) | YES  |     | NULL    |       |
| phno    | int(11)     | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.029 sec)

MariaDB [practicals]> insert into studentk values(1,'jayesh','kirtane','ds',983434);
Query OK, 1 row affected (0.028 sec)

MariaDB [practicals]> insert into studentk values(2,'ajit','kale','ds',983934);
Query OK, 1 row affected (0.010 sec)

MariaDB [practicals]> insert into studentk values(3,'sushant','babar','ds',984034);
Query OK, 1 row affected (0.012 sec)

MariaDB [practicals]> insert into studentk values(4,'jibran','khan','ds',989034);
Query OK, 1 row affected (0.011 sec)

MariaDB [practicals]> insert into studentk values(4,'yash','gupta','ds',911034);
Query OK, 1 row affected (0.011 sec)

MariaDB [practicals]> insert into studentk (lname)
    -> values('mayekar');
Query OK, 1 row affected (0.010 sec)

MariaDB [practicals]> select * from studentk;
+------+---------+---------+---------+--------+
| roll | fNAME   | lNAME   | divison | phno   |
+------+---------+---------+---------+--------+
|    1 | jayesh  | kirtane | ds      | 983434 |
|    2 | ajit    | kale    | ds      | 983934 |
|    3 | sushant | babar   | ds      | 984034 |
|    4 | jibran  | khan    | ds      | 989034 |
```

```
|    3 | sushant | babar   | ds      | 984034 |
|    4 | jibran  | khan    | ds      | 989034 |
|    4 | yash    | gupta   | ds      | 911034 |
| NULL | NULL    | mayekar | NULL    |   NULL |
+------+---------+---------+---------+--------+
6 rows in set (0.050 sec)

MariaDB [practicals]> update studentk
    -> set fname = 'tejas'
    -> where lname='mayekar';
Query OK, 1 row affected (0.015 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [practicals]> select * from studentk;
+------+---------+---------+---------+--------+
| roll | fNAME   | lNAME   | divison | phno   |
+------+---------+---------+---------+--------+
|    1 | jayesh  | kirtane | ds      | 983434 |
|    2 | ajit    | kale    | ds      | 983934 |
|    3 | sushant | babar   | ds      | 984034 |
|    4 | jibran  | khan    | ds      | 989034 |
|    4 | yash    | gupta   | ds      | 911034 |
| NULL | tejas   | mayekar | NULL    |   NULL |
+------+---------+---------+---------+--------+
6 rows in set (0.001 sec)

MariaDB [practicals]> delete from studentk
    -> where lname='mayekar';
Query OK, 1 row affected (0.011 sec)

MariaDB [practicals]> select * from studentk;
+------+---------+---------+---------+--------+
| roll | fNAME   | lNAME   | divison | phno   |
+------+---------+---------+---------+--------+
|    1 | jayesh  | kirtane | ds      | 983434 |
|    2 | ajit    | kale    | ds      | 983934 |
|    3 | sushant | babar   | ds      | 984034 |
|    4 | jibran  | khan    | ds      | 989034 |
|    4 | yash    | gupta   | ds      | 911034 |
+------+---------+---------+---------+--------+
5 rows in set (0.001 sec)

MariaDB [practicals]>
```

Conclusion:

Data definition language commands such as SELECT, INSERT, UPDATE and DELETE are studied and performed operations on table in SQL.

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

CSE-AI and DS Department

Experiment No 5:

Aim: To Perform basic DCL (Data Control Language) commands on SQL

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

DCL is abbreviation of Data Control Language. It is used to create roles, permissions, and

referential integrity as well it is used to control access to database by securing it.

- **GRANT** – Gives user's access privileges to database
- **REVOKE** – Withdraws user's access privileges to database given with the GRANT command

GRANT PRIVILEGES ON TABLE

We can grant users various privileges to tables. These permissions can be any combination of

SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, or ALL.

a) Grant privileges using the GRANT statement

The grant statement provides various types of access to database objects such as tables,

views and sequences and so on.

Syntax:

GRANT<objectprivileges>ON<objectname>TO<username>

[WITH GRANT OPTION];

Example:

GRANT SELECT,INSERT,UPDATE,DELETE ON Employees TO vijay;

b) Revoke permissions using the REVOKE statement:

The REVOKE statement is used to deny the Grant given on an object.

Syntax:

REVOKE<objectprivilege>ONFROM<username>;

Example:

REVOKE INSERT ON Employees FROM Vijay;

Steps to be followed for DCL command execution

Step 1: create a table

Syntax: create table table_name;

Step 2: insert values in table

Syntax: insert into table_namevalues(attributes datatype(size));

Step 3: create user

Syntax: create user user_name identified by *****;

Step 4: grant session to user

Syntax: grant create session to user_name;

Step 5: grant operations to user;

Syntax: grant insert,delete on table_name to user_name;

Step6: check login for user with password

Syntax:- connect username

Password:****

Step 7: Insert into system. table_namevalues()

Step 8: connect system with password

Step9: select * from table_name

Step 10: revoke operations from user

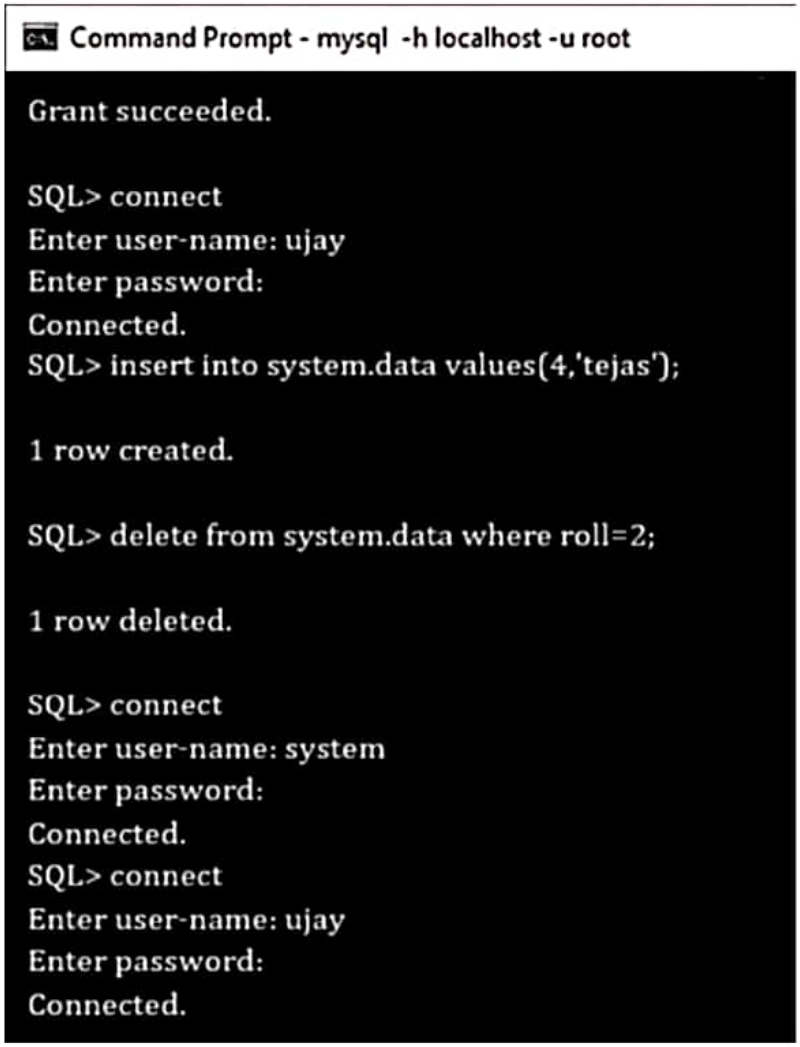
Syntax: revoke insert on table_name fromuser_name;

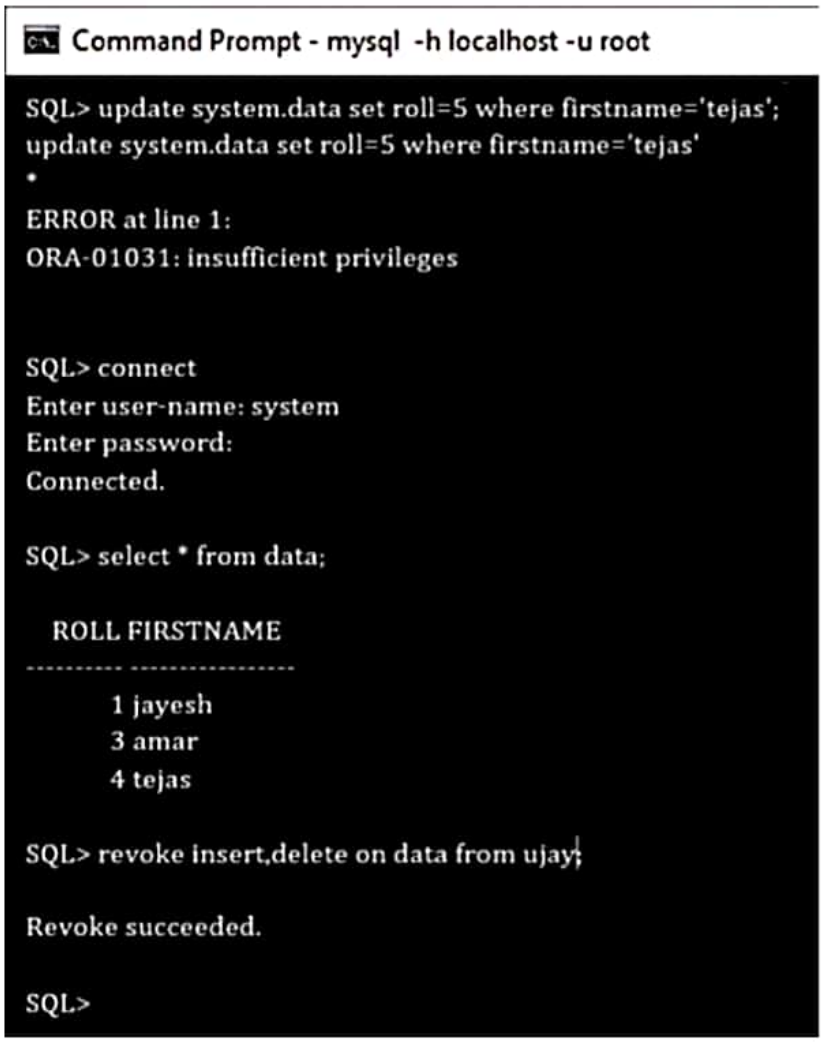
EXERCISE

3) Create a table and perform GRANT, REVOKE operations of DCL on the table?

Command Prompt - mysql -h localhost -u root

```
SQL> grant create session to ujay;

Grant succeeded.

SQL> desc data;
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------

ROLL                                               NUMBER(2)
FIRSTNAME                                          VARCHAR2(9)

SQL> select * from data;

ROLL FIRSTNAME
---------- ---------
1 jayesh
2 yash
3 amar

SQL> grant insert,delete on data to ujay;
```





Conclusion:

Data control language commands such as **GRANT, REVOKE** are studied and performed

operations on table in SQL.

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

CSE-AI and DS Department

Experiment No 6:

Aim: To Perform basic **TCL (Transaction control Language) commands** on SQL.

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

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

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.

Synatx: commit;

Rollback

If we have used the UPDATE command to make some changes into the database,

and realise that those changes were not required, then we can use the ROLLBACK

command to rollback those changes, if they were not commited using the COMMIT

command.

SYNTAX: rollback to savepoint_name;

Savepoint

command is used to temporarily save a transaction so that you can rollback to that

point whenever required.

SYNTAX: savepointsavepoint_name;

Example:

Consider a table shown by table_name 'class'

Roll_no	Name
1	Ashish
2	Shweta
3	Vijay

INSERT INTO class VALUES(4, 'Rahul');

COMMIT;

UPDATE class SET name = 'Abhijit' WHERE id = '4';

SAVEPOINT A;

INSERT INTO class VALUES(5, 'ajay');

SAVEPOINT B;

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

SAVEPOINT C;

SELECT * FROM class;

Roll_no	Name
1	Ashish
2	Shweta
3	Vijay
4	Abhijit
5	Ajay
6	Nisha

Now use the ROLLBACK command to roll back the state of data to the savepoint B.

ROLLBACKTO B;

SELECT*FROM class;

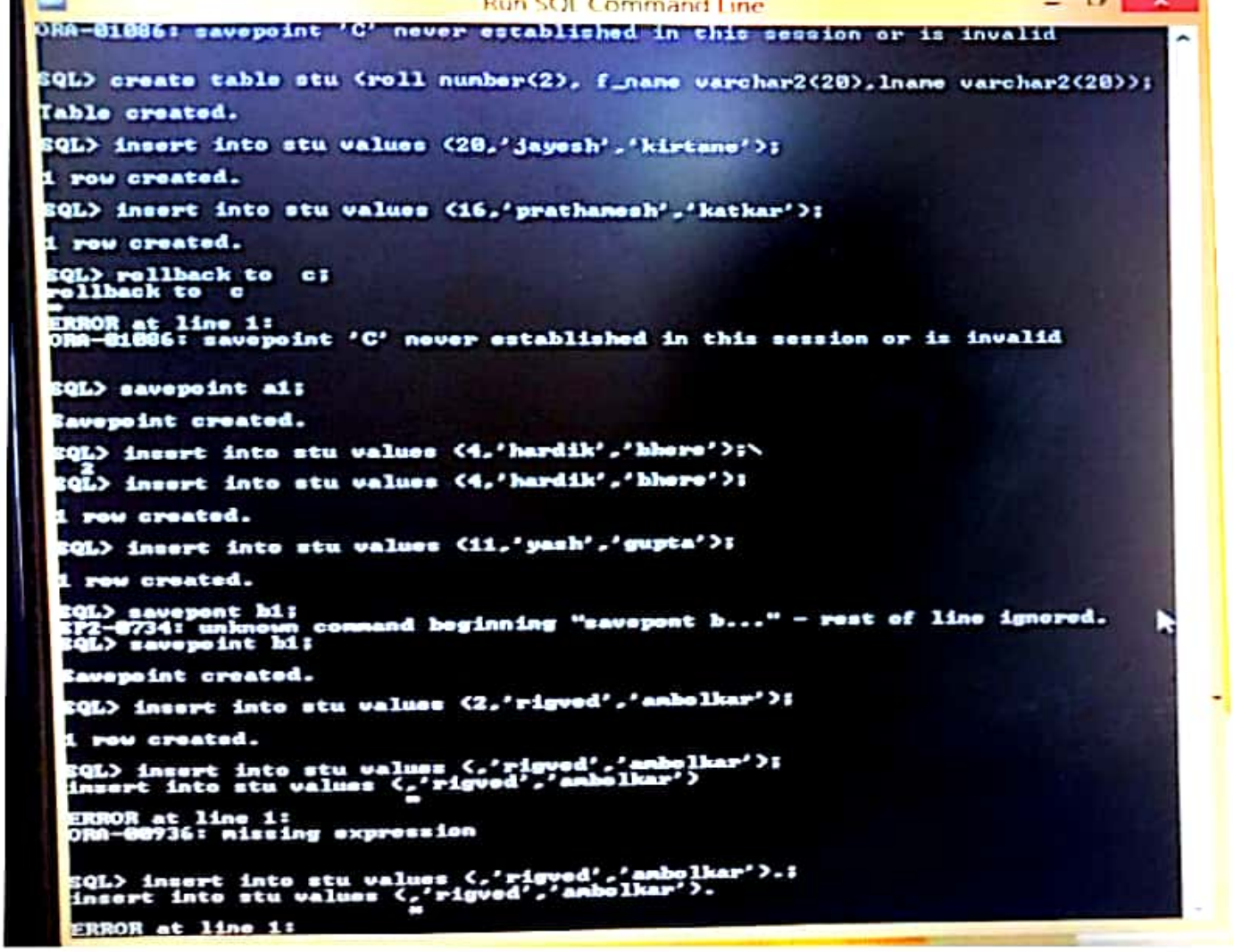
Roll_no	Name
1	Ashish
2	Shweta
3	Vijay
4	Abhijit
5	Ajay

Now use the ROLLBACK command to roll back the state of data to the savepoint A.

ROLLBACKTO A;

SELECT*FROM class;

Roll_no	Name
1	Ashish
2	Shweta
3	Vijay
4	Abhijit



Conclusion:

Transaction control language commands such as **COMMIT, ROLLBACK and SAVEPOINT** are studied and performed operations on table in SQL.

server.It manipulates the data in database for display purpose like aggregate function. In DRL/DSL, for accessing the data it uses the DI command that is SELECT.The SELECT command allows database users to retrieve the specific information they desire from an operatio database.

FROM	It is used for selecting a table name in a database
WHERE	It specifies which rows to retrieve
GROUP BY	It is used to arrange the data into groups
HAVING	It selects among the groups defined by the GROUP BY clause
ORDER BY	It specifies an order in which to return the rows.

multiple rows are grouped together as input on certain criteria to form a single

value of more significant meaning.

A group function returns a result based on group of rows.

1. avg–
   syntax: select avg(column_name) as avg from table_name;
2. max–
   syntax: select max(column_name) as max from table_name;
3. min–
   syntax: select min(column_name) as min from table_name;
4. sum–
   syntax: select sum(column_name) as sum from table_name;
5. Count -- It counts all, inclusive of duplicates and nulls.
   syntax: select count (column_name) as count from table_name;

SPECIAL OPERATORS:

1.In / not in – used to select a equi from a specific set of values

2.Any - used to compare with a specific set of values

3.Between / not between ⇨ used to find between the ranges

4.Like / not like ⇨ used to do the pattern matching

- Display all the details of the records whose employee name starts with 'A'
  SQL> select * from emp where nameclike 'A%';
- Display all the details of the records whose employee name ends with 'A'
  SQL> select * from emp where nameclike '%A';
- Display all the details of the records whose employee with alphabet 'A'
  SQL> select * from emp where nameclike '%A%';
- Display all the details of the records whose employee with position ofan alphabet
  SQL> select * from emp where nameclike '_A%';
- Display the rows whose salary ranges from 15000 to 30000.
  SQL>select * from employee where sal between 15000 and 30000;

```
MariaDB [practicals]> create table exp7(roll int(2),name varchar(10),marks int(3));
Query OK, 0 rows affected (0.035 sec)

MariaDB [practicals]> insert into exp7 values(1,'yash',98);
Query OK, 1 row affected (0.006 sec)

MariaDB [practicals]> insert into exp7 values(2,'jayesh',99);
Query OK, 1 row affected (0.010 sec)

MariaDB [practicals]> insert into exp7 values(3,'tejas',89);
Query OK, 1 row affected (0.011 sec)

MariaDB [practicals]> insert into exp7 values(4,'omar',91);
Query OK, 1 row affected (0.010 sec)

MariaDB [practicals]> insert into exp7 values(5,'rigved',50);
Query OK, 1 row affected (0.009 sec)

MariaDB [practicals]> insert into exp7 values(8,'anjali',29);
Query OK, 1 row affected (0.002 sec)

MariaDB [practicals]> select min(marks) as m from exp7;
+------+
| m    |
+------+
|   29 |
+------+
1 row in set (0.024 sec)

MariaDB [practicals]> select max(marks) as m from exp7;
+------+
| m    |
+------+
|   99 |
+------+
1 row in set (0.000 sec)

MariaDB [practicals]> select sum(marks) as s from exp7;
+------+
| s    |
+------+
|  456 |
+------+
1 row in set (0.009 sec)
```

```
MariaDB [practicals]> select sum(marks) as s from exp7;
+------+
| s    |
+------+
|  456 |
+------+
1 row in set (0.009 sec)

MariaDB [practicals]> select avg(marks) as a from exp7;
+---------+
| a       |
+---------+
| 76.0000 |
+---------+
1 row in set (0.001 sec)

MariaDB [practicals]> select count(marks) as c from exp7;
+---+
| c |
+---+
| 6 |
+---+
1 row in set (0.001 sec)

MariaDB [practicals]> select * from exp7 where name like 't%';
+------+-------+-------+
| roll | name  | marks |
+------+-------+-------+
|    3 | tejas |    89 |
+------+-------+-------+
1 row in set (0.001 sec)

MariaDB [practicals]> select * from exp7 where name like '%y';
Empty set (0.001 sec)

MariaDB [practicals]> select * from exp7 where name like '%h';
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    1 | yash   |    98 |
|    2 | jayesh |    99 |
+------+--------+-------+
2 rows in set (0.000 sec)
```

```
+------+--------+-------+
2 rows in set (0.000 sec)

MariaDB [practicals]> select * from exp7 where name like '%a%';
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    1 | yash   |    98 |
|    2 | jayesh |    99 |
|    3 | tejas  |    89 |
|    4 | omar   |    91 |
|    8 | anjali |    29 |
+------+--------+-------+
5 rows in set (0.001 sec)

MariaDB [practicals]> select * from exp7 where name like '_a%';
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    1 | yash   |    98 |
|    2 | jayesh |    99 |
+------+--------+-------+
2 rows in set (0.000 sec)

MariaDB [practicals]> select * from exp7 ORDER BY marks ASC;
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    8 | anjali |    29 |
|    5 | rigved |    50 |
|    3 | tejas  |    89 |
|    4 | omar   |    91 |
|    1 | yash   |    98 |
|    2 | jayesh |    99 |
+------+--------+-------+
6 rows in set (0.001 sec)

MariaDB [practicals]> select * from exp7 ORDER BY marks DESC;
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    2 | jayesh |    99 |
|    1 | yash   |    98 |
|    4 | omar   |    91 |
```

```
MariaDB [practicals]> select * from exp7 ORDER BY marks DESC;
+------+--------+-------+
| roll | name   | marks |
+------+--------+-------+
|    2 | jayesh |    99 |
|    1 | yash   |    98 |
|    4 | omar   |    91 |
|    3 | tejas  |    89 |
|    5 | rigved |    50 |
|    8 | anjali |    29 |
+------+--------+-------+
6 rows in set (0.001 sec)

MariaDB [practicals]>
```

Conclusion:

Data Retrieval Language/Data Selection Language operations and aggregate

functions such as COUNT, MIN, MAX, AVG are studied and performed operations on

table in SQL.

Saraswati Education Society's

SARASWATI COLLEGE OF ENGINEERING KHARGHAR

CSE-AI and DS Department

Experiment No 8

Aim: To Perform SET operations in SQL

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

Set operators are used to join the results of two (or more) SELECT statements.The SET operators

available in SQL are UNION,UNION ALL,INTERSECT,and MINUS.

UNION

When multiple SELECT queries are joined using UNION operator, SQL displays

the combined result from all the compounded SELECT queries,after removing all

duplicates and in sorted order (ascending by default), without ignoring the NULL

values.in case of UNION the number of columns and datatypes must be same on

both the sides.It Returns all distinct rows selected by both the queries

SYNTAX : (select * from table1_name) union (select * from table1_name);

Union all

Returns all rows selected by either query including the duplicates.UNION and

UNION ALL are similar in their functioning with a slight difference. But UNION

ALL gives the result set without removing duplication and sorting the data.

SYNTAX : (select * from table1_name) union all (select * from table1_name);

INTERSECT

Using INTERSECT operator, SQL displays the common rows from both the

SELECT statements, with no duplicates and data arranged in sorted order

(ascending by default). In case of INTERSECT also the number of columns and

datatypes must be same on both the sides. It Returns rows selected that are common to

both queries

SYNTAX : (select * from table1_name) intersect (select * from table1_name);

MINUS

Minus operator displays the rows which are present in the first query but absent in

the second query, with no duplicates and data arranged in ascending order by

default.Returns all distinct rows selected by the first query and are not by the

second

SYNTAX : (select * from table1_name) minus (select * from table1_name);

Steps to execute set operations commands

Step 1 : create two tables table1 & table2

Step 2 : insert entries in both the tables

Step 3 : perform operations using above syntax

Step4 : observe the result

EXERCISE

```
1 row created.

SQL> select * from student2;

      ROLL NAME           POINTER
---------- ---------- ----------
         3 anjali            3.5
         7 deepti            8.5
         8 tanmayee          7.2
         9 janhavi           6.2
        10 aisha             9.7

SQL> (select * from student1) union (select * from student2);

      ROLL NAME           POINTER
---------- ---------- ----------
         1 tejas             5.6
         2 yash              6.5
         3 anjali            3.5
         4 prathamesh        5.3
         5 jayesh            9.8
         7 deepti            8.5
         8 tanmayee          7.2
         9 janhavi           6.2
        10 aisha             9.7

9 rows selected.

SQL> (select * from student1) intersect (select * from student2);

      ROLL NAME           POINTER
---------- ---------- ----------
         3 anjali            3.5

SQL> (select * from student1) union all (select * from student2);

      ROLL NAME           POINTER
---------- ---------- ----------
         1 tejas             5.6
         2 yash              6.5
         3 anjali            3.5
         4 prathamesh        5.3
         5 jayesh            9.8
         3 anjali            3.5
         7 deepti            8.5
         8 tanmayee          7.2
         9 janhavi           6.2
        10 aisha             9.7

10 rows selected.

SQL> (select * from student1) minus (select * from student2);

      ROLL NAME           POINTER
---------- ---------- ----------
         1 tejas             5.6
```

```
SQL*Plus: Release 11.2.0.2.0 Production on Mon Apr 4 11:05:37 2022

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> create table student1 (roll number(2), name varchar2(10),pointer float(6));

Table created.

SQL> desc student1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ROLL                                               NUMBER(2)
 NAME                                               VARCHAR2(10)
 POINTER                                            FLOAT(6)

SQL> insert into student1(1,'tejas',5.6);
insert into student1(1,'tejas',5.6)
                     *
ERROR at line 1:
ORA-00928: missing SELECT keyword


SQL> insert into student1 values(1,'tejas',5.6);

1 row created.

SQL> insert into student1 values(2,'yash',6.5);

1 row created.

SQL> insert into student1 values(3,'anjali',3.5);

1 row created.

SQL> insert into student1 values(4,'prathamesh',5.3);

1 row created.

SQL> insert into student1 values(5,'jayesh',9.8);

1 row created.

SQL> select * from student1;

      ROLL NAME          POINTER
---------- ---------- ----------
         1 tejas             5.6
         2 yash              6.5
         3 anjali            3.5
         4 prathamesh        5.3
```

```
         1 tejas                5.6
         2 yash                 6.5
         3 anjali               3.5
         4 prathamesh           5.3
         5 jayesh               9.8
         3 anjali               3.5
         7 deepti               8.5
         8 tanmayee             7.2
         9 janhavi              6.2
        10 aisha                9.7

10 rows selected.

SQL> (select * from student1) minus (select * from student2);

      ROLL NAME           POINTER
---------- ---------- ----------
         1 tejas                5.6
         2 yash                 6.5
         4 prathamesh           5.3
         5 jayesh               9.8

SQL> (select * from student2) minus (select * from student1);

      ROLL NAME           POINTER
---------- ---------- ----------
         7 deepti               8.5
         8 tanmayee             7.2
         9 janhavi              6.2
        10 aisha                9.7

SQL>
```

Resources Required: H/W: P4 machine

S/W: Oracle 10g

Theory:

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.

Example: Let's use the above EMPLOYEE table and SALARY table:

Input:

1. ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000

An outer join is basically of three types:

a. Left outer join
b. Right outer join
c. Full outer join

a. Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1. EMPLOYEE ⟕ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY

Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

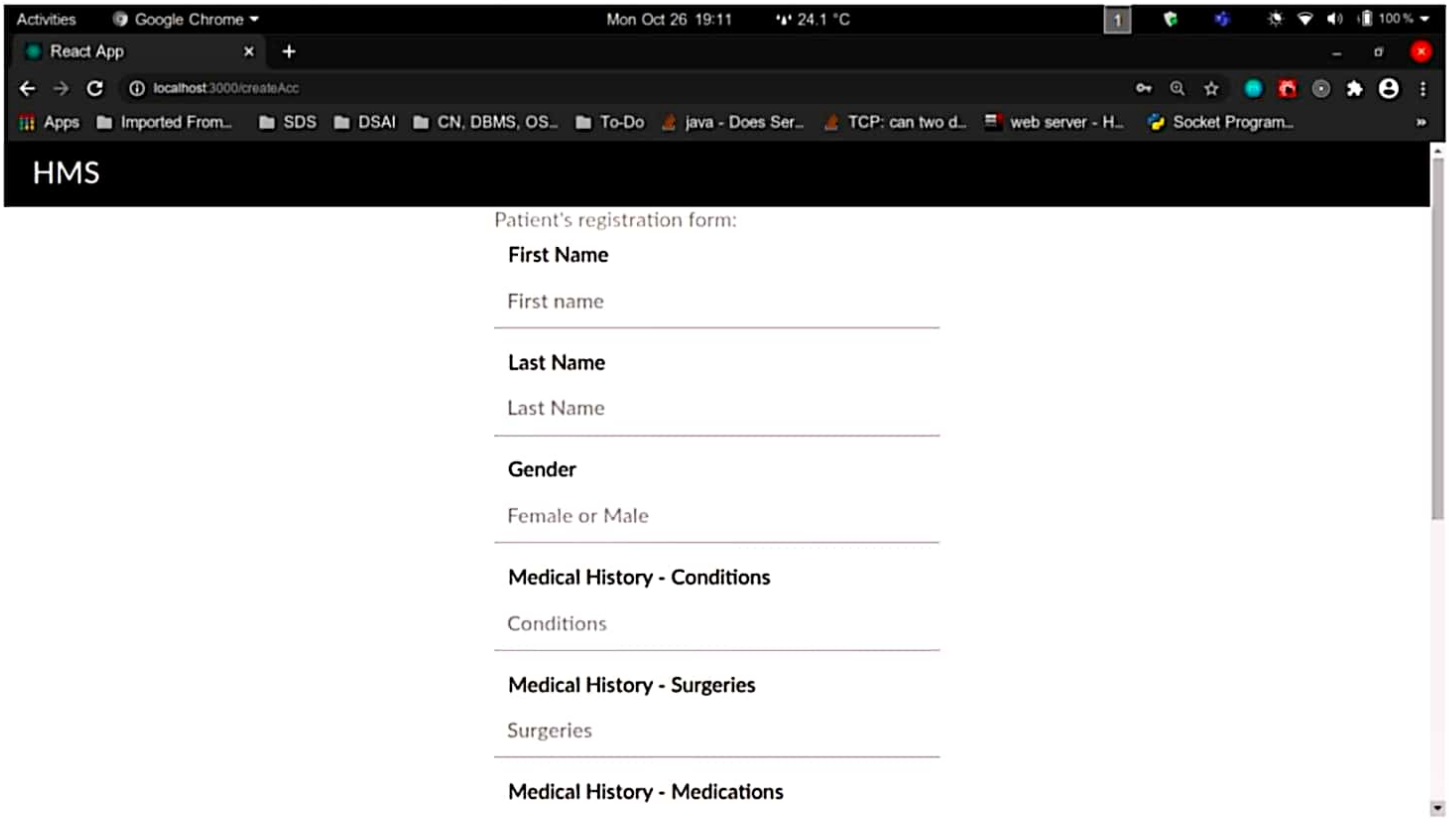
1. EMPLOYEE ⟗ FACT_WORKERS

Output:

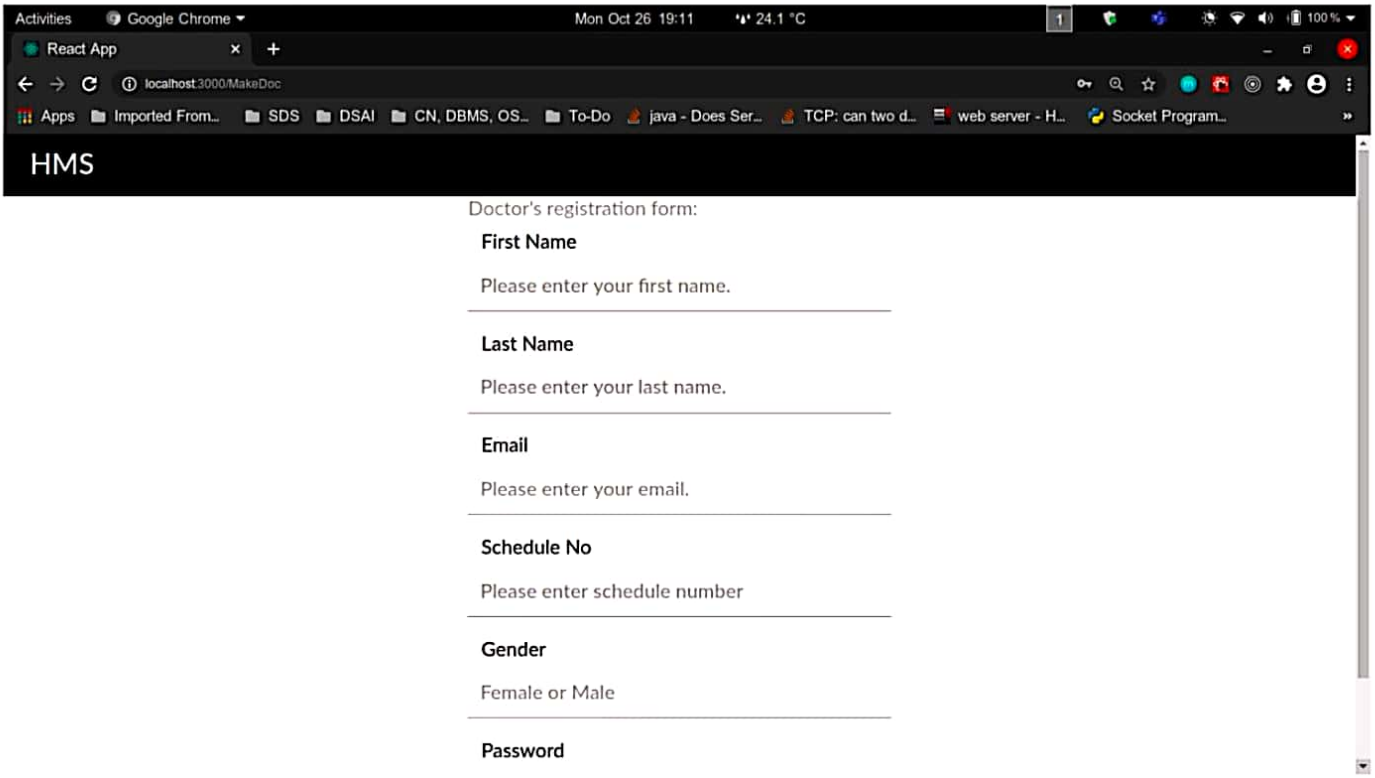
```
SQL> insert into customer values(5,'omar',43,65);

1 row created.

SQL> insert into salesman1 values(1,'prathamesh','pen',6);

1 row created.

SQL> insert into salesman1 values(2,'omkar','eraser',22);

1 row created.

SQL> insert into salesman1 values(3,'deepti','mobile',67);

1 row created.

SQL> insert into salesman1 values(4,'sushant','book',34);

1 row created.

SQL> insert into salesman1 values(5,'ajit','mouse',43);

1 row created.

SQL> select * from customer;

        ID NAME                      AGE     AMOUNT
---------- ---------- ---------- ----------
         1 jayesh                    19         80
         2 yash                      40         10
         3 tejas                     90         50
         4 anjali                    34         56
         5 omar                      43         65

SQL> select * from salesman1;

        ID FNAME      ITEM       COMMISION
---------- ---------- ---------- ----------
         1 prathamesh pen                 6
         2 omkar      eraser             22
         3 deepti     mobile             67
         4 sushant    book               34
         5 ajit       mouse              43

SQL> select customer.name,customer.amount,salesman1.item
  2  salesman1.fname from customer innerjoin salesman1 on
  3  customer.id = salesman1.id;
salesman1.fname from customer innerjoin salesman1 on
         *
ERROR at line 2:
ORA-00923: FROM keyword not found where expected


SQL> select customer.name,customer.amount,salesman1.item,
  2  salesman1.fname from customer innerjoin salesman1 on
  3  customer.id = salesman1.id;
salesman1.fname from customer innerjoin salesman1 on
```

```
SQL> select customer.NAME, salesman1.ITEM from customer inner join salesman1 on
customer.ID=salesman1.ID;

NAME       ITEM
---------- ----------
jayesh     pen
yash       eraser
tejas      mobile
anjali     book
omar       mouse

SQL> select customer.NAME, salesman1.ITEM from customer left join salesman1 on c
ustomer.ID=salesman.ID;
select customer.NAME, salesman1.ITEM from customer left join salesman1 on custom
er.ID=salesman.ID

       *
ERROR at line 1:
ORA-00904: "SALESMAN"."ID": invalid identifier


SQL> select customer.NAME, salesman1.ITEM from customer left join salesman1 on c
ustomer.ID=salesman1.ID;

NAME       ITEM
---------- ----------
jayesh     pen
yash       eraser
tejas      mobile
anjali     book
omar       mouse

SQL> select customer.NAME, salesman1.ITEM from customer right join salesman1 on
customer.ID=salesman1.ID;

NAME       ITEM
---------- ----------
jayesh     pen
yash       eraser
tejas      mobile
anjali     book
omar       mouse

SQL> select customer.NAME, salesman1.ITEM from customer full join salesman1 on c
ustomer.ID=salesman1.ID;

NAME       ITEM
---------- ----------
jayesh     pen
yash       eraser
tejas      mobile
anjali     book
omar       mouse

SQL>
```

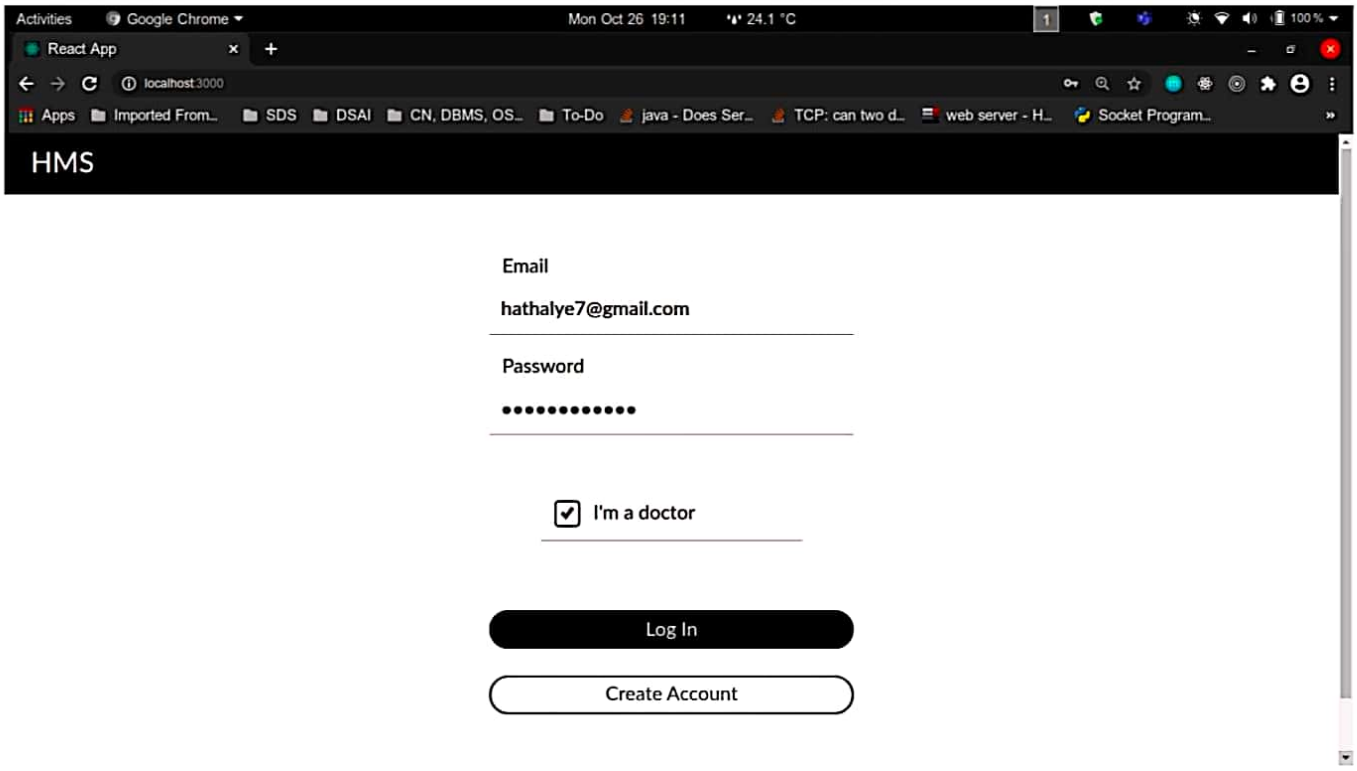
1. Patient registering on the system:



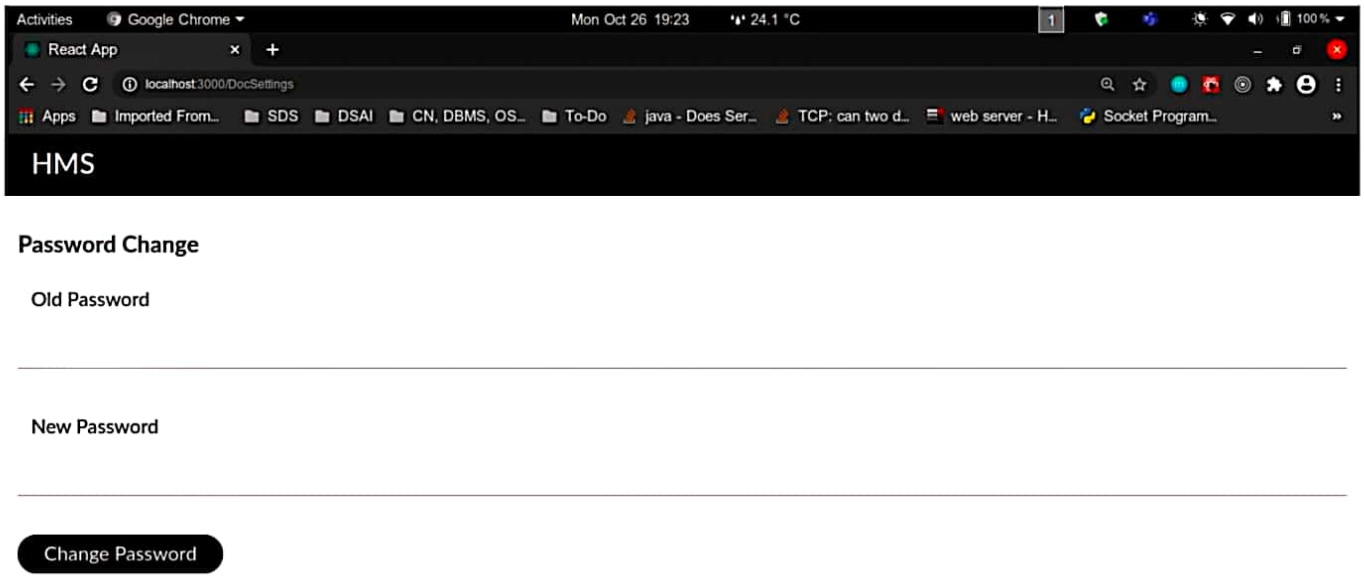
2. Doctor registering on the system:



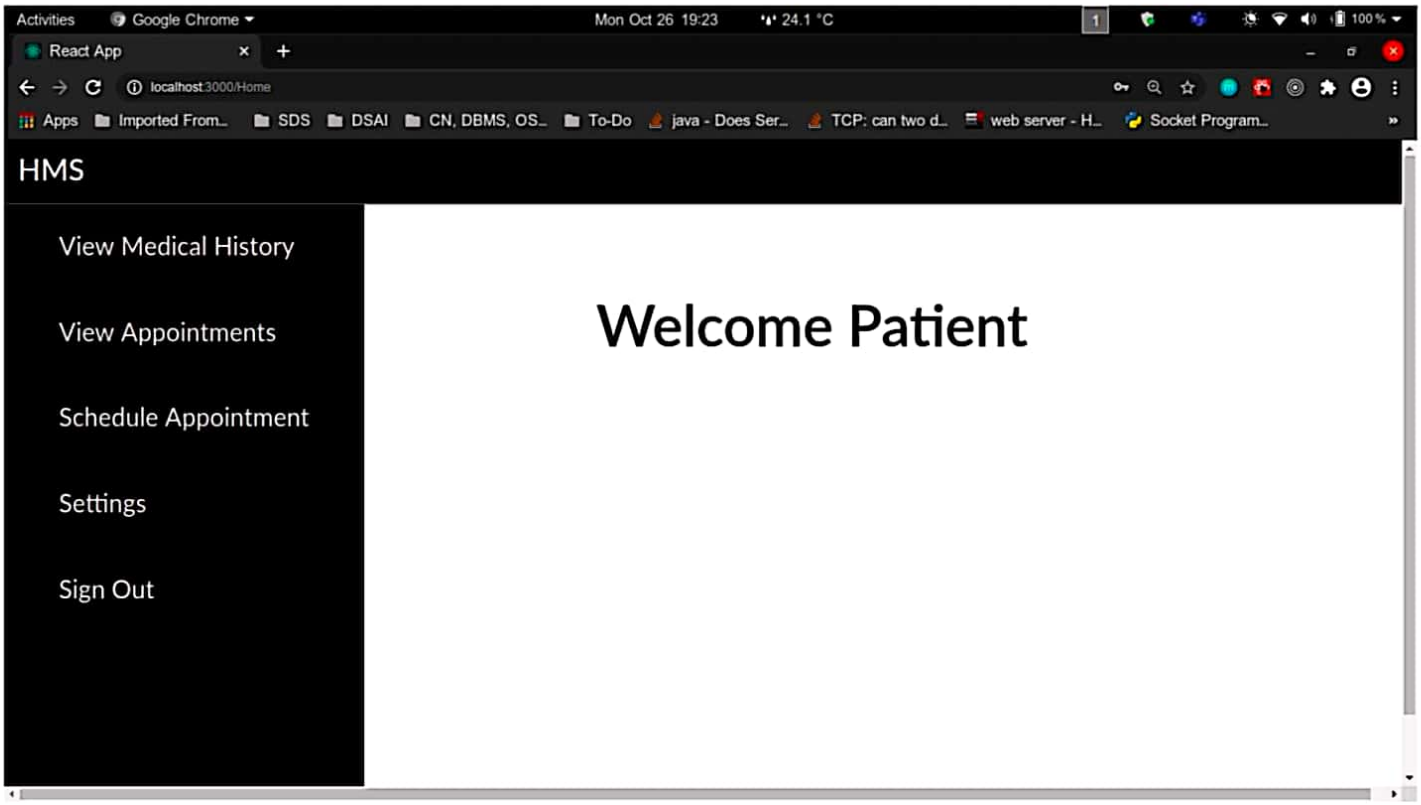
3. Log In Screen:



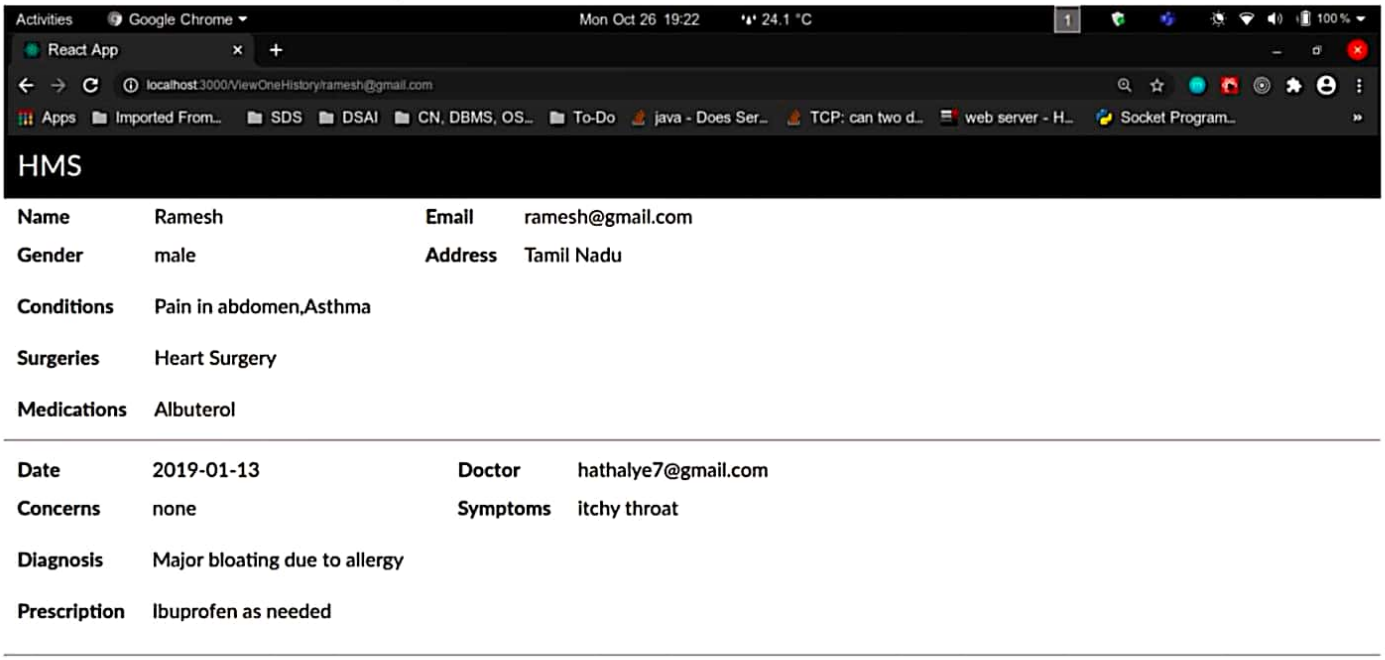
4. Password Reset Screen:



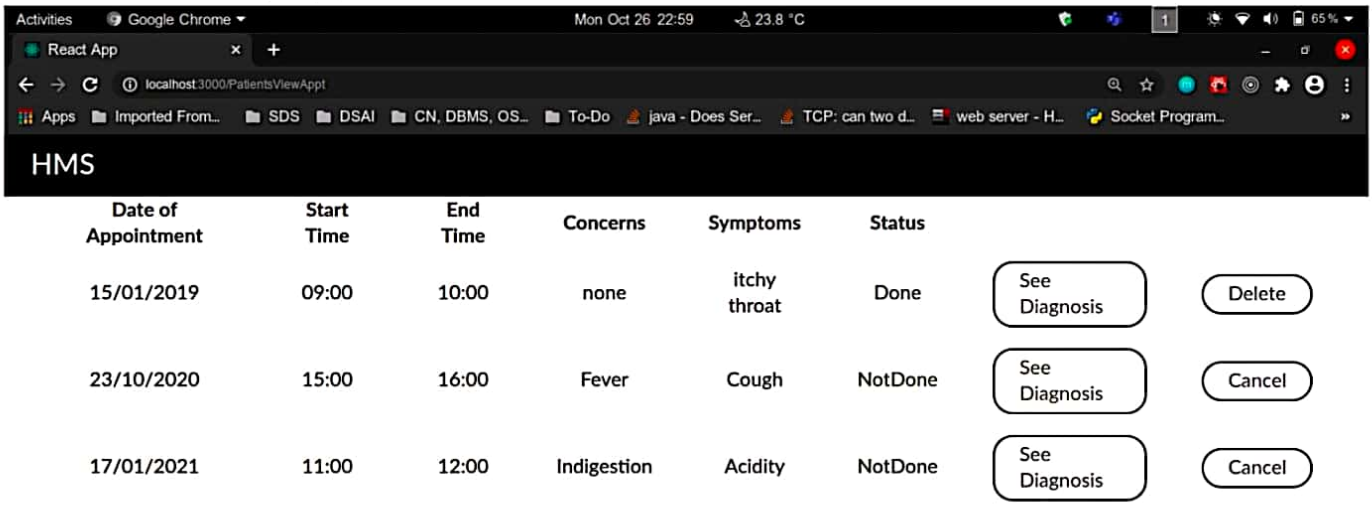
5. Patient Home Screen:



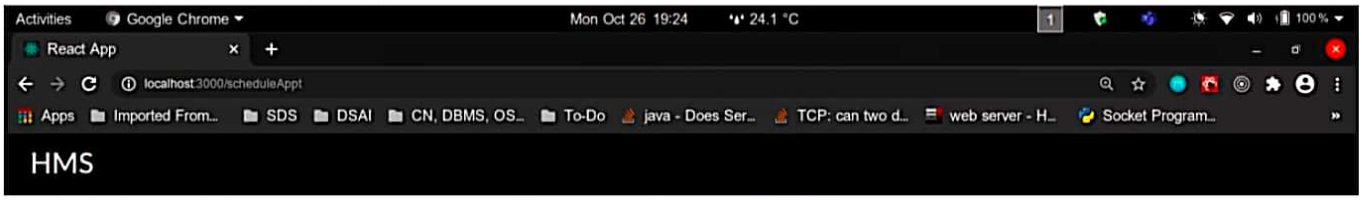
6. Patient Viewing History:

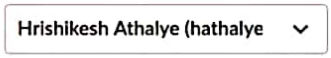


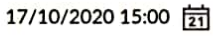
7. Patient Viewing Appointments:



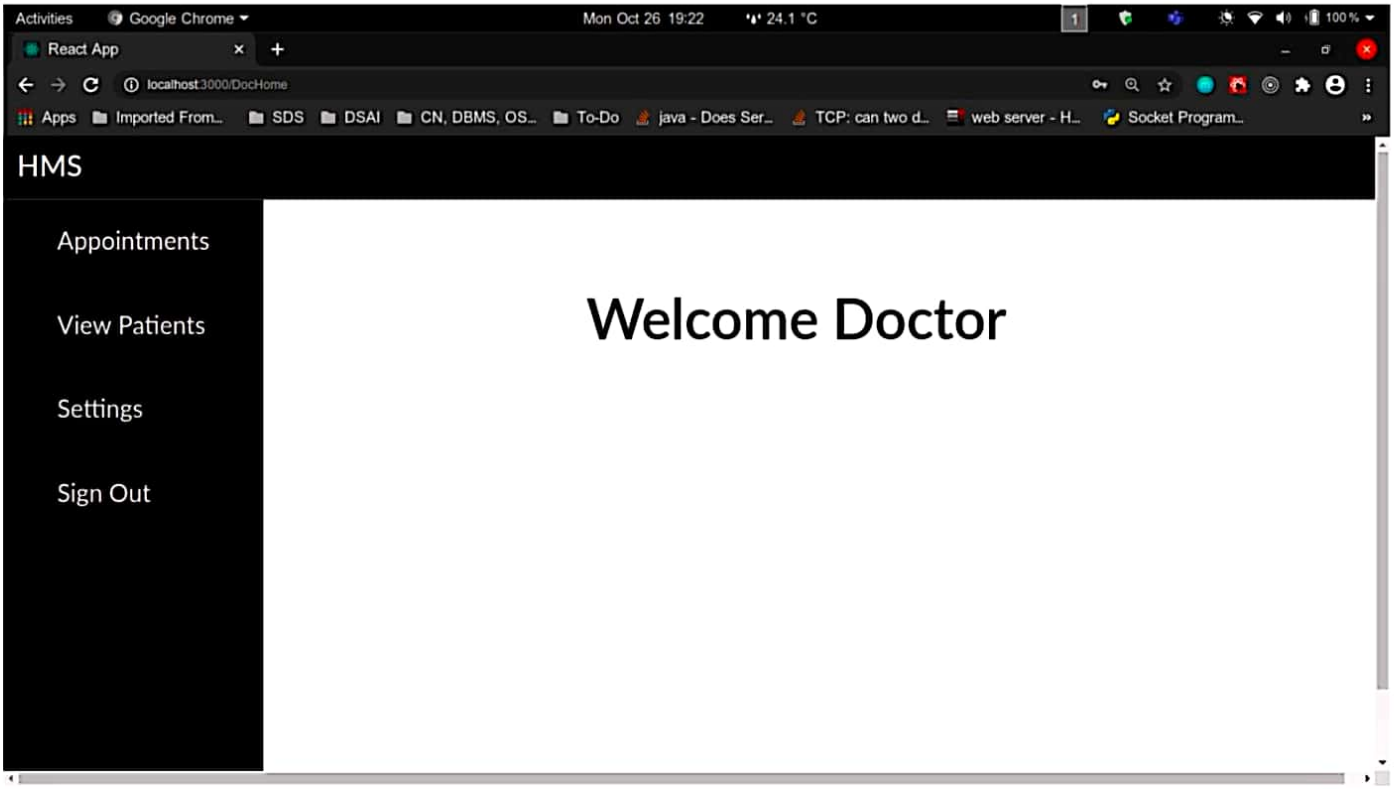
8. Patient Scheduling Appointment:



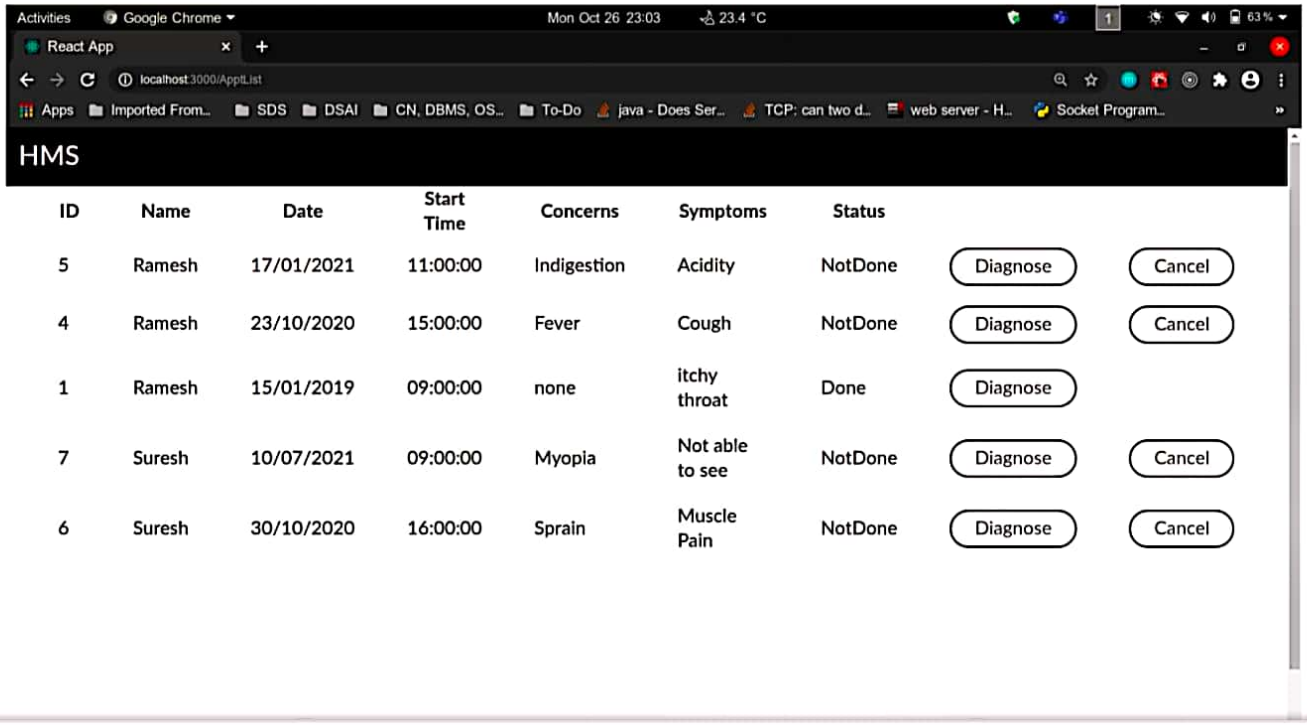




9. Doctor Home Screen:

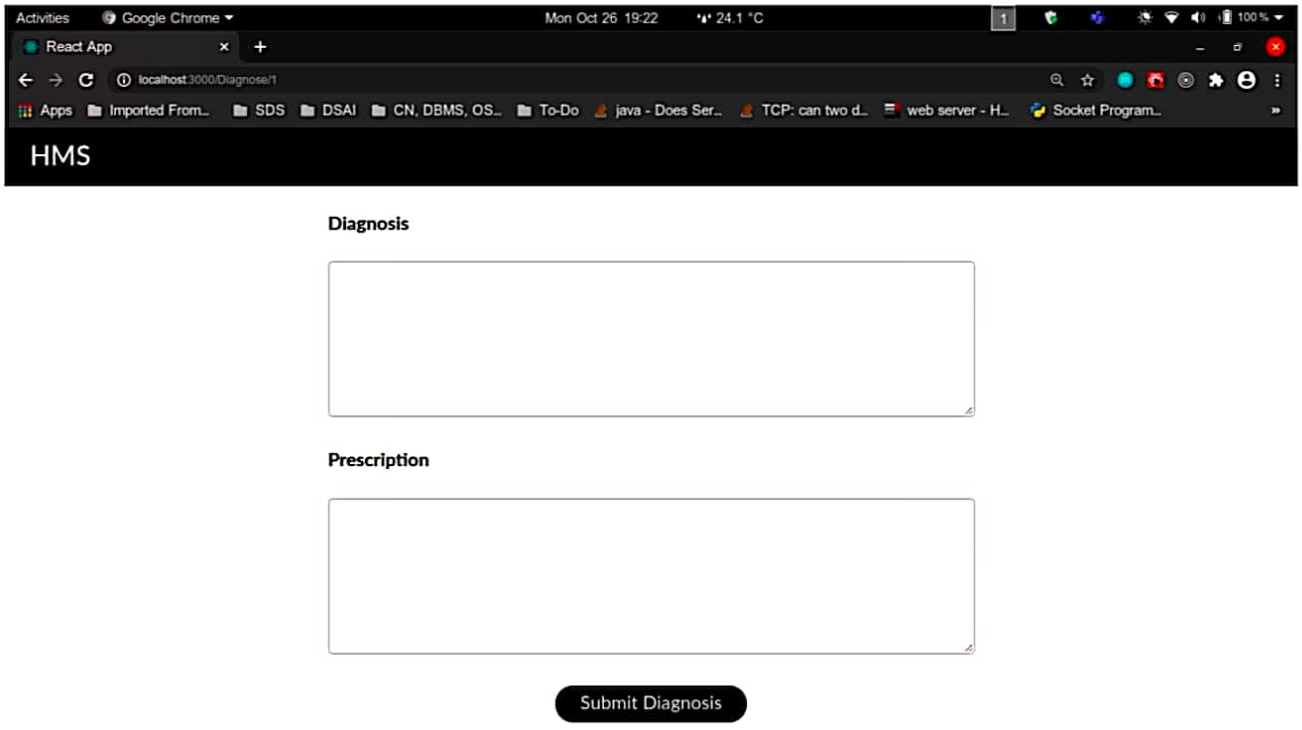


10. Doctor Viewing Appointment:



ID	Name	Date	Start Time	Concerns	Symptoms	Status		
5	Ramesh	17/01/2021	11:00:00	Indigestion	Acidity	NotDone	Diagnose	Cancel
4	Ramesh	23/10/2020	15:00:00	Fever	Cough	NotDone	Diagnose	Cancel
1	Ramesh	15/01/2019	09:00:00	none	itchy throat	Done	Diagnose	
7	Suresh	10/07/2021	09:00:00	Myopia	Not able to see	NotDone	Diagnose	Cancel
6	Suresh	30/10/2020	16:00:00	Sprain	Muscle Pain	NotDone	Diagnose	Cancel

11. Doctor Giving Diagnosis:



12. Doctor Viewing Patient History: