RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(Establishment through Act 18 Of 2008 by Government of AP)

ONGOLE CAMPUS

C E R T I F I C A T E

DATABASE MANAGEMENT SYSTEM LABORATORY

Name of The Faculty: B.MURALI

Designation: ASSISTANT PROFESSOR

Course code	Course name	Course Category	L-T-P	Credits
20CS2182	**Database Management Systems Laboratory**	**PCC**	**0-0-3**	**1.5**

Course Objectives:

1. Analyze the problem and identify the Entities and Relationships, keys for given database.
2. Design, develop and query a database.
3. Able to construct queries and maintain a simple database using MySQL.
4. Normalization of data present in database tables.
5. Develop triggers programs using PL/SQL.

List of Experiments:

1. Designing the Database through Identifying Entities, Relationship Attributes.

MySQL

1. Queries to facilitate acquaintance of Built-In Functions, String Functions, Numeric Functions,
2. Queries to facilitate acquaintance of Date Functions and Conversion Functions.
3. Queries for Creating, Dropping, and Altering Tables
4. Queries using operators in SQL
5. Queries to Retrieve and Change Data: Select, Insert, Delete, and Update
6. Queries using Group By, Order By, and Having Clauses
7. Queries on Controlling Data: Commit, Rollback, and Save point
8. Queries for creating Views, and Constraints
9. Queries on Joins (Outer and Inner joins)
10. Queries on Correlated Sub-Queries

PL/SQL

1. Write a PL/SQL Code using Basic Variable, Anchored Declarations, and Usage of Assignment Operation
2. Write a PL/SQL block using SQL and Control Structures in PL/SQL
3. Write a PL/SQL Code using Cursors, Exceptions and Composite Data Types
4. Write a PL/SQL Code using Procedures, Functions, and Packages FORMS

Course Outcomes:
After completing this course the student must demonstrate the knowledge and ability to:

CO 1	Identify the entities, attributes, relationships, keys for given database.
CO 2	Design a database schema for given problem.
CO 3	Formulate queries using MySQL DML, DDL commands.
CO 4	Formulate SQL queries using constraints and set comparison operators.
CO 5	Apply the normalization techniques for development of application software to realistic problems.
CO 6	Develop PL/SQL programs using triggers, procedures
CO 7	Ability to design and implement given case study.

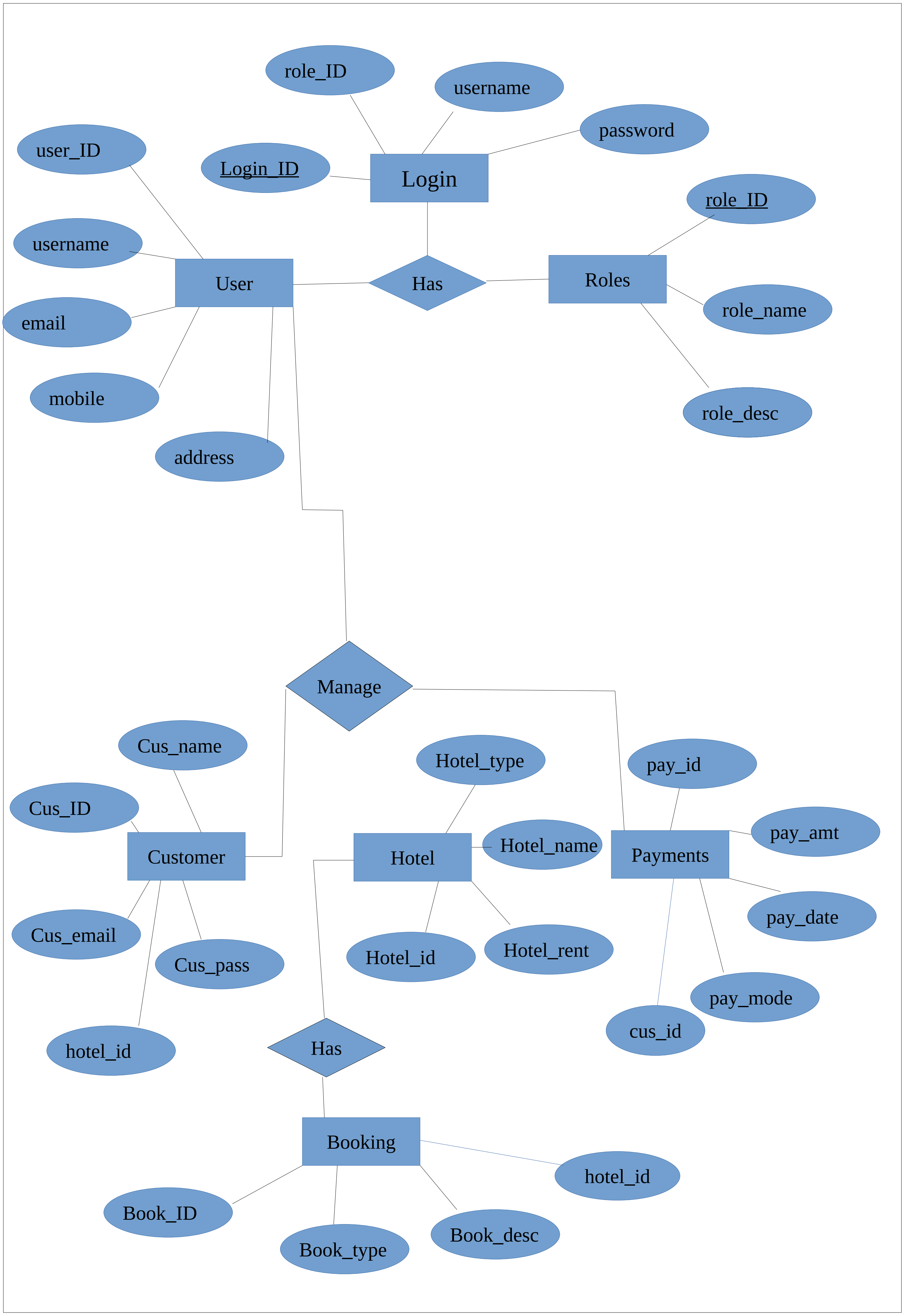
Course Nature		**Practical**		
Assessment Method				
Assessment Tool	Experiments	Record	Viva-Voce/ Quiz/MCQ/Lab project	Total
Weightage (%)	25%	5%	10%	40%
			End Semester Examination weightage (%)	60%

ER DIAGRAMS

AIM: Designing of ER diagram for Hotel Management System

Entity: It is a thing or an element in real world which is distingushable from other objects.

ENTITY	ATTRIBUTES
Login	Loginid,roleid,username,password
User	Userid,email,username,mobile,address
Customer	cust_id,cust_name,cust_pass,cus_mobile,hotel_id
Booking	Book_id,Book_type,Book_desc,hotel_id
Hotel	Hotel_id,Hotel_name,Hotel_rent,Hotel_type
Payments	pay_date,pay_amt,pay_id,pay_mode,cus_id
Roles	role_id,role_name,role_desc


role_ID
username
password
user_ID
Login_ID
Login
username
role_ID
User
Has
Roles
email
role_name
mobile
role_desc
address
Manage
Cus_name
Hotel_type
pay_id
Cus_ID
Hotel_name
pay_amt
Customer
Hotel
Payments
pay_date
Cus_email
Cus_pass
Hotel_id
Hotel_rent
pay_mode
cus_id
hotel_id
Has
Booking
hotel_id
Book_ID
Book_type
Book_desc

LOGIN

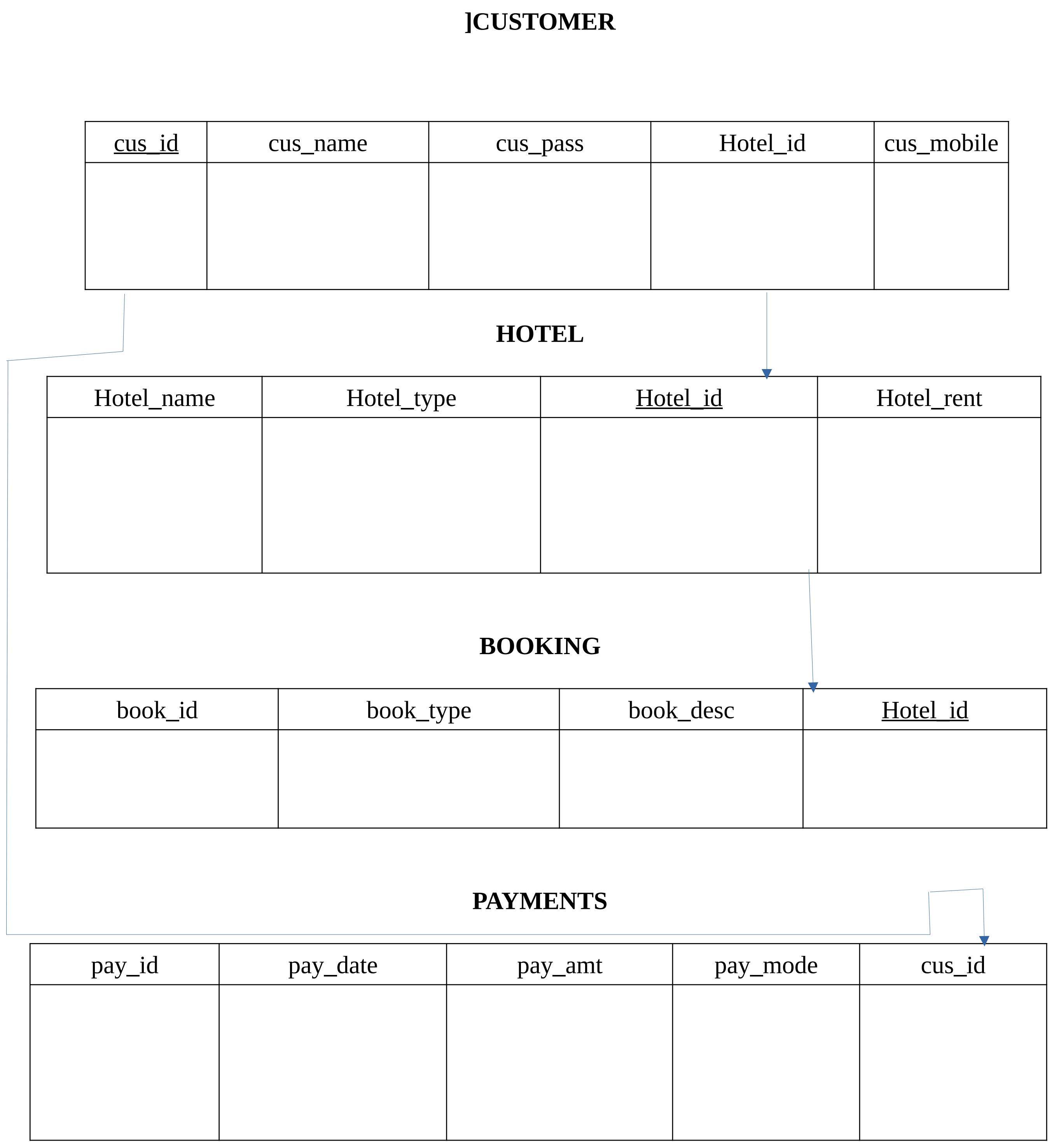
Loginid	roleid	username	password

ROLES

role_id	role_name	role_desc

USER

user_id	username	email	password	address
	\			



]CUSTOMER

cus_id	cus_name	cus_pass	Hotel_id	cus_mobile

HOTEL

Hotel_name	Hotel_type	Hotel_id	Hotel_rent

BOOKING

book_id	book_type	book_desc	Hotel_id

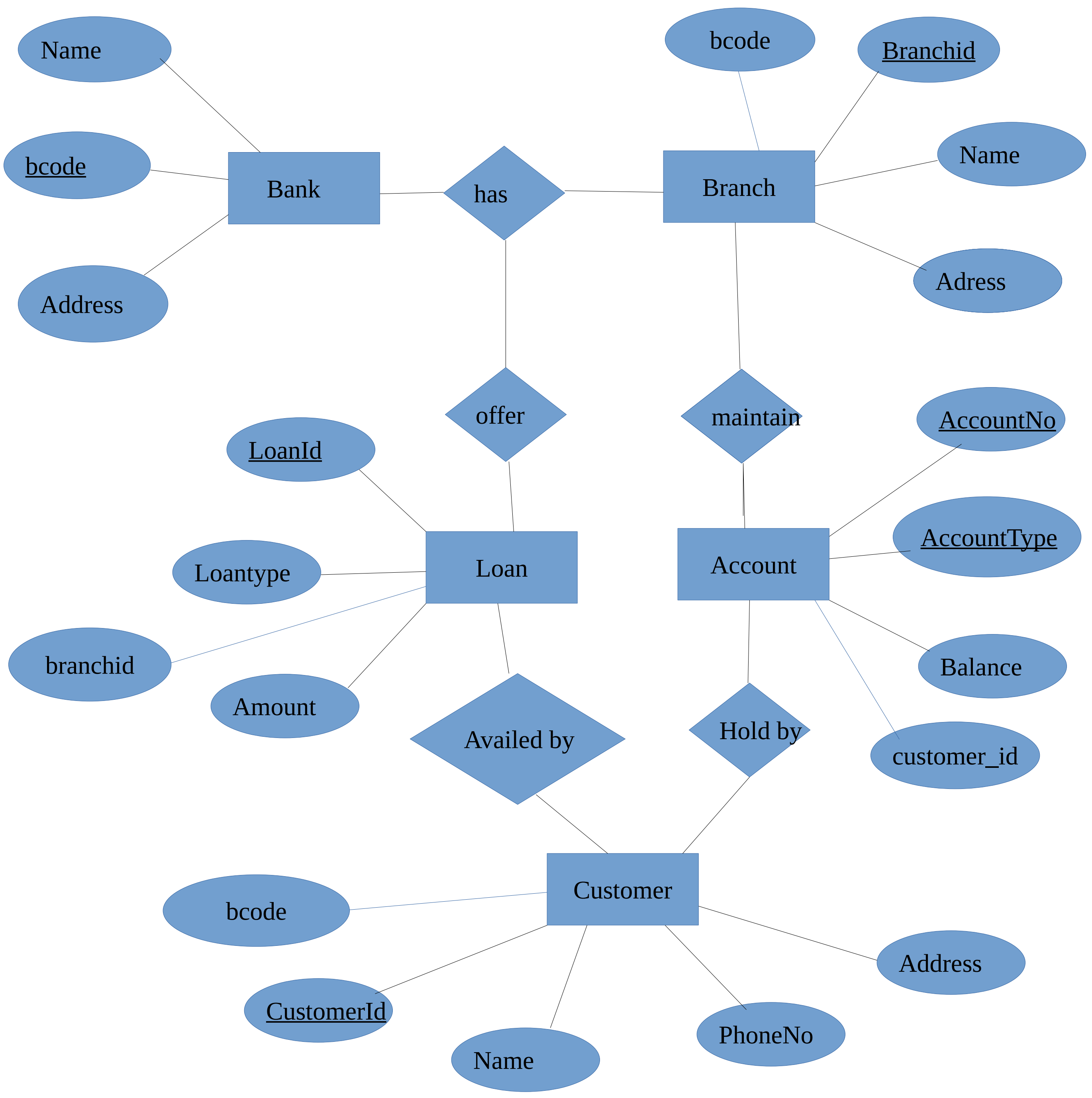
PAYMENTS

pay_id	pay_date	pay_amt	pay_mode	cus_id

AIM: Designing ER diagrams for Bank Management System with 5 entities.

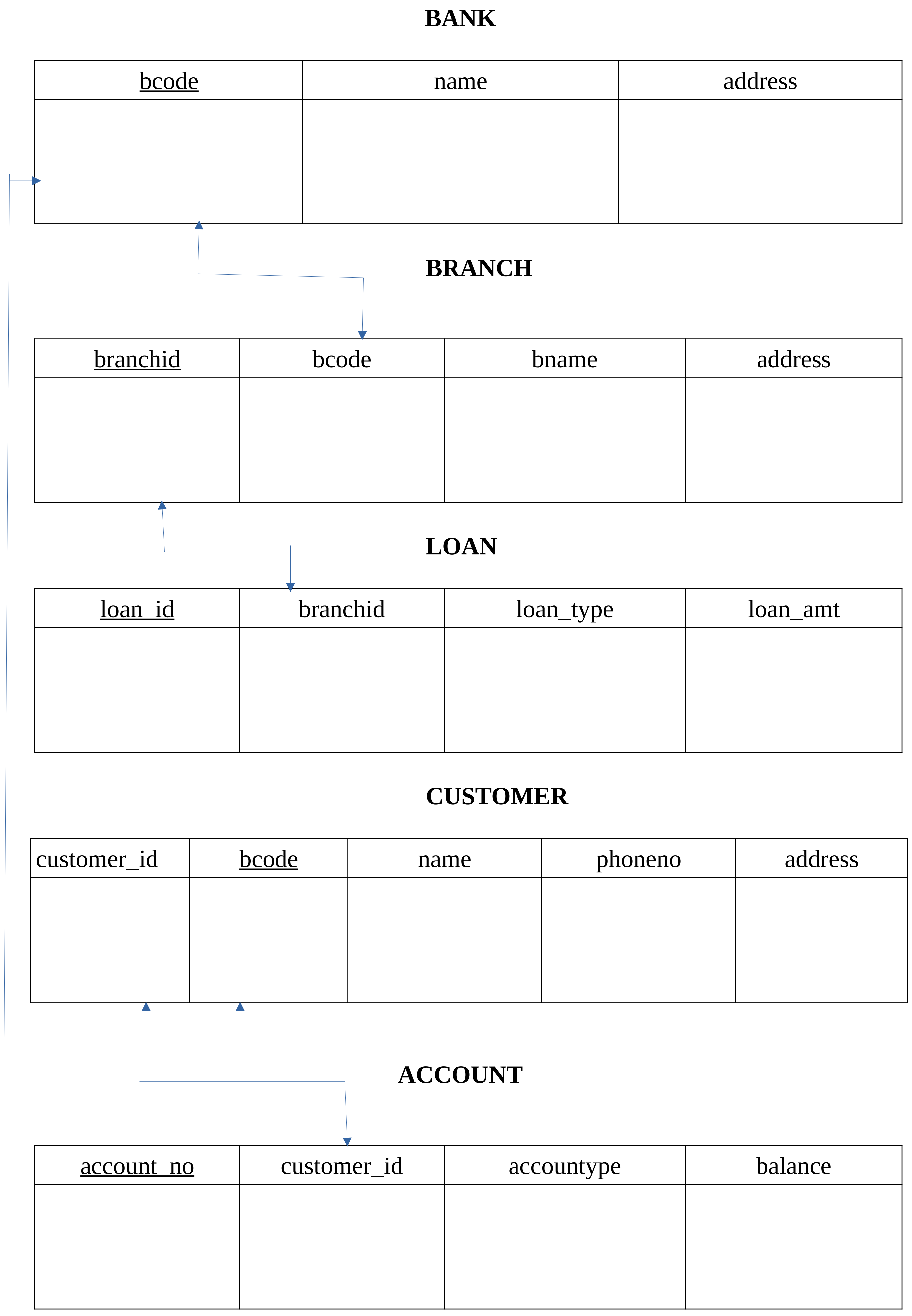
Entity: It is a thing or an element in real world which is distingushable from other objects.

ATTRIBUTES	ENTITIES
Bank	Name,bcode,address
Branch	branch_id,name,address,bcode
Account	acc_no,acc_type,balance,customer_id
Customer	customer_id,name,phoneno,address,bcode
Loan	loan_id,loanamt,loantype,branchid



Name
bcode
Address
Bank
has
Branch
bcode
Branchid
Name
Adress
offer
maintain
LoanId
Loantype
branchid
Amount
Loan
Account
AccountNo
AccountType
Balance
customer_id
Availed by
Hold by
Customer
bcode
CustomerId
Name
PhoneNo
Address

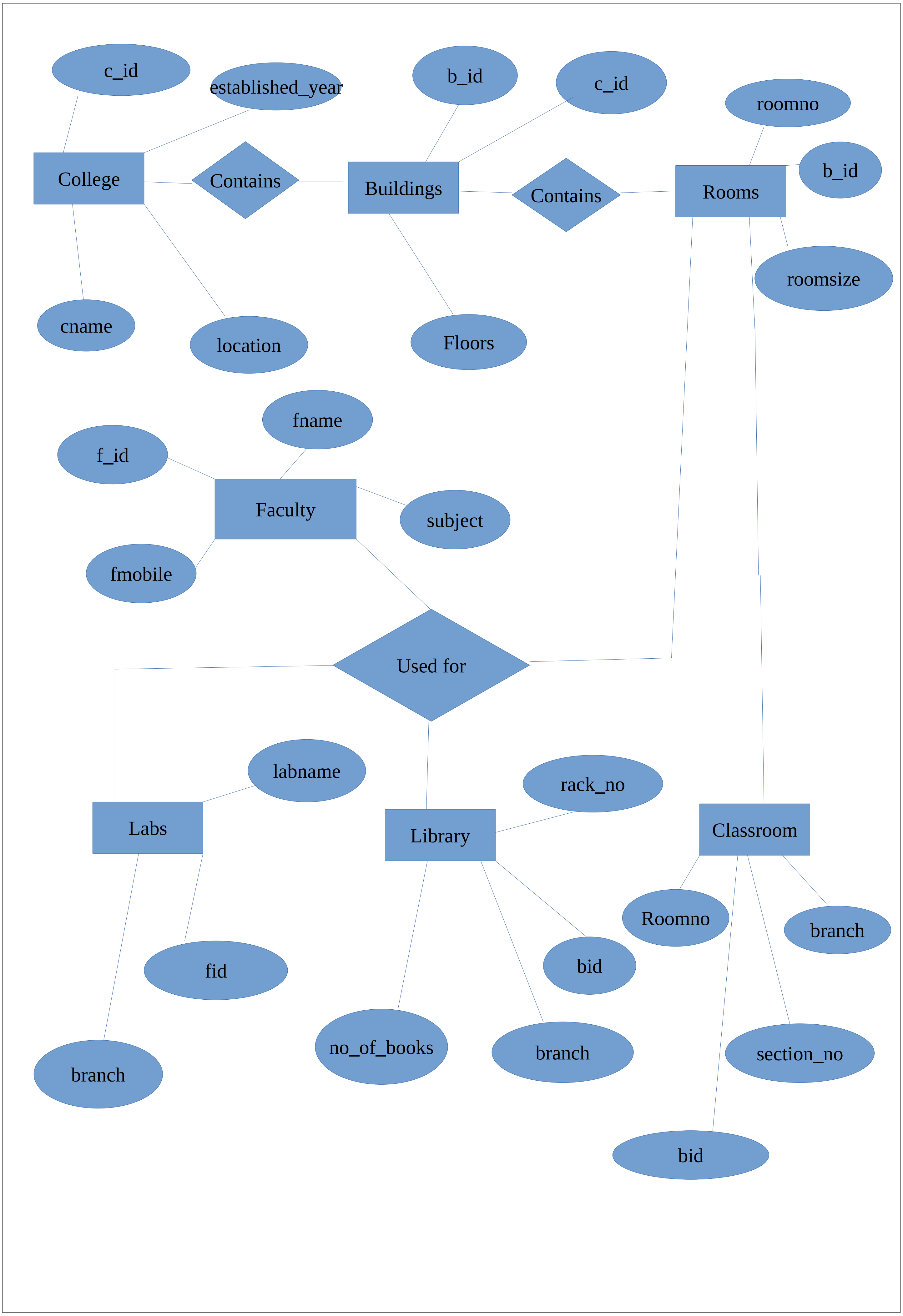
f

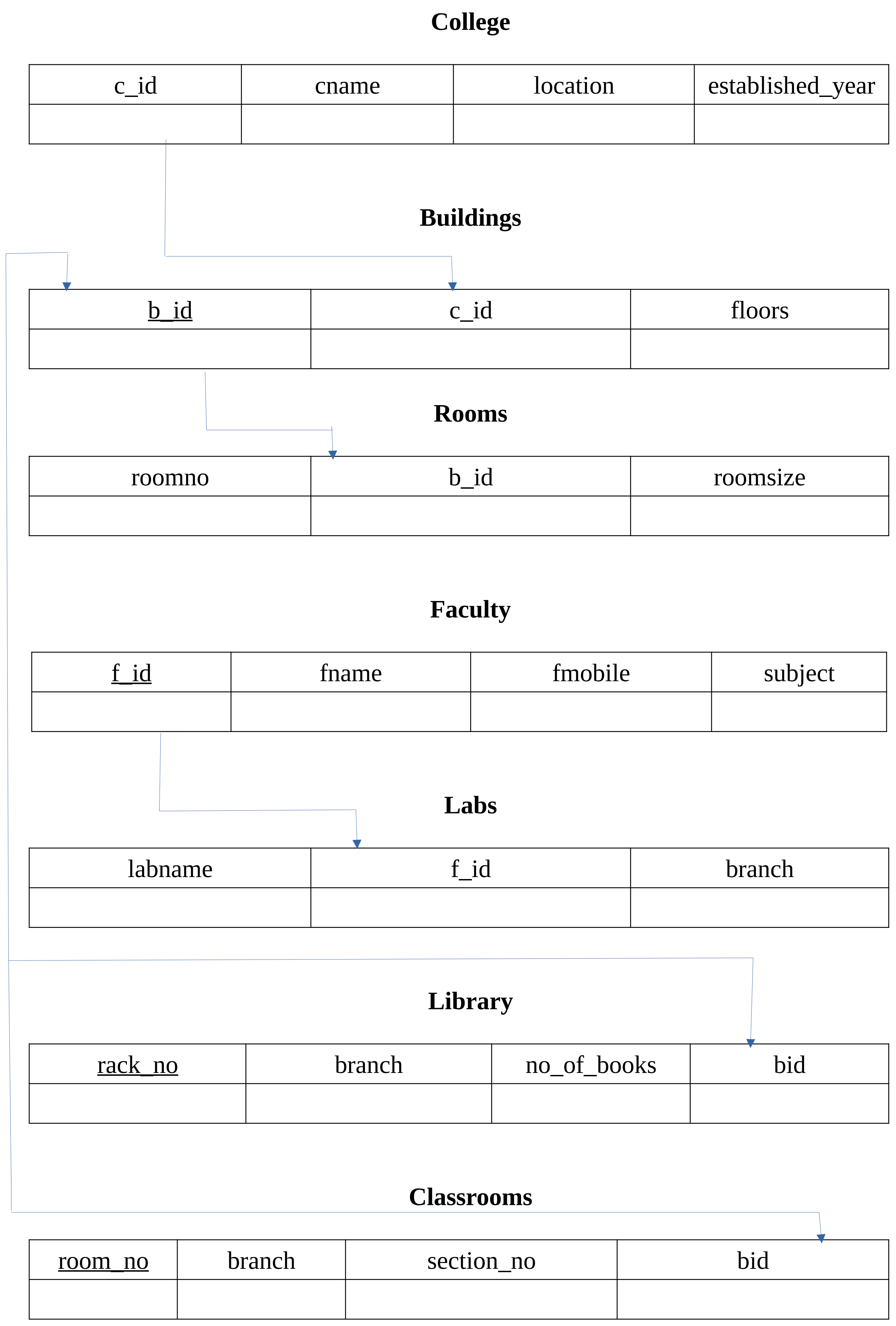

BANK
bcode
name
address
BRANCH
branchid
bcode
bname
address
LOAN
loan_id
branchid
loan_type
loan_amt
CUSTOMER
customer_id
bcode
name
phoneno
address
ACCOUNT
account_no
customer_id
accountype
balance

AIM: Designing ER diagrams for College Management System with 6 entities.

Entity: It is a thing or an element in real world which is distingushable from other objects.

ENTITIES	ATTRIBUTES
College	Name,locatio,established_year
Buildings	reg_id,floors,model
Rooms	Model,roomno,size
Labs	Labname,eqipment,location
Library	Bookslist,inoutbook,registerbook
Classroom	Year,cno,regid


c_id
established_year
b_id
c_id
roomno
College
Contains
Buildings
Contains
Rooms
b_id
roomsize
cname
location
Floors
fname
f_id
Faculty
subject
fmobile
Used for
labname
rack_no
Labs
Library
Classroom
Roomno
branch
bid
fid
no_of_books
branch
section_no
branch
bid



College

c_id	cname	location	established_year

Buildings

b_id	c_id	floors

Rooms

roomno	b_id	roomsize

Faculty

f_id	fname	fmobile	subject

Labs

labname	f_id	branch

Library

rack_no	branch	no_of_books	bid

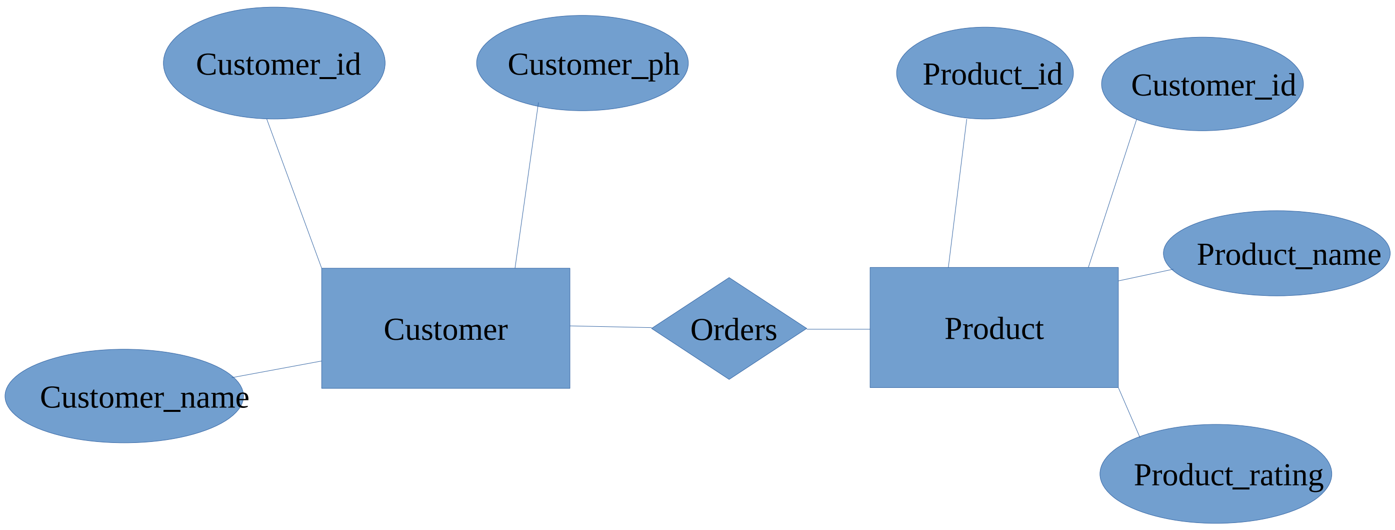
Classrooms

room_no	branch	section_no	bid

AIM: Designing ER diagrams with 2 entities.

Entity: It is a thing or an element in real world which is distingushable from other objects.

ENTITIES	ATTRIBUTES
Customer	customer_id,customer_name,customer_phno
Product	Productid,customer_id,product_name,product_rating


Customer_id
Customer_ph
Product_id
Customer_id
Product_name
Customer
Orders
Product
Customer_name
Product_rating

Customer

customer_id	customer_name	customer_phno

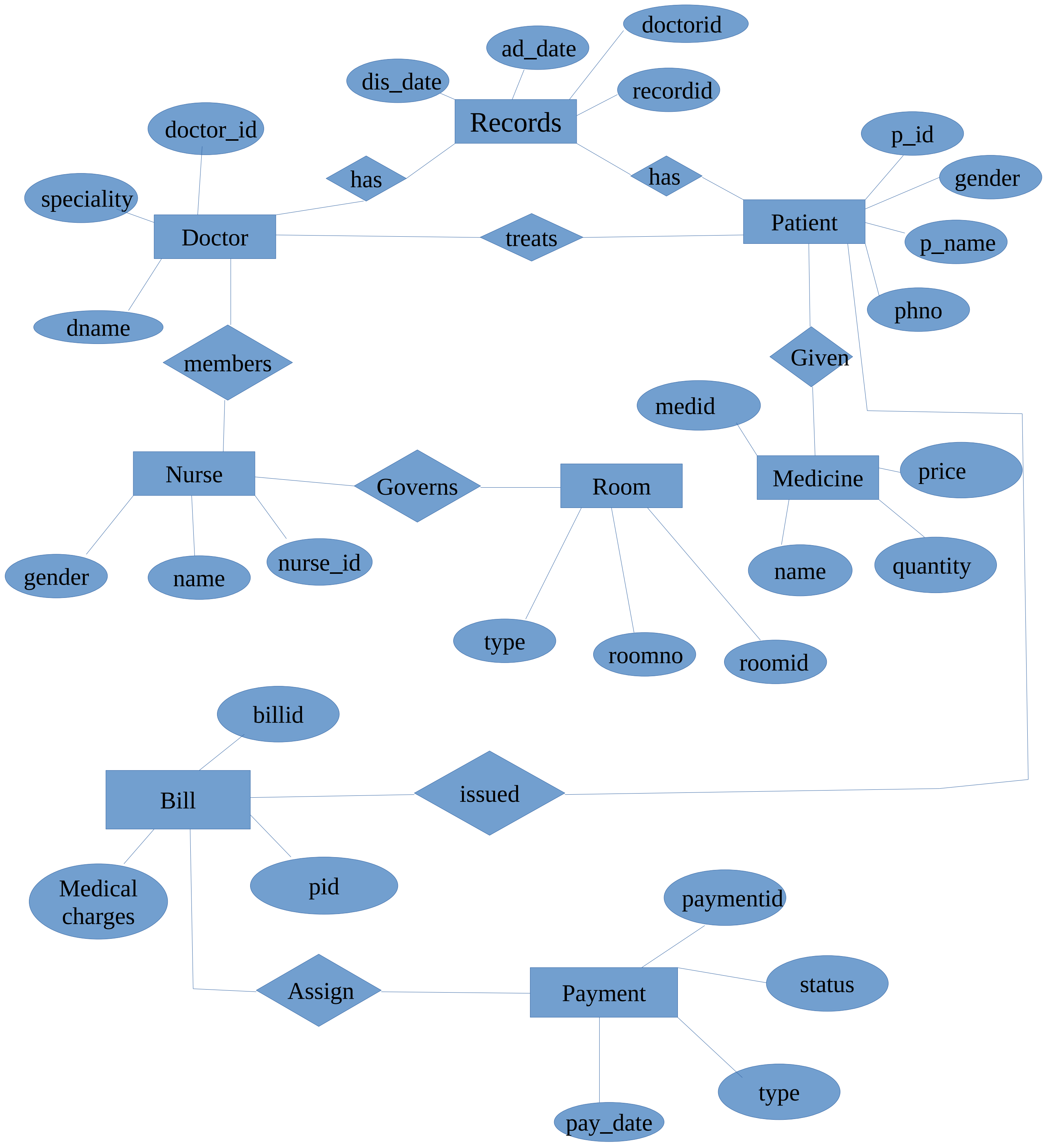
Product

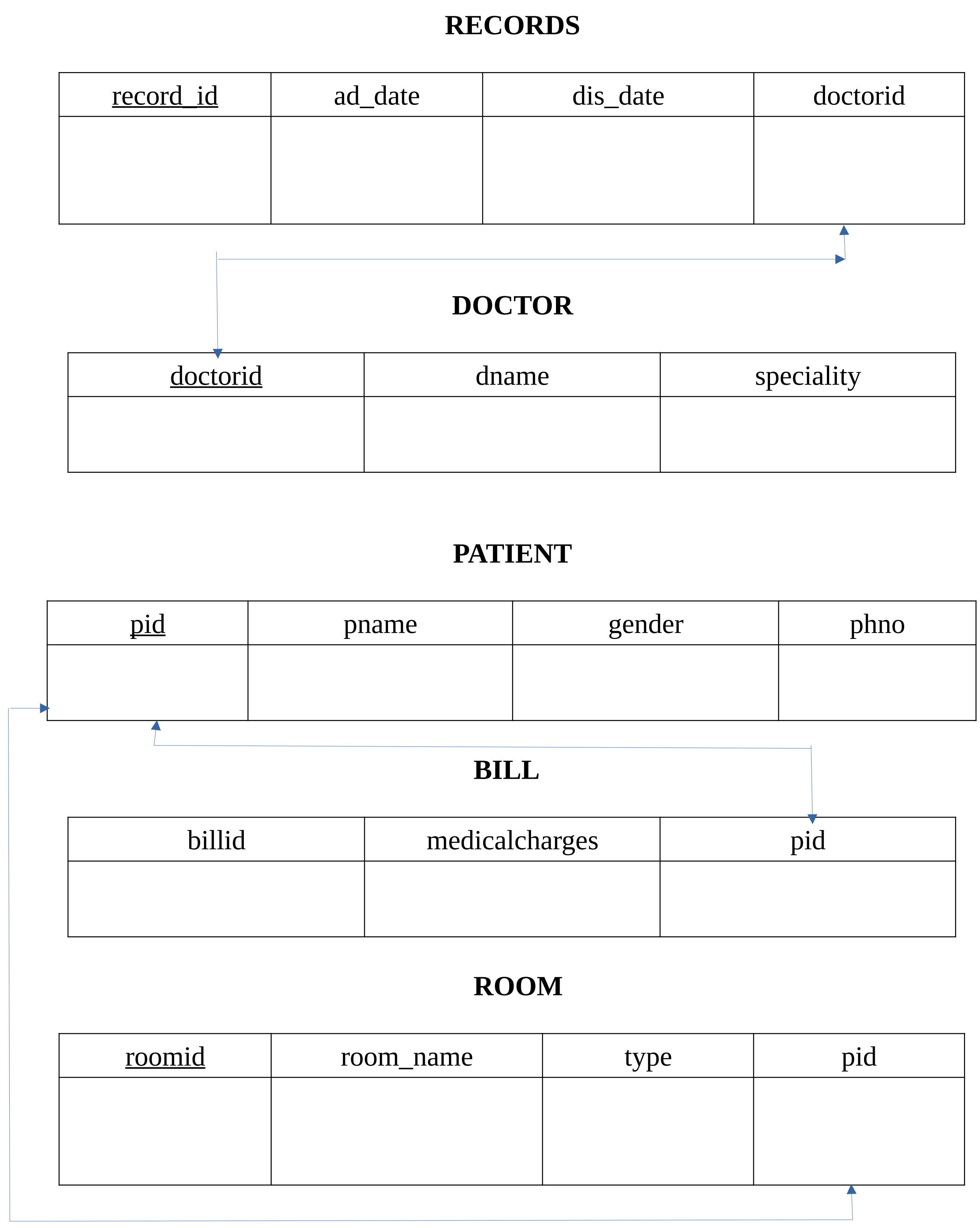
customer_id	productid	product_name	product_rating

AIM: Designing ER diagrams for Hospital Management System with 8 entities.

Entity: It is a thing or an element in real world which is distingushable from other objects.

ENTITIES	ATTRIBUTES
Records	ad_date,dis_date,recordid,doctorid
Patient	Pid,pname,gender,phno
Doctor	Doctorid,dname,speciality
Nurse	Nursename,gender
Room	room_id,room_no,type
Medicine	med_id,price,quantity,name
Bill	Billid,medical_charges,pid
Payment	Paymentid,status,type,name,paydate


doctorid
ad_date
dis_date
recordid
Records
doctor_id
has
has
p_id
speciality
gender
Doctor
treats
Patient
p_name
phno
dname
members
Given
medid
Nurse
Governs
Room
Medicine
price
gender
name
nurse_id
name
quantity
type
roomno
roomid
billid
Bill
issued
Medical charges
pid
paymentid
Assign
Payment
status
type
pay_date



RECORDS

record_id	ad_date	dis_date	doctorid

DOCTOR

doctorid	dname	speciality

PATIENT

pid	pname	gender	phno

BILL

billid	medicalcharges	pid

ROOM

roomid	room_name	type	pid

NURSE

nurseid	name	gender

PAYMENTS

paymentid	status	type	pay_date

MYSQL

PROGRAM NO-1

1. DDL LANGUAGE COMMANDS

AIM:To implementing queries for creating ,dropping and altering a table using DDL language

DESCRIPTION:DDL is means Data Definition Language, it is used to define the database schema and structures.
There are five commands in DDL language:
1.create
2.alter
i.add---adding in specific position
ii.modify
iii.drop
iv.rename
3.rename table
4.truncate
5.drop

1.Creating a table:

The MySQL CREATE TABLE command is used to create a new table into the database.
A table creation command requires three things:
i.Name of the table
ii.Names of fields
iii.Definitions for each field

SYNTAX: CREATE TABLE table_name (column_name data_type(size), column_name data_type(size),…...);

Example:
1.create table cus_tbl(cus_id integer not null auto_increment, cus_firstname varchar(100) not null, cus_surname varchar(100) not null, primary key(cus_id));

-->describe cus_tbl;
output:
```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+---------------
```

2.Alter commands:

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" ,”RENAME”and "MODIFY" commands according to the situation.

i.Add column:

Syntax: ALTER TABLE table_name ADD new_column_name column_definition
[FIRST | AFTER column_name];

Parameters:
table_name: It specifies the name of the table that you want to modify.
new_column_name: It specifies the name of the new column that you want to add to the table.
column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).
FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

Example:
```
mysql>alter table cus_tbl add s_no integer first;
```

```
-->describe cus_tbl;
```

Output:
```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+-----------------------+--------------------------------------------
```

i.adding column in a specific position...

Example:

```
mysql> alter table cus_tbl add cus_phno varchar(20) after cus_id,add cus_Address
varchar(20) after cus_firstname;

mysql> describe cus_tbl;
```

Output:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_phno      | varchar(20)  | YES  |     | NULL    |                |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_Address   | varchar(20)  | YES  |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

ii.modify---changing the datatype of a column:

SYNTAX:

```
1. ALTER TABLE table_name ADD new_column_name column_definition
2. [ FIRST | AFTER column_name ],
3. ADD new_column_name column_definition [ FIRST | AFTER column_name ],
4. ...
5. ;
```

Example:

```
mysql> alter table cus_tbl modify cus_firstname char(20)null;

mysql> describe cus_tbl;
```

Output:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_phno      | varchar(20)  | YES  |     | NULL    |                |
| cus_firstname | char(20)     | YES  |     | NULL    |                |
| cus_Address   | varchar(20)  | YES  |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+-----------------------------------+------------------------------
```

iii.drop---deleting a column

Syntax:_

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

Example:

```
mysql> alter table cus_tbl drop column cus_address;

mysql> describe cus_tbl;
```

Output:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_phno      | varchar(20)  | YES  |     | NULL    |                |
| cus_firstname | char(20)     | YES  |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+-----------------------+--------------------+-----------------------
```

iv.rename--renaming the name of a column

SYNTAX: ALTER TABLE table_name rename COLUMN old_column_name TO new_column_name;

Example:

```
mysql> alter table cus_tbl rename column cus_surname to cus_name;
mysql> describe cus_tbl;
```

Output:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_phno      | varchar(20)  | YES  |     | NULL    |                |
| cus_firstname | char(20)     | YES  |     | NULL    |                |
| cus_name      | varchar(100) | NO   |     | NULL    |                |
+----------------+------------------+-------------------------------
```

3.rename—*renamig the table*

Syntax: ALTER TABLE old_table_name RENAME TO new_table_name;

Example:mysql> alter table cus_tbl rename to customer_tbl;
mysql> show tables;

Output:

```
+---------------+
| Tables_in_ddl |
+---------------+
| customer_tbl  |
+----------+-----
```

4.**truncate**

truncate—deleting all the records not the structure

The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.

Syntax:TRUNCATE TABLE table_name;

Example:

mysql> truncate table customer_tbl;
mysql> describe customer_tbl;

Output:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| s_no          | int          | YES  |     | NULL    |                |
| cus_id        | int          | NO   | PRI | NULL    | auto_increment |
| cus_phno      | varchar(20)  | YES  |     | NULL    |                |
| cus_firstname| char(20)      |YES   |     | NULL    |                |
| cus_name      | varchar(100) | NO   |     | NULL    |                |
------------------------------------------------------------------------
```

5.drop:

MYSQL DROP table statement removes the complete data with structure.

Syntax: DROP TABLE table_name;

Example:

mysql> drop customer_table;
mysql> show tables;

Output:

Empty set (0.00 sec)

PROGRAM NO-2

2.DML LANGUAGE COMMANDS

AIM: To implementing the queries for inserting,updating,deleting and retrieving the data from a table using DML language.

DESCRIPTION:
DML is means Data Manipulation Language which deals with data manipulation, and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE etc, and it is used to store, modify, retrieve,delete and update data in database.
1)INSERT – insert data into a table.
2)UPDATE – updates existing data within a table.
3)DELETE – Delete all records from a database table.
4)SELECT – retrieve data from the database.

1.insert:
MySQL INSERT statement is used to insert data in MySQL table within the database. We can insert single or multiple records using a single query in MySQL.

Syntax:
i.The SQL INSERT INTO command is used to insert data in MySQL table.
Following is a generic syntax:

```
1. INSERT INTO table_name ( field1, field2,...fieldN )
2. VALUES ( value1, value2,...valueN );
```

Field name is optional. If you want to specify partial values, field name is mandatory.
ii. Syntax for all fields:

```
1. INSERT INTO table_name VALUES ( value1, value2,...valueN );
```

Example:

```
//i.insert command for all fileds
mysql> insert into student values(1,'deepika',89,'6300416456');
//ii.insert command for partial fields
mysql> insert into student(sno,sname)values(2,'sai');
//iii.insert command for multiple records
mysql> insert into student values(3,'ganesh',90,'7658962173'),(4,'vasantha',78,'7567269562'),
(5,'gnapika',98,'7387897432');
```

mysql> select * from student;

Output:

```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |  NULL | NULL       |

|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
|    5 | gnapika  |    98 | 7387897432 |
+------+----------+-------+------------
```

2.UPDATE:

MySQL UPDATE statement is used to update data of the MySQL table within the database. It is used when you need to modify the table.

Syntax:UPDATE table_name SET field1=new-value1, field2=new-value2 [WHERE Clause]

Example:

```
  mysql> update student set marks=94,s_phno='9789482978' where sno=2;
  mysql> select * from student;
```

Output:

```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
|    5 | gnapika  |    98 | 7387897432 |
+------+----------+-------+------------
```

3.DELETE:

MySQL DELETE statement is used to delete data from the MySQL table within the database. By using delete statement, we can delete records on the basis of conditions.

Syntax:DELETE FROM table_name WHERE (Condition specified);

Example:

mysql> delete from student where sno=5;
mysql> select * from student;

Output:

```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
+------+----------+-------+------------
```

4.SELECT:

The MySQL SELECT statement is used to fetch data from the one or more tables in MySQL. We can retrieverecords of all fields or specified fields.

Syntax for specified fields:

1. SELECT expressions FROM tables [WHERE conditions];

Syntax for all fields:

1. SELECT * FROM tables [WHERE conditions];

Example:

mysql> select sname,marks from student where marks>85;

Output:

```
+---------+-------+
| sname   | marks |
+---------+-------+
| deepika |    89 |
| sai     |    94 |
| ganesh  |    90 |
+---------+-------
```

PROGRAM NO-3

_3.TCL LANGUAGE COMMANDS

AIM: To implementing the queries for commit,savepoint and rollback the data of a table using TCL language.

DESCRIPTION:TCL means transaction control language.These are used to control the transaction in a database
There are three types in TCL command
1.commit
2.savepoint
3.rollback

1.commit:

It commits the current transaction means making changes permanently.
Syntax:commit;
Example:

```
mysql> select * from student;
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
+------+----------+-------+------------+
mysql> commit;
```

2.savepoint:

It is used to store data temporarily.It is an restoring point which lies in between the commit and rollback.
Syntax:savepoint savepoint_name;
Example:

```
mysql> start transaction;
mysql> savepoint 2rows;
mysql> insert into student values(5,'gnapika',56,'7789347893');
mysql> select * from student;
```

Output:

```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |

|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
|    5 | gnapika  |    56 | 7789347893 |
+------+----------+-------+------------+
mysql> rollback to 2rows;
mysql> select * from student;
```
Output:
```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
+------+----------+-------+------------
```

3.rollback:

The rollback statement restores the database changes to the starting of transaction on i.e previous commit.

Syntax:rollback;

Example:
```
mysql> start transaction;
mysql> select * from student;
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
-------------------------------------------

mysql> insert into student values(5,'gnapika',56,'7789347893');
mysql> rollback;
```
Output:
```
mysql> select * from student;
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
------------------------------------------
```

PROGRAM NO-4

4.DCL LANGUAGE COMMANDS

AIM: To apply DCL commands on a database.

DESCRIPTION:DCL full form Data Control Language.It is used to provide control to the data in a database.
There are two types in DCL commands.
1.grant
2.revoke

1.grant:

It is used to provide access or privilages or permissions on the database objects to the user.

//creating user name to give acces...

*SYNTAX:*CREATE USER 'username' IDENTIFIED BY 'password';

```
mysql>  create user deepu@localhost identified by 'Deepika123#@!';
```

Syntax:GRANT permission_type ON database.table TO 'username'@'localhost';

Example:
```
mysql> grant insert,select on deepika.student to deepu@localhost;
```

//open in new tab
```
mysql -u deepu -p;

Enter password: Deepika123#@!
```

//performing insert,select....

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| deepika            |
| information_schema |
| performance_schema |
+--------------------+
mysql> use deepika;

Database changed
```

```
mysql> insert into student values(4,'gnapika',76,'9378326432');
mysql> select * from student;
```

Output:

```
+------+----------+-------+------------+
| sno  | sname    | marks | s_phno     |
+------+----------+-------+------------+
|    1 | deepika  |    89 | 6300416456 |
|    2 | sai      |    94 | 9789482978 |
|    3 | ganesh   |    90 | 7658962173 |
|    4 | vasantha |    78 | 7567269562 |
|    4 | gnapika  |    76 | 9378326432 |
+------+----------+-------+------------
```

2.Revoke:

To take back privileges from a specific user, use the REVOKE command. It works similar to the GRANT command.

Syntax:REVOKE permission_type ON database.table TO 'username'@'localhost';

//revoke the insert permission in the original tab

it is used to remove access right or privilages to the database objects

Example:

```
mysql> revoke insert on deepika.student from deepu@localhost;

mysql> insert into student values(4,'gnapika',76,'9378326432');
```

Output:

```
ERROR 1142 (42000): INSERT command denied to user 'deepu'@'localhost' for table
'student'
```

PROGRAM NO-5

5.OPERATORS

AIM: To apply operators in database tables.

DESCRIPTION:

To retrieve data from the table based on some rules, then use conditional operators. Conditional operators return true or false esteem based on the instance of the variables.

1.comparison operators--- **=,<,>,<=,>=,!=**

2.arithmetic operators---**+,-,*,/,%**

3.logical operators-**--AND,OR,NOT**

4.special operators**—between,isnull,is notnull,like,in**

5.relational set operators**—union,unionall,intersection,minus**

```
mysql> select * from employee;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|    1 | deepika | 27899  |      1 |
|    2 | sai     | 56576  |      2 |
|    3 | ganesh  | 60000  |      3 |
|    4 | vasuu   | 78000  |      2 |
|    5 | gnapika | 54565  |      3 |
|   6|  dinesh|     null    |    4    |
+-------+---------+--------+--------+
5 rows in set (0.00 sec)
```

1.comparision operators:

comparison operators are = ,>,<,>=,<=.

Syntax:select * from table_name where column_name [operator] value;

EXAMPLE:

i.mysql> select *from employee where empsal=78000;

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|    4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------
```

```
ii.mysql> select *from employee where empsal<60000;
+-------+---------+--------+--------
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------

iii.mysql> select *from employee where empsal>55000;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------

iv.mysql> select *from employee where empsal<=60000;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------

v.mysql> select *from employee where empsal>=60000;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------
vi.mysql> select *from employee where empsal!=78000;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

2.arithmetic operators{+,-,*,/,%}

Arithmetic operators are +,-,*,/ we can use the arithmetic operators with the table attribute in a conditional expression

Syntax:select column_name [operator] value from table_name;

Examples:

```
mysql> select empsal+1000 from employee;
+-------------+
| empsal+1000 |
+-------------+
|       28899 |
|       57576 |
|       61000 |
|       79000 |
|       55565 |
+-------------+
```

```
mysql> select empsal-1000 from employee;
+-------------+
| empsal-1000 |
+-------------+
|       26899 |
|       55576 |
|       59000 |
|       77000 |
|       53565 |
+-------------+
```

```
mysql> select empsal*2 from employee;
+----------+
| empsal*2 |
+----------+
|    55798 |
|   113152 |
|   120000 |
|   156000 |
|   109130 |
+----------+
mysql> select empsal/2 from employee;
+----------+
| empsal/2 |
+----------+
|  13949.5 |
|    28288 |
|    30000 |
|    39000 |
|  27282.5 |
+----------+
```

```
mysql> select empsal%2 from employee;

+----------+
| empsal%2 |
+----------+
|        1 |
|        0 |
|        0 |
|        0 |
|        1 |
+----------
```

3.LOGICAL OPERATORS{AND,OR,NOT}

The logical operators are AND, OR, NOT, are used in between two or more conditional expression.

i.AND
To specify multiple conditions in the column of MySQL table use Logical AND operator. i.e. MySQL Logical And operator compares two results and returns true if both of the results are true.

Syntax: Select * from <table_name> where <condition1> and <condition2> and <condition3>;
Example:
mysql> select * from employee where deptno=3 and empsal>50000;
Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

ii.OR
To specify multiple conditions on different columns in MySQL table use Logical OR operator. MySQL Logical OR operator compares two expressions and returns true if both of the expressions is valid.

Syntax: Select * from <table_name> where <condition1> or <condition2> or <condition3>;

Example:
mysql> select * from employee where deptno=1 or deptno=2 or deptno=3;
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

iii.NOT EQUAL
MySQL NOT EQUAL operator is used to return a set of rows from the table, after making sure that two expressions placed on either sides of the table are NOT NULL.

Syntax: Select * from <table_name> where <column_name> is not equal to;
Example:
mysql> select * from employee where deptno<>3;
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------
```

4.special operators(BETWEEN,ISNULL,ISNOTNULL,LIKE,IN,ALL,ANY)

i.between
MySQL BETWEEN AND operator checks whether a value is present between minimum and maximum range of an expression.

Syntax: Select * from <table_name> where <column_name> between minimum and maximum;

Example:
mysql> select * from employee where empsal between 50000 and 60000;
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

ii.isnull
MySQL IS NULL operator will review whether a esteem is NULL.

Syntax : Select * from <table_name> where <column_name> is null;
Example:
mysql> select * from employee where empsal is null;
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     6 | dinesh  | NULL   |      4 |
+-------+---------+--------+--------
```

iii.isnotnull
MYSQL IS NOT NULL operator will review whether the esteem IS Not Null or not.

Syntax : select * from <table_name> where <column_name> is not null.
Example:
mysql> select * from employee where empsal is not null;
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

iv.like
The Like operation is used with wildcard characters [%, -]
To find out the patterns within the string attributes % it means any and all following or preceding characters.
_ (Under score) it means any single characters

Example:
mysql> select * from employee where empname like '___p%';
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     5 | gnapika | 54565  |      3 |
+-------+---------+--------+--------
```

v.in
This operator is used to check whether all attribute values match any value with in a value list. The values in the list are all of same data types
Syntax:select * from table_name where column_name in(value1,value2,....);
Example:
mysql> select * from employee where empsal in(60000,78000);
Output:
```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
--------------------------------------------
```

vi.all
The ALL operator:

- returns a boolean value as a result
- returns TRUE if ALL of the subquery values meet the condition
- is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

SYNTAX:SELECT ALL *column_name(s)*
FROM *table_name*
WHERE *condition*;

EXAMPLE:mysql> select * from employee where 5>all(select deptno from employee);*//true*

OUTPUT:
```
+-------+---------+--------+--------+
| empno | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika |  27899 |      1 |
|     2 | sai     |  56576 |      2 |
|     3 | ganesh  |  60000 |      3 |
|     4 | vasuu   |  78000 |      2 |
|     5 | gnapika |  54565 |      3 |
|     6 | dinesh  |   NULL |      4 |
+-------+---------+--------+--------+
6 rows in set (0.00 sec)
```

2)mysql> select * from employee where 5<all(select deptno from employee);*//false*
OUTPUT:
Empty set (0.00 sec)

V.ANY
The ANY operator:

- returns a boolean value as a result
- returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

SYNTAX:SELECT *column_name(s)* FROM *table_name* WHERE *column_name operator* ANY (SELECT *column_name* FROM *table_name* WHERE *condition*);
EXAMPLE:
1)mysql> select * from employee where 2>any(select deptno from employee);*//true*
OUTPUT:
```
+-------+---------+--------+--------+
| empno | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika |  27899 |      1 |
|     2 | sai     |  56576 |      2 |
|     3 | ganesh  |  60000 |      3 |
|     4 | vasuu   |  78000 |      2 |
|     5 | gnapika |  54565 |      3 |
|     6 | dinesh  |   NULL |      4 |
+-------+---------+--------+--------+
6 rows in set (0.00 sec)
```

2)mysql> select * from employee where 1>any(select deptno from employee);*//false*
OUTPUT:
Empty set (0.00 sec)

5.RELATIONAL SET OPERATIONS

SQL SET Operators are behaving same like mathematical sets, these sql set operators are classified into four types, which is given below;
❖Union
❖Union all
❖Minus
❖Intersect

```
mysql> select * from s1;
+------+-------+--------+
| sid  | sname | smarks |
+------+-------+--------+
|    1 | deepu |     89 |
|    2 | sai   |     67 |
|    3 | ganii |     78 |
|    4 | vasuu |     65 |
+------+-------+--------
```

```
mysql> select * from s2;
+------+---------+--------+
| sid  | sname   | smarks |
+------+---------+--------+
|    1 | deepu   |     89 |
|    2 | gnapika |     67 |
|    3 | ganii   |     78 |
|    4 | dinesh  |     65 |
+------+---------+--------
```

1.UNION

Union operator returns all the value from the entire table. But it do not include duplicate values, it means it not return duplicate values.

Syntax:select * from table_name1 union select * from table_name2;

Example:

mysql> select * from s1 union select * from s2;

Output:

```
+------+---------+--------+
| sid  | sname   | smarks |
+------+---------+--------+
|    1 | deepu   |     89 |
|    2 | sai     |     67 |
|    3 | ganii   |     78 |
|    4 | vasuu   |     65 |
|    2 | gnapika |     67 |
|    4 | dinesh  |     65 |
+------+---------+--------
```

2.UNION ALL

Union all operator returns all the value from the entire table including duplicate values.

Syntax:select * from table_name1 unionall select * from table_name2;

Example:

mysql> select * from s1 union all select * from s2;

Output:

```
+------+---------+--------+
| sid  | sname   | smarks |
+------+---------+--------+
|    1 | deepu   |     89 |
|    2 | sai     |     67 |
|    3 | ganii   |     78 |
|    4 | vasuu   |     65 |
|    1 | deepu   |     89 |
|    2 | gnapika |     67 |
|    3 | ganii   |     78 |
|    4 | dinesh  |     65 |
+------+---------+--------
```

3.INTERSECT

Intersect operator return the common value from all the variables

Syntax:select * from table_name1 intersection select * from table_name2;

Example:select * from s1 union select * from s2;

Output:

```
+------+-------+--------+
| sid  | sname | smarks |
+------+-------+--------+
|    1 | deepu |     89 |
|    3 | ganii  |     78 |
+------+------+--------
```

4.MINUS:

Minus operator return the values which are not available in first table but not available in second table.

Syntax:select * from table_name1 minus select * from table_name2;

Example:

select * from s1 minus select * from s2;

Output:

```
+------+-------+--------+
| sid  | sname | smarks |
+------+-------+--------+
|    2 | sai   |     67 |
|    4 | dinesh  |     65 |
+------+---------+--------
```

PROGRAM NO-6

6.CLAUSES

AIM: To apply clauses in database tables.

DESCRIPTION:

Clause is defined as a set of rules, that makes to understand the concepts of MySQL command in Database.
MySQL Clauses are very similar to SQL clause, except some functional operations.
Types of clauses are
1.Distinct clause.
2.Where clause.
3.Group By clause.
4.Having clause.
5.Order by clause .

1.Distinct clause:

Used to eliminate the duplicate values in a specific column

Syntax:select distinct column_name from table_name;

Example:

```
mysql> select * from employee;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | NULL   |      3 |
+-------+---------+--------+--------
```

Output:

```
mysql> select distinct deptno from employee;
+--------+
| deptno |
+--------+
|      1 |
|      2 |
|      3 |
+--------
```

2.where clause:

Where clause will specify the condition for the result set in the table.

Syntax:select * from table_name where column_name ='value';

Example:

mysql> select empname from employee where empsal>55000;

Output:

```
+---------+
| empname |
+---------+
| sai     |
| ganesh  |
| vasuu   |
+---------
```

3.Groupby clause:

To divide the data of a table into groups based on single column use Group By clause.

Syntax:select columnname,condition from table_name group by column_name;

Example:

mysql> select deptno,max(empsal) from employee group by deptno;

Output:

```
+--------+-------------+
| deptno | max(empsal) |
+--------+-------------+
|      1 | 27899       |
|      2 | 78000       |
|      3 | 60000       |
+--------+-------------
```

4.Having clause:

To specify a condition with group by clause use Having clause.

Syntax:select columnname,condition from table_name group by column_name having condition;

Example:

mysql> select deptno,max(empsal) from employee group by deptno having max(empsal)>55000;

Output:

```
+--------+-------------+
| deptno | max(empsal) |
+--------+-------------+
|      2 | 78000       |
|      3 | 60000       |
+--------+-------------
```

5.order by clause:

To arrange data of a table either in ascending order or descending order based on single column use Orderby clause i.e., Select * from the table name will show all the rows from the table and order by clause is used to arrange the specific column in ascending or descending order.

Syntax:select * from table_name order by column_name;

Example:

mysql> select * from employee order by empsal;

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     6 | dinesh  | NULL   |      3 |
|     1 | deepika | 27899  |      1 |
|     5 | gnapika | 54565  |      3 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------+
```

//descending order

mysql> select * from employee order by empsal desc;

output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     4 | vasuu   | 78000  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     2 | sai     | 56576  |      2 |
|     5 | gnapika | 54565  |      3 |
|     1 | deepika | 27899  |      1 |
|     6 | dinesh  | NULL   |      3 |
+-------+---------+--------+--------
```

PROGRAM NO-7

7.JOINS

AIM: To implement the queries of joins in tables in a database.

DESCRIPTION:
Join is used to retrieve the data from more than one table in a single query
There are following joins are available in SQL:
1.Equi join.
2.Cartesian join[cross join].
3.Non-equi join.
4.Outer join.
5.Self join.

1.equi join:

√ It is used to retrieve the data from more than 1 table based on "equality" condition.
√ Tables must have common column between them to apply this join.
√ If N tables are joined N-1 conditions are applied.
Syntax:select * from table1_name,table2_name where condition;
Example:
Table1:
```
mysql> select * from dept;
+--------+-----------+
| deptno | deptname  |
+--------+-----------+
|      1 | software  |
|      2 | hardware  |
|      3 | developer |
|      4 | tester    |
+--------+-----------
```
Table2:
```
mysql> select * from employee;
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     1 | deepika | 27899  |      1 |
|     2 | sai     | 56576  |      2 |
|     3 | ganesh  | 60000  |      3 |
|     4 | vasuu   | 78000  |      2 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | NULL   |      3 |
+-------+---------+--------+--------+
```

Example:
mysql> select empname,deptname from employee,dept where employee.deptno=dept.deptno;
Output:
```
+---------+-----------+
| empname | deptname  |
+---------+-----------+
| deepika | software  |
| sai     | hardware  |
| ganesh  | developer |
| vasuu   | hardware  |
| gnapika | developer |
| dinesh  | developer |
------------------------
```

2) Cartesian join: -[cross join]

√ It is used to retrieve the data from the multiple tables without any conditions.
√ It is used to retrieve data analysis report.
√ No need to have common column between tables to apply this type of join

Syntax:select * from table1_name crossjoin table2_name;

Example1:

mysql> select * from employee cross join dept;

Output:

```
+-------+---------+--------+--------+--------+-----------+
| empid | empname | empsal | deptno | deptno | deptname  |
+-------+---------+--------+--------+--------+-----------+
|     1 | deepika | 27899  |      1 |      4 | tester    |
|     1 | deepika | 27899  |      1 |      3 | developer |
|     1 | deepika | 27899  |      1 |      2 | hardware  |
|     1 | deepika | 27899  |      1 |      1 | software  |
|     2 | sai     | 56576  |      2 |      4 | tester    |
|     2 | sai     | 56576  |      2 |      3 | developer |
|     2 | sai     | 56576  |      2 |      2 | hardware  |
|     2 | sai     | 56576  |      2 |      1 | software  |
|     3 | ganesh  | 60000  |      3 |      4 | tester    |
|     3 | ganesh  | 60000  |      3 |      3 | developer |
|     3 | ganesh  | 60000  |      3 |      2 | hardware  |
|     3 | ganesh  | 60000  |      3 |      1 | software  |
|     4 | vasuu   | 78000  |      2 |      4 | tester    |
|     4 | vasuu   | 78000  |      2 |      3 | developer |
|     4 | vasuu   | 78000  |      2 |      2 | hardware  |
|     4 | vasuu   | 78000  |      2 |      1 | software  |
|     5 | gnapika | 54565  |      3 |      4 | tester    |
|     5 | gnapika | 54565  |      3 |      3 | developer |
|     5 | gnapika | 54565  |      3 |      2 | hardware  |
|     5 | gnapika | 54565  |      3 |      1 | software  |
|     6 | dinesh  | NULL   |      3 |      4 | tester    |
|     6 | dinesh  | NULL   |      3 |      3 | developer |
|     6 | dinesh  | NULL   |      3 |      2 | hardware  |
|     6 | dinesh  | NULL   |      3 |      1 | software  |
+-------+---------+--------+--------+--------+-----------
```

Example2:
mysql> select empname,deptname from employee cross join dept;
Output:

```
+---------+-----------+
| empname | deptname  |
+---------+-----------+
| deepika | tester    |
| deepika | developer |
| deepika | hardware  |
| deepika | software  |
| sai     | tester    |
| sai     | developer |
| sai     | hardware  |
| sai     | software  |
| ganesh  | tester    |
| ganesh  | developer |
| ganesh  | hardware  |
| ganesh  | software  |
| vasuu   | tester    |
| vasuu   | developer |
| vasuu   | hardware  |
| vasuu   | software  |
| gnapika | tester    |
| gnapika | developer |
| gnapika | hardware  |
| gnapika | software  |
| dinesh  | tester    |
| dinesh  | developer |
| dinesh  | hardware  |
| dinesh  | software  |
-------------------------
```

3.Non Equi-join:

√ It is used to retrieve the data from multiple tables based on other condition but not equal.
√ Non Equi Join uses all comparison operators except the equal (=) operator like !=, >=, <=, <, >.

Syntax:select * from table_name1,table_name2 where condition(notequal);

Example:
mysql> select empname,deptname from employee,dept where employee.deptno!
=dept.deptno;
Output:
```
+---------+-----------+
| empname | deptname  |
+---------+-----------+
| deepika | tester    |
| deepika | developer |
| deepika | hardware  |
| sai     | tester    |
| sai     | developer |
| sai     | software  |
| ganesh  | tester    |
| ganesh  | hardware  |
| ganesh  | software  |
| vasuu   | tester    |
| vasuu   | developer |
| vasuu   | software  |
| gnapika | tester    |
| gnapika | hardware  |
| gnapika | software  |
| dinesh  | tester    |
| dinesh  | hardware  |
| dinesh  | software  |
+---------+-----------
```

4.Outer join:

This join returns all the rows from one table and only those rows from second table which meets the condition.
3types:
 a)left outer join
 b)right outer join
c)full outer join

a)left outer join:

√ Returns all the rows from left table(ie, first table) and rows that meet the condition from second table.
√ All the rows from left table and only matching rows from the right table.

Syntax:select * from table_name1 left outer join table_name2 on condition;

Example:
mysql> select * from employee left outer join dept on employee.deptno=dept.deptno;
Output:
```
+-------+---------+--------+--------+--------+-----------+
| empid | empname | empsal | deptno | deptno | deptname  |
+-------+---------+--------+--------+--------+-----------+
|     1 | deepika | 27899  |      1 |      1 | software  |
|     2 | sai     | 56576  |      2 |      2 | hardware  |
|     3 | ganesh  | 60000  |      3 |      3 | developer |
|     4 | vasuu   | 78000  |      2 |      2 | hardware  |
|     5 | gnapika | 54565  |      3 |      3 | developer |
|     6 | dinesh  | NULL   |      3 |      3 | developer |
|     7 | ramesh  | 45000  |      7 |   NULL | NULL      |
---------------------------------------------------------
```

b)right outer join:

√ Returns all the rows from right table(ie, second table) and rows that meet the condition from first table.
√ All rows from right table and only matching rows from the left table.

Syntax:select * from table_name1 right outer join table_name2 on condition;
Example:
mysql> select * from employee right outer join dept on employee.deptno=dept.deptno;
Output:
```
+-------+---------+--------+--------+--------+-----------+
| empid | empname | empsal | deptno | deptno | deptname  |
+-------+---------+--------+--------+--------+-----------+
|     1 | deepika | 27899  |      1 |      1 | software  |
|     4 | vasuu   | 78000  |      2 |      2 | hardware  |
|     2 | sai     | 56576  |      2 |      2 | hardware  |
|     6 | dinesh  | NULL   |      3 |      3 | developer |
|     5 | gnapika | 54565  |      3 |      3 | developer |
|     3 | ganesh  | 60000  |      3 |      3 | developer |
|  NULL | NULL    | NULL   |   NULL |      4 | tester    |
+-------+---------+--------+--------+--------+-----------
```

c)full outer join;
√ Combines the results of both left and right outer joins.
√ Non-matching records from both the tables will be left blank
Syntax:select * from table_name1 full outer join table_name2 on condition;

Example:
mysql>select * from employee full outer join dept on employee.deptno=dept.deptno;
Output:
```
+-------+---------+--------+--------+--------+-----------+
| empid | empname | empsal | deptno | deptno | deptname  |
+-------+---------+--------+--------+--------+-----------+
|     1 | deepika | 27899  |      1 |      1 | software  |
|     2 | sai     | 56576  |      2 |      2 | hardware  |
|     3 | ganesh  | 60000  |      3 |      3 | developer |
|     4 | vasuu   | 78000  |      2 |      2 | hardware  |
|     5 | gnapika | 54565  |      3 |      3 | developer |
|     6 | dinesh  | NULL   |      3 |      3 | developer |
|     7 | ramesh  | 45000  |      7 |   NULL | NULL      |
| NULL|NULL|NULL|   NULL|    4 |tester   |
---------------------------------------------------------
```

5)self join:

√ Here the table is joined (compared) to itself.
√ The output shows the low salary employee names, high salary employee names from the employee table with itself.
Syntax:select s1.column_name,s2.column_name from table1s1,table2s2 where s1.common_col_name=s2.common_col_name;
Example:
mysql> select a.empname low_salary_emp,b.empname high_salary_emp from employee a,employee b where a.empsal<b.empsal;
Output:
```
+----------------+-----------------+
| low_salary_emp | high_salary_emp |
+----------------+-----------------+
| ramesh         | sai             |
| gnapika        | sai             |
| deepika        | sai             |
| ramesh         | ganesh          |
| gnapika        | ganesh          |
| sai            | ganesh          |
| deepika        | ganesh          |
| ramesh         | vasuu           |
| gnapika        | vasuu           |
| ganesh         | vasuu           |
| sai            | vasuu           |
| deepika        | vasuu           |
| ramesh         | gnapika         |
| deepika        | gnapika         |
| deepika        | ramesh          |
+----------------+-----------------
```

PROGRAM NO-8

8.FUNCTIONS

AIM: To implement queries based on various functions in a database.

DESCRIPTION:
➔ All SQL functions are inbuilt functions.
➔ These are classified as two types:
1. Single row function
2. Multiple row function

1.Single row functions:

There are the one who works on the single row and return one output for row.
EX: Conversion Function,Character Function (or) String Function,Numeric Function.

i.conversion function:

a)upper(): This function convert a string to Uppercase.
Syntax: select upper(string);
Example:
mysql> select upper("deepika");
Output:

```
+------------------+
| upper("deepika") |
+------------------+
| DEEPIKA          |
+------------------+
```

b)lower(): This function converts the string to all lowercase.
Syntax:select lower(string);
Example:
mysql> select lower("DEEPIKA");
Output:

```
+------------------+
| lower("DEEPIKA") |
+------------------+
| deepika          |
+------------------
```

ii.string functions:

a)concat(): This function is used to combine two strings.
Syntax:select concat(“string1”,”string2”);

Example:

mysql> select concat("deepika","jesta");

Output:

```
+---------------------------+
| concat("deepika","jesta") |
+---------------------------+
| deepikajesta              |
+---------------------------
```

b)strcmp(): This function is used to compare two strings.

Syntax:select strcmp(string1,string2);

Example:

mysql> select strcmp("deepu","deepika");

Output:

```
+---------------------------+
| strcmp("deepu","deepika") |
+---------------------------+
|                         1 |
+---------------------------
```

c)length(): This function is used count the length of the string.

Syntax:select length(“string”);

Example:

mysql> select length("deepika");

Output:

```
+-------------------+
| length("deepika") |
+-------------------+
|                 7 |
+-------------------+
```

d)substr(): This function is used to return a porttion of string given start point to end point.

Syntax:select substr(string,startpoint);

Example:

mysql> select substr("deepika",5);

Output:

```
+---------------------+
| substr("deepika",5) |
+---------------------+
| ika                 |
-------------------
```

e)instr(): This function is used to return a numeric position of a character (or) string.

Syntax:select instr(string,character);
Example:
mysql> select instr("rgukt","u");
Output:
```
+---------------------+
| instr("rgukt","u") |
+---------------------+
|                  3 |
---------------------
```

f)lpad(): This function is used to insert the symbol with the actual length of the string from left to right.
Syntax:select lpad(string,length,symbol);
Example:
mysql> select lpad("rgukt",10,"*");
Output:
```
+----------------------+
| lpad("rgukt",10,"*") |
+----------------------+
| *****rgukt           |
+----------------------
```

g)rpad(): This function is used to insert the symbol with the actual length of the string from right to left.
Syntax:select rpad(string,length,symbol);
Example:
mysql> select rpad("rgukt",10,"*");
Output:
```
+----------------------+
| rpad("rgukt",10,"*") |
+----------------------+
| rgukt*****           |
-------------------------
```

h)ltrim(): This function is used to remove leading spaces in a given string from left side.
Syntax:select ltrim(“   string”);
Example:
mysql> select ltrim("world   ");
Output:
```
+-----------------------+
| ltrim("world       ") |
+-----------------------+
| world                 |
+-----------------------
```

iii.numeric functions

1.Truncate(): Remove decimal part.
Syntax:select trunc(number having decimalpart);
Example:
mysql> select round (27.6);
Output:

```
+--------------+
| round (27.6) |
+--------------+
|           28 |
+--------------+
```

2.mod(): This used to find the remainder of a division.
Synatx:select mod(dividend,divisor);
Example:
mysql> select mod(56,6);
Output:

```
+-----------+
| mod(56,6) |
+-----------+
|         2 |
+-----------+
```

3.least():used to find the least value in a given set of values.
Syntax:select least(set of numbers);
Example:
mysql> select least(-2,-6,-1,0);
Output:

```
+-------------------+
| least(-2,-6,-1,0) |
+-------------------+
|                -6 |
+-------------------+
```

4.greatest():used to find the greatest value in a given set of values.
Syntax:select greatest(set of numbers);
Example:
mysql> select greatest(-2,-6,-1,0);
Output:

```
+----------------------+
| greatest(-2,-6,-1,0) |
+----------------------+
|                    0 |
+----------------------+
```

5.sqrt(): It is used to find the square root of a number.
Syntax:select sqrt(number);
Example:
mysql> select sqrt(64);
Output:
```
+----------+
| sqrt(64) |
+----------+
|        8 |
+----------
```

6.ceil(): It is used to find the upperbound of a value.
Syntax:select ceil(decimal number);
Example:
mysql> select ceil(25.3);
Output:
```
+------------+
| ceil(25.3) |
+------------+
|         26 |
--------------
```

7.floor():it is used to find the lowerbound of a value.
Syntax:select floor(decimal number);
Example:
mysql> select floor(25.3);
Output:
```
+-------------+
| floor(25.3) |
+-------------+
|          25 |
-----------------
```

8.power(): It is used to find the power of particular value.
Syntax:select power(number,powervalue);
Example:
mysql> select power(8,2);
Output:
```
+------------+
| power(8,2) |
+------------+
|         64 |
+------------
```

2)multiple row functions:

➢ These are works upon group of rows and return one result for the complete set of rows.
➢ These are also called as "group function" (or) "aggregate fuctions".
➢ The following are the aggregate functions:

i)sum()
ii)avg()
iii)count()
iv)min()
v)max()
vi)first()
vii)last()

a)sum(): The function is used to get sum of numeric column.
Syntax:select sum(column_name) from table_name;
Example:

```
mysql> select sum(empsal) from employee;
```

Output:

```
+-------------+
| sum(empsal) |
+-------------+
|      322040 |
+-------------+
```

b)avg(): This function is used to get the average of numeric column.
Syntax:select avg(column_name) from table_name;
Example:

```
mysql> select avg(empsal) from employee;
```

Output:

```
+-------------------+
| avg(empsal)       |
+-------------------+
| 53673.333333333336 |
-------------------------
```

c)count(): This function is used to get the number of rows in a table.

Syntax:select count(*) from table_name;

Example:

```
mysql> select count(*) from employee;
```

Output:

```
+----------+
| count(*) |
+----------+
|        7 |
+----------
```

d)min(): This function is used to get the minimum value from a column.
Syntax:select min(column_name)from table_name;
Example:
mysql> select min(empsal) from employee;
Output:
```
+-------------+
| min(empsal) |
+-------------+
| 27899       |
+-------------+
```

e)max(): This function is used to get the maximum value from a column.
Syntax:select max(column_name)from table_name;
Example:
mysql> select max(empsal) from employee;
Output:
```
+-------------+
| max(empsal) |
+-------------+
| 78000       |
+-------------
```
f)first(): This function is used to get the first value of selected column.

Syntax:select column_name from table_name limit1;
Example:
mysql> select empname from employee limit 1;

Output:
```
+---------+
| empname |
+---------+
| deepika |
+---------+
|       2 |
+----------
```
g)last(): This function is used to get the lastvalue of a selected column.

Syntax:select column_name from table_name order by column_name desc limit1;

Example:
mysql> select empname from employee order by empname desc limit 1;
Output:
```
+---------+
| empname |
+---------+
| vasuu   |
+---------
```

PROGRAM NO-9

9.VIEWS

AIM: To implement Queries of creating,altering,inserting and deleting the views.

DESCRIPTION:

MySQL Views are the database objects that can be created on a table to improve the performance of MySQL Server. A view has unique Constraints look on data from one or more tables. It can organize data in unique order, focus or hide some data.
MySQL Views comprises of a stored query accessible as a fundamental table composed of the outcome set.
Beside standard tables a perspective does not shaped a part of the physical schema. It is a virtual table, dynamic in the database.
View is a stored query, and it can be attributed like a table.

MySQL Views object is mainly classified into:
1)create a view
2)inserting into view
3)delete from view
4)Update the view
5)alter a view
6)truncate view
7)drop a view

1)create a view:

Syntax for creat view:

create view view_name as select column 1 , column2,...... from table name where condition;

Example:

mysql> create view deptview as select * from employee where deptno=3;
mysql> select * from deptview;

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | NULL   |      3 |
+-------+---------+--------+--------
```

Insert rows in a view:

Syntax :
Insert into view_name(column 1,column 2) values value1(name),value2(age);

Example:
mysql>insert into deptview(empid,empname,empsal,deptno) values (7,"sruthi",89000,3);

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | NULL   |      3 |
|      7| sruthi  |  89000 |       3|
+-------+---------+--------+--------
```

3)Delete from view:

Syntax : Delete from view_name where condition;
Example:
mysql>delete from deptview where empsal=89000;
mysql>select * from deptview;
Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | NULL   |      3 |
+-------+---------+--------+--------
```

4)update the view:

Syntax:update view_name set columnname= "value"where condition;
Example:
mysql>update deptview set empsal=50000 where empid=6;
mysql>select * from view;
Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     3 | ganesh  | 60000  |      3 |
|     5 | gnapika | 54565  |      3 |
|     6 | dinesh  | 50000  |      3 |
+-------+---------+--------+--------
```

5)alter a view:

Syntax:alter view viewname as select * from tablename where condition;

Example:

```
mysql> alter view deptview as select * from employee where deptno=2;
mysql> select * from deptview;
```

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|     2 | sai     | 56576  |      2 |
|     4 | vasuu   | 78000  |      2 |
+-------+---------+--------+--------
```

6)truncate view:

Syntax: truncate view view_name;

Example:

```
mysql>truncate view deptview;
mysql>select * from deptview;
```

Output:

```
+-------+---------+--------+--------+
| empid | empname | empsal | deptno |
+-------+---------+--------+--------+
|   -   |    -    |  -     |      - |
|     - |    -      |   -    |      - |
+-------+---------+--------+--------
```

7)drop a view:

Syntax:drop view view_name;

Example:

```
mysql> drop view deptview;
mysql> select * from deptview;
```

Output:

```
ERROR 1146 (42S02): Table 'deepika.deptview' doesn't exist
```

PROGRAM NO-10

10.SUB QUERIES

AIM: To implement SQL queries using Sub queries.

DESCRIPTION:

A sub query is a query inside another query. A sub query or inner query is used to generate the required information used as input for outer query. A sub query has characteristics. · A sub query is normally expressed inside parenthesis · The inner query or sub query is executed first.

Syntax for nested query:

select column_list from table_name where column_name operator (select column_name from table_name where condition);

1)where sub queries:

The most common type of sub query used on inner select query on the right side of a where clause comparison expression.

Example:

mysql> select empname,empsal from employee where empsal>(select avg(empsal)from employee);

Output:

```
+---------+--------+
| empname | empsal |
+---------+--------+
| sai     | 56576  |
| ganesh  | 60000  |
| vasuu   | 78000  |
| gnapika | 54565  |
+---------+--------
```

2)In sub queries:

When you want to compare single attributewith the list of value we can use in sub queries inside.

Example:

mysql> select empname,empsal from employee where empsal in (select max(empsal)from employee group by deptno);

Output:

```
+---------+--------+
| empname | empsal |
+---------+--------+
| deepika | 27899  |

| ganesh  | 60000  |
| vasuu   | 78000  |
| ramesh  | 45000  |
+---------+--------
```

3)Having subqueries:

An sub queries used in the where clause, we can use a sub queries with a havingclause, Having clause is used to restrict the rows by using the condition in the group by query.

Example:

```
mysql> select deptno from employee group by deptno having avg(empsal)>(select
avg(empsal) from employee);
```

Output:

```
+--------+
| deptno |
+--------+
|      2 |
|      3 |
+--------
```

4)From subqueries:

As you already know, in the from clause, this specifies table name from which the data can be drawn; because the output of the select statement is another table.

Example:

```
mysql> select empname,empsal from (select *from employee)alias;
```

Output:

```
+---------+--------+
| empname | empsal |
+---------+--------+
| deepika | 27899  |
| sai     | 56576  |
| ganesh  | 60000  |
| vasuu   | 78000  |
| gnapika | 54565  |
| dinesh  | NULL   |
| ramesh  | 45000  |
+---------+--------
```

****************************THE END*********************************

DATABASES

AIM: To create a database of Hotel management system with 7 entities

DESCRIPTION:

Creating a database with related tables in it using SQL commands and joining all the tables using join operations.

CREATING DATABASE:

```
sql>create database Hotel;
sql>use Hotel;
```

CREATION OF TABLES:

Login Table:

```
sql>create table Login(Loginid integer primary key,roleid integer,username
varchar(30),password varchar(10),foreign key(roleid)references        Roles(role_id));
```

Roles Table:

```
sql>create table Roles(role_id integer primary key,role_name
create table Roles(role_id integer primary key,role_name
char(20),role_desc varchar(30));
```

User Table:

```
sql>create table User(user_id integer primary key,username varchar(20),email
varchar(30),password varchar(10),address varchar(30));
```

Customer Table:

```
sql>create table Customer(cus_id integer primary key,cus_name
char(20),cus_pass integer,Hotel_id integer,cus_mobile integer,foreign
key(Hotel_id)references hotel(Hotel_id));
```

hotel Table:

```
sql>create table hotel(Hotel_id integer primary key,Hotel_name
char(20),Hotel_type char(20),Hotel_rent varchar(30));
```

Booking Table:

```
sql>create table Booking(book_id integer primary key,book_type
char(15),book_desc char(20),Hotel_id integer,foreign key(Hotel_id)
references hotel(Hotel_id));
```

Payments Table:

sql>create table Payments(pay_id varchar(20) primary key,pay_date varchar(20),pay_amt integer,pay_mode char(15),cus_id integer,foreign key(cus_id) references Customer(cus_id));

Inserting Values Into Tables:

Login Table:

sql>insert into Login values(1234,7890,sahara,sahara@14), (1235,7891,javid,javid@13),(1236,7892,mona,mona@15),(1237,7893,hima,hima@45),(1238,7894,balu,balu@67);

Loginid	roleid	username	password
1234	7890	sahara	sahara@14
1235	7891	javid	javid@13
1236	7892	mona	mona@15
1237	7893	hima	hima@45
1238	7894	balu	balu@67

Roles Table:

sql>insert into Roles values(7890,’Manager’,’supervising staff’),
(7891,’Receptionist’,’respond to all guests questions’),
(7892,’AsstManager’,’maintain staff descpline’),(7893,’Housekeeper’,’clean and tidy up rooms’),(7894,’Kitchenstaff’,’cooking for the guests’);

role_id	role_name	role_desc
7890	Manager	supervising staff
7891	Receptionist	respond to all guests
7892	AsstManager	maintain staff escipline
7893	Housekeeper	clean and tidy up rooms
7894	KItchenstaff	cooking for the guests

User Table:

sql>insert into User values(202,’sahara’,’sahara@gmail.com’,’sahara@14’,’ongole’),
(203,’javid’,’javid@gmail.com’,’javid@13’,’hyderabad’),
(204,’mona’,’mona@gmail.com’,’mona@15’,’vizag’),
(205,’hima’,’hima@gmail.com’,’hima@45’,’vijaywada’),
(206,’balu’,’balu@gmail.com’,’balu@67’,’kakinada’);

user_id	**username**	**gmail**	**password**	**address**
202	sahara	sahara@gmail.com	sahara@14	ongole
203	javid	javid@gmail.com	javid@13	hyderabad
204	mona	mona@gmail.com	mona@45	vizag
205	hima	hima@gmail.com	hima@45	vijaywada
206	balu	balu@gmail.com	balu@67	kakinada

Customer Table:

sql>insert into Customer values(956,’karan’,4444,’1234,987654321),
(957,’raj’,8888,1235,987654322),(958,’vardhan’,1111,1236,987654323),
(959,’mahi’,6666,1237,987654324),(960,’janu’,9999,1238,987654325);

cus_id	**cus_name**	**cus_pass**	**Hotel_id**	**cus_mobile**
956	karan	4444	1234	987654321
957	raj	8888	1235	987654322
958	vardhan	1111	1236	987654323
959	mahi	6666	1237	987654324
960	janu	9999	1238	987654325

hotel Table:

sql>insert into hotel values(1234,’Vihara’,’Resort’,100000),(1235,’Novatel’,’Chainhotel’,50000),(1236,’Vasista’,’Motel’,75000),(1237,’Prakruthi’,’Boutique’,45000),(1238,’Anand’,’Allsuits’,40000);

Hotel_id	Hotel_name	Hotel_type	Hotel_rent
1234	Vihara	Resort	100000
1235	Novatel	Chainhotel	50000
1236	Vasista	Motel	75000
1237	Prakruthi	Boutique	45000
1238	Anand	Allsuits	40000

Booking Table:

sql>insert into Booking values(275,’Direct’,’instant booking’,1234),(276,’Indirect’,’online booking’,1235),(277,’Direct’,’guranteed booking’,1236),(278,’Indirect’,’instant booking’,1237),(279,’Direct’,’guranteed booking’,1238);

book_id	book_type	book_desc	Hotel_id
275	Direct	Instant booking	1234
276	Indirect	Online booking	1235
277	Direct	Guranteed booking	1236
278	Indirect	Instant booking	1237
279	Direct	Guranteed booking	1238

Payments Table:

sql>insert into Payments values(‘P111’,’14-02-2023’,45000,’online’,956), (‘P112’,’10-01-2023’,50000,’BHIM’,957),(‘P113’,’27-01- 2023’,40000,’UPI’,958),(‘P114’,’15-02-2023’,100000,’Handcash’,959),
(‘P115’,’06-03-2023’,55000,’;Cards’,960);

pay_id	pay_date	Pay_amt	pay_mode	cus_id
P111	14-02-2023	45000	online	956
P112	10-01-2023	50000	BHIM	957
P113	27-01-2023	40000	UPI	958
P114	15-02-2023	100000	Handcash	959
P115	06-03-2023	55000	Cards	960

APPLYING JOINS:

```
sql>select hotel.Hotel_name,Booking.book_type from hotel  join Booking on
        hotel.Hotel_id=Booking.Hotel_id;
```

Hotel_name	book_type
Vihara	Direct
Novatel	Indirect
Vasista	Direct
Prakruthi	Indirect
Anand	Direct

RESULT:

Creation of Hotel management database created successfully.

AIM: To Create database of Bank Management System with 10 Related Tables.

Program:

```
sql>create database bank;
sql>use bank;

sql>create table bank_det(
        bank_name char(20) primary key,
        location char(30),
        no_of_account_holders int,
        turnoverperyear int,
        manager char(10),
        no_of_branches int
        );

sql> create table account(
account_id int(20) primary key,
customername char(20),
created_date varchar(10),
bankname    char(10),
account_balance int,
foreign key(bankname) references bank_det(bank_name)
);

sql>create table loan(
        loan_id int(20) primary key,
        accountid int(20),
        granted_date varchar(20),
        granted_amount int,
        duration_of_loan varchar(20),
        monthly_payment int,
        interest int,foreign key(accountid) references account(account_id)
        );
```

```
sql>create table customer_det(
        customer_name char(20) primary key,
        acc_id int(20),
        customerid int,
        customer_phno varchar(20),
        customer_age int,
        cust_add varchar(30),
        customerdob varchar(20),
        gender char(20),
        foreign key(acc_id) references account(account_id)
        );

sql>create table manager_det(
        manager_name char(20) primary key,
        bkname char(20),
        email varchar(20),
        salary int,
        contact varchar(20),
        address varchar(20),
        foreign key(bkname) references bank_det(bank_name)
        );

sql>create table branches(
        branch_name char(20)primary key,
        banknm char(20),
        branchid int,
        branch_city char(10),
        manager char(10),
        no_of_cust int,
        no_of_emp int,
        turnover int,
        foreign key(banknm)references bank_det(bank_name),
        foreign key(manager)references manager_det(manager_name)
        );
```

```
sql>create table transaction(
        trans_id varchar(20) primary key,
        acc_no int(20),cust_name char(20),
        ondate varchar(20),
        credit_or_debit char(10),
        amount int,balance int,
        payment_type char(20),
        foreign key(acc_no)references account(account_id),
        foreign key(cust_name) references customer_det(customer_name)
        );

sql>create table atm(
        atm_branch char(20),
        atm_no int primary key,
        card_holdername char(20),
        card_balance int,
        date_of_issue varchar(20) ,
        location varchar(20),
        foreign key(atm_branch) references branches(branch_name)
        );

sql>create table bank_emp_det(
        emp_id int,
        empname char(20) primary key,
        emp_pos char(20),bank char(20),
        empsal int, emp_phno varchar(20),
        emp_add varchar(20),
        empage int,gender char(20),
        foreign key(bank) references bank_det(bank_name)
        );
```

```
sql>create table services(
        servicename char(20)primary key,
        corresp_Emp char(20),no_custo int,bname char(20),
        branname char(20),managername char(20),
        foreign key(branname) references branches(branch_name));
```

Inserting Data into Tables:

```
sql>insert into bank_det values('SBI','HYD',30,100000,'ganesh',10),
('HDFC','BANGLORE',100,200000,'ajay',15),
('ICICI','MUMBAI',150,300000,'vikram',18),
('UNIONBANKOFINDIA','VIZAG',50,50000,'krunal',8),
('APGB','VIJAYWADA',75,250000,'rahul',11),
('PUNJABNATIONAL','PUNJAB',120,3000000,'vishal',35),
('CANARABANK','ONGOLE',100,450000,'githa',10),
('KARURVYSYA','TANGUTUR',50,800000,'sriram’,7),
('AXISBANK','CHENNAI',100,750000,'krishna’,15),
('INDIANBANK','KOLKATTA',200,500000,'pragwan',20);
```

bank_name	**location**	**no_of_account_holders**	**turnoverperyear**	**manager**	**no_of_branches**
SBI	HYD	30	100000	ganesh	10
HDFC	BANGLORE	100	200000	ajay	15
ICICI	MUMBAI	150	300000	vikram	18
UNIONBANKOFINDIA	VIZAG	50	500000	krunal	8
APGB	VIJAYWADA	75	250000	rahul	11
PUNJABNATIONAL	PUNJAB	120	3000000	vishal	35
CANARABANK	ONGOLE	100	450000	githa	10
KARURVYSYA	TANGUTUR	50	800000	sriram	7
AXISBANK	CHENNAI	100	750000	krishna	15
INDIANBANK	KOLKATTA	200	500000	pragwan	20

```
sql>insert into account values(321654981,'sai','02-01-2021','SBI',15000),
                (321654980,'ravi','30-05-2022','HDFC',3000),
                (321654979,'vasu','03-03-2020','KARURVYSYA',2000),
        (321654978,'ganesh','05-10-2022','CANARABANK',30000),
        (321654982,'srinu','30-12-2022','INDIANBANK',40000),
        (321654983,'laxman','15-01-2022','AXISBANK',30000),
        (321654984,'Pavan','11-03-2020','APGB',5000),
                (321654985,'Arjun','03-12-2022','ICICI',13000),
                (321654986,’Bharath’,’20-10-2021’,’HDFC’,15000),
                (321654987,’Suresh’,’23-09-2022’,’SBI’,10000);
```

account_id	**customername**	**created_date**	**bankname**	**account_balance**
321654981	sai	02-01-2021	SBI	15000
321654980	ravi	30-05-2022	HDEC	3000
321654979	vasu	03-03-2020	KARURVYSYA	2000
321654978	ganesh	05-10-2022	CANARABANK	30000
321654982	srinu	30-12-2022	INDIANBANK	40000
321654983	laxman	15-01-2022	AXISBANK	30000
321654984	Pavan	11-03-2020	APGB	5000
321654985	Arjun	03-12-2022	ICICI	13000
321654986	Bharath	20-10-2021	HDFC	15000
321654987	Suresh	23-09-2022	SBI	10000

sql>insert into loan values(789456123,321654987,'09-10-2022',50000,'2years',2000,2),(789456122,321654986,'02-03-2022',30000,'1year',1500,2),(789456121,321654985,'02-01-2023',70000,'3years',4000,1),(789456120,321654984,'09-11-2021',35000,'1year',3500,2), (789456119,321654983,'02-03-2022',15000,'1year',500,2),(789456118,321654982,'03-01-2023',20000,'2years',1000,2),(789456117,321654981,'29-10-2021',50000,'2years',3500,2), (789456116,321654980,'02-06-2022',80000,'3years',2500,3),(789456115,321654979,'02-06-2022',70000,'3years',4000,1),(789456114,321654978,'09-10-2022',50000,'2years',3500,2);

loan_id	**accountid**	**granted_date**	**granted_amount**	**duration_of_loan**	**monthly_payment**	**interest**
789456123	321654987	09-10-2022	50000	2years	2000	2
789456122	321654986	02-03-2022	30000	1year	1500	2
789456121	321654985	02-01-2023	70000	3years	4000	1
789456120	321654984	09-11-2021	35000	1year	3500	2
789456119	321654983	02-03-2022	15000	1year	500	2
789456118	321654982	03-01-2023	20000	2years	1000	2
789456117	321654981	29-10-2021	50000	2years	3500	2
789456116	321654980	02-06-2022	80000	3years	2500	3
789456115	321654979	02-06-2022	70000	3years	4000	1
789456114	321654978	09-10-2022	50000	2years	3500	2

sql>insert into customer_det values
('ravi',321654980,12345,'9876543210',25,'prakasamdt','23-09-1998','M'),
('vasu',321654979,12346,'9876543211',23,'krishnadt','05-01-2000','M'),
('sai',321654981,12347,'9876543212',19,'nelloredt','17-04-2003','M'),(
('srinu',321654982,12348,'9876543213',24,'kadapadt','09-05-1999','M'),
('laxman',321654983,12349,'9876543214',26,'chittoordt','08-10-1997','M'),
('suresh',321654987,12350,'9876543215',23,'westgodavaridt','07-12- 2000','M'),
('bharath',321654986,12351,'9876543216',28,'eastgodavaridt','11-02- 1995','M'),
('arjun',321654985,12352,'9876543217',20,'vizagdt','29-09-2003','M'),
('Pavan',321654984,12353,'9876543218',22,'srikakulamdt','03-08-2001','M'),
('ganesh',321654978,12354,'9876543219',23,'kurnooldt','08-10-2000','M');

customer_name	acc_id	customerid	customer_phno	customer_age	cust_add	customerdob	gender
ravi	321654980	12345	9876543210	25	prakasamdt	23-09-1998	M
vasu	321654979	12346	9876543211	23	krishnadt	05-01-2000	M
sai	321654981	12347	9876543212	19	nelloredt	17-04-2003	M
srinu	321654982	12348	9876543213	24	kadapadt	09-05-1999	M
laxman	321654983	12349	9876543214	26	chittordt	08-10-1997	M
suresh	321654987	12350	9876543215	23	westgodavaridt	07-12-2000	M
bharath	321654986	12351	9876543216	28	eastgodavaridt	11-02-1995	M
arjun	321654985	12352	9876543217	20	vizagdt	29-09-2003	M
Pavan	321654986	12353	9876543218	22	srikakulamdt	03-08-2001	M
ganesh	321654978	12354	9876543219	23	kurnooldt	08-10-2000	M

sql>insert into manager_det values('krishna','AXISBANK','mn@axisbk.gmail.com',125000,'6543219870', 'chennai'),
('githa','CANARABANK','mn@canbk.gmail.com',100000,'6543219871','ongole'),
('ajay','HDFC','mn@hdfcbk.gmail.com',150000,'6543219872','banglore'),
('pragwan','INDIANBANK','mn@indibk.gmail.com',50000,'6543219873','kolkatta'),
('vikram','ICICI','mn@icibk.gmail.com',75000,'6543219874','mumbai'),
('rahul','APGB','mn@apgbbk.gmail.com',115000,'6543219875','vijayawada'),
('sriram','KARURVYSYA','mn@kvbk.gmail.com',125000,'6543219876','tangutur'),
('krunal','UNIONBANKOFINDIA','mn@ubibk.gmail.com',85000,'6543219877', 'vizag'),
('ganesh','SBI','mn@sbibk.gmail.com',200000,'6543219878','hyd'),
('vishal','PUNJABNATIONAL','mn@pnbk.gmail.com',120000,'6543219879','punjab');

manager_name	bkname	email	salary	contact	address
krishna	AXISBANK	mn@axisbk.gmail.com	125000	6543219870	chennai
githa	CANARABANK	mn@canbk.gmail.com	100000	6543219871	ongole
ajay	HDFC	mn@hdfcbk.gmail.com	150000	6543219872	banglore
pragwan	INDIANBANK	mn@indibk.gmail.com	50000	6543219873	kolkatta
vikram	ICICI	mn@icibk.gmail.com	75000	6543219874	mumbai
rahul	APGB	mn@apgbbk.gmail.com	115000	6543219875	vijayawada
sriram	KARURVYSYA	mn@kvbk.gmail.com	125000	6543219876	tangutur

sql>insert into branches values
('ponnalursbi','SBI',14163,'ponnalur','ganesh',10,5,60000),
('madhapurhdfc','HDFC',14164,'vizag','ajay',20,6,65000),
('kondpicanara','CANARABANK',14165,'ongole','githa',11,4,70000),
('kanigiriaxis','AXISBANK',14166,'kanigiri','krishna',15,7,100000),
('kmpliUBI','UNIONBANKOFINDIA',14167,'kmpli','krunal',12,6,100000),
('chlkluripetaindibk','INDIANBANK',14168,'chilkpeta','pragwan',15,6,35000)
('kmpliapgb','APGB',14169,'kmpli','rahul',14,6,70000),(
'tanguturkvnk','KARURVYSYA',14170,'tangutur','sriram',13,7,45000),
('skonda','ICICI',14171,'skonda','vikram',10,5,60000),
('gandinagarpnb','PUNJABNATIONAL',14172,'gujarath','vishal',16,6,19000);

branch_name	**banknm**	**branchid**	**branch_city**	**manager**	**no_of_cust**	**no_of_emp**	**turnover**
ponnalursbi	SBI	14163	ponnalur	ganesh	10	5	60000
madhapurhdfc	HDFC	14164	vizag	ajay	20	6	65000
kondpicanara	CANARABANK	14165	ongole	githa	11	4	70000
kanigiriaxis	AXISBANK	14166	kanigiri	krishna	15	7	1000000
kmpliUBI	UNIONBANKOFINDIA	14167	kmpli	krunal	12	6	1000000
chlkluripetaindibk	INDIANBANK	14168	chilkpeta	pragwan	15	6	35000
kmpliapgb	APGB	14169	kmpli	rahul	14	6	70000
tanguturvnk	KARURVYSYA	14170	tangutur	sriram	13	7	45000
skonda	ICICI	14171	skonda	vikram	10	5	60000
gandinagarpnb	PUNJABNATIONAL	14172	gujarath	vishal	16	6	19000

```
sql>insert into
transaction(trans_id,acc_no,cust_name,ondate,credit_or_debit,
amount,balance,payment_type)values
('TNX54321',321654982,'srinu','07-01-2023','credit',10000,50000,'cash'),(
'TNX54322',321654981,'sai','07-11-2022','debit',12000,3000,'cash'),
('TNX54323',321654980,'ravi','18-09-2022','credit',15000,18000,'cheque'),
('TNX54324',321654978,'ganesh','09-11-2022','credit',8000,38000,'cash'),
('TNX54325',321654983,'laxman','16-07-
2022','debit',10000,20000,'mobilepayment'),('TNX54326',321654986,'Bharath','30-12-
2022','debit',7000,8000,'moneyorder'),
('TNX54327',321654985,'Arjun','09-01-2023','debit',3000,10000,'debitcard'),
('TNX54328',321654979,'vasu','19-05-2022','credit',3000,5000,'cash'),
('TNX54329',321654987,'Suresh','07-10-2022','credit',10000,20000,'cash'),
('TNX54330',321654984,'pavan','27-10-2022','credit',10000,15000,'cash');
```

trans_id	**acc_no**	**cust_name**	**ondate**	**credit_or_debit**	**amount**	**balance**	**payment_type**
TNX54321	321654982	srinu	07-01-2023	credit	10000	50000	cash
TNX54322	321654981	sai	07-11-2022	debit	12000	3000	cash
TNX54323	321654980	ravi	18-09-2022	credit	15000	18000	cheque
TNX54324	321654978	ganesh	09-11-2022	credit	8000	38000	cash
TNX54325	321654983	laxman	16-07-2022	debit	10000	20000	mobilepayment
TNX54326	321654986	Bharath	30-12-2022	debit	7000	8000	moneyorder
TNX54327	321654985	Arjun	09-01-2023	debit	3000	10000	debitcard
TNX54328	321654979	vasu	19-05-2022	credit	3000	5000	cash
TNX54329	321654987	Suresh	07-10-2022	credit	10000	20000	cash
TNX54330	321654984	pavan	27-10-2022	credit	10000	15000	cash

```
sql>insert into atm values
('skonda',7654321,'vasu',5000,'09-04-2021','skonda' ),
('madhapurhdfc',7654322,'ravi',18000,'19-09-2022','vizag'),
('ponnalursbi',7654323,'sai',3000,'09-04-2021','ponnalur'),
('chlkpuripetaindibk',7654324,'srinu',50000,'08-01-2023','chilkpeta'),
('kondpicanara',7654325,'ganesh',38000,'09-11-2022','ongole'),
('kmpliapgb',7654326,'Pavan',15000,'09-04-2021','kmpli'),
('ponnalursbi',7654327,'suresh',20000,'09-10-2022','ponnalur'),
('kanigiriaxis',7654328,'laxman',20000,'29-04-2022','kanigiri'),
('madhapurhdfc',7654329,'Bharath',8000,'09-11-2021','vizag'),
('skonda',7654330,'Arjun',10000,'31-12-2022','skonda');
```

atm_branch	**atm_no**	**card_holdername**	**card_balance**	**date_of_issue**	**location**
skonda	7654321	vasu	5000	09-04-2021	skonda
madhapurhdfc	7654322	ravi	18000	19-09-2022	vizag
ponnalursbi	7654323	sai	3000	09-04-2021	ponnalur
chlkpuripetaindibk	7654324	srinu	50000	08-01-2023	chilkpeta
kondpicanara	7654325	ganesh	38000	09-11-2022	ongole
kmpliapgb	7654326	Pavan	15000	09-04-2021	kmpli
ponnalursbi	7654327	suresh	20000	09-10-2022	ponnalur
kanigiriaxis	7654328	laxman	20000	29-04-2022	kanigiri
madhapurhdfc	7654329	Bharath	8000	09-11-2021	vizag
skonda	7654330	Arjun	10000	31-12-2022	skonda

```
sql>insert into bank_emp_det values
(1,'ganesh','bkmanager','SBI',125000,'7658962173','vizayanagaram',35,'M'),
(2,'Anitha','asstmanager','HDFC',75000,'6300416456','prakasam',26,'F'),
(3,'ramesh','cashier','SBI',50000,'9392573228','pksm',45,'M'),
(4,'sunitha','accountant','ICICI',65000,'7729941686','mopadu',40,'F'),
(5,'sai','empgold','AXISBANK',75000,'8897168680','guntur',25,'M'),
(6,'jayalakshmi','loanemp','CANARABANK',45000,'7658962714','vzgm',42,'F
(7,'ramaswamy','asstmanager','PUNJABNATIONAL',125000,'7658962175',
'vzgm',45,'M'),(8,'vasavi','cashier','HDFC',65000,'7658962176','vzgm',27,'F'),
(9,'Jyothi','goldemp','INDIANBANK',75000,'9177567526','hyd',24,'F'),
(10,'dinesh','loanemp','KARURVYSYA',55000,'9640004739','banglore',25,'M');
```

emp_id	empname	emp_pos	bank	empsal	emp_phno	emp_add	empage	gender
1	ganesh	bkmanager	SBI	125000	7658962173	vizayanagaram	35	M
2	Anitha	asstmanager	HDFC	75000	6300416456	prakasam	26	F
3	ramesh	cashier	SBI	50000	9392573228	pksm	45	M
4	sunitha	accoutant	ICICI	65000	7729941686	mopadu	40	F
5	sai	empgold	AXISBANK	75000	8897168680	guntur	25	M
6	jayalakshmi	loanemp	CANARABANK	45000	7658962714	vzgm	42	F
7	ramaswamy	asstmanager	PUNJABNATIONAL	125000	7658962175	vzgm	45	M
8	vasavi	cashier	HDFC	65000	7658962176	vzgm	27	F
9	Jyothi	goldemp	INDIANBANK	75000	9177567526	hyd	24	F
10	dinesh	loanemp	KARURVYSYA	55000	9640004739	banglore	25	M

```
sql>insert into services values
('goldloan’,'Nivas',5,'AXISBANK','kanigiriaxis','krishna'),
('irrigationloan','Kushal',4,'ICICI','skonda','vikram'),
('educationloan’,'Teja',3,'CANARABANK','kondpicanara','githa'),
('mobilebanking','Jaanu',5,'HDFC','madhapurhdfc','ajay'),
('creditdebitcards','vishwa',8,'APGB','kmpliapgb','rahul'),
('accountopening','datha',10,'UNIONBANKOFINIDA','kmpliUBI','krunal'),
('atm','manohar',7,'KARURVYSYA','tanguturkvnk','sriram'),
('savings','sreevidya',9,'PUNJABNATIONAL','gandinagarpnb','vishal'),
('cheque','Sravani',6,'INDIANBANK','chlkpuripetaindibk','pragwan'),
('homeloan','Praveena',8,'SBI','ponnalursbi','ganesh')
```

servicename	**corresp_Emp**	**no_custo**	**bname**	**branname**	**managername**
goldloan	Nivas	5	AXISBANK	kanigiriaxis	krishna
irrigationloan	Kushal	4	ICICI	skonda	vikram
educationaloan	Teja	3	CANARABANK	kondpicanara	githa
mobilebanking	Jaanu	5	HDFC	madhapurhdfc	ajay
creditdebitcards	vishwa	8	APGB	kmpliapgb	rahul
accountopening	datha	10	UNIONBANKOFINDIA	kmpliUBI	krunal
atm	manohar	7	KARURVYSYA	tanguturkvnk	sriram
savings	sreevidya	9	PUNJABNATIONAL	gandhinagarpnb	vishal
cheque	Sravani	6	INDIANBANK	chlkpuripetaindibk	pragwan
homeloan	Praveena	8	SBI	ponnalursbi	ganesh

USING JOIN COMMANDS:

sql>select bank_det.location,account.customername from bank_det left join account on bank_det.bank_name=account.bankname;

location	**customername**
VIJAYWADA	Pavan
CHENNAI	laxman
ONGOLE	ganesh
BANGLORE	ravi
BANGLORE	Bharath
MUMBAI	Arjum
KOLKATTA	srinu
TANGUTUR	vasu
PUNJAB	NULL
HYD	sai
HYD	Suresh
VIZAG	NULL

RESULT:

Bank Management database is created successfully.

AIM: To create data base for College with 6 Relation Tables.

DESCRIPTION: Creating a database with related tables in it using SQL commands and joining all the tables using join operations.

CREATING DATABASE:

```
sql>create database College;
sql>use College;
```

CREATING TABLES:

COLLEGE TABLE:

```
sql>create table college(
        c_id integer,primary key(c_id),cname char(30),location char(30),
       estd_year integer);
```

BUILDINGS TABLE:

```
sql>create table buildings(
       b_id integer,primary key(b_id),c_id integer,foreign key(c_id) references
       college(c_id),floors integer);
```

ROOMS TABLE:

```
sql>create table rooms(
        roomno integer,b_id integer,primary key(b_id),roomsize varchar(10));
```

FACULTY TABLE:

```
sql>create table faculty(
       f_id integer,primary key(f_id),fname varchar(30),fmobile bigint,
       subject char(30));
```

LABS TABLE:

```
sql>create table labs(
       labname varchar(30),f_id integer,foreign key(f_id)references
       faculty(f_id),branch char(10));
```

LIBRARY TABLE:

```
sql>create table library(
      rack_no integer primary key,branch char(10),no_of_books integer,
     b_id integer,foreign key(b_id) references buildings(b_id));
```

CLASSROOMS TABLE:

```
sql>create table classrooms(
     room_no varchar(10)primary key,branch varchar(10),section_no varchar(5),
    b_id integer,foreign key(b_id)references buildings(b_id));
```

DISPLAYING TABLES IN DATABASE:

```
sql>show tables;
```

```
Tables_in_College
   College
   buildings
    rooms
    faculty
    labs
    library
    classrooms
```

INSERTING DATA INTO TABLES:

```
sql>insert into college values
        (202,"vikas colleges","hyderabad",2006),
        (540,"vedha institutes","chennai",2012),
        (917,"Sri institutes","kakinada",2016),
         (600,"vignan institutes","ongole",1995),
         (105,"Rakesh institutes","chennai",2002),
          (205,"kamal institutes","kurnool",2000),
```

```
sql>select * from college;
```

```
c_id   cname                location    estd_year
202    vikas colleges       hyderabad   2006
540    vedha institutes     chennai     2012
917    sri institutes       kakinada    2016
600    vignan institutes    ongole      1995
105    Rakesh institutes    chennai     2002
205    kamal institutes     kurnool     2000
```

sql>insert into buildings values
(015,205,5),(023,202,6),(503,540,10),(456,345,8)
(512,540,2),(089,917,5),(098,600,9),(012,405,7);

sql>select * from buildings;

b_id	**c_id**	**floors**
015	205	5
023	202	6
503	540	10
456	345	8
512	540	2
089	917	5
098	600	9
012	405	7

sql>insert into rooms values
(20,015,”small”),(21,512,”medium”),(22,456,”large”),
(24,503,”small”),(25,089,”medium”),(26,890,”large”),
(28,023,”small”),(27,098,”medium”),(23,012,”large”);

sql>select * from rooms;

roomno	**b_id**	**roomsize**
20	015	small
21	512	medium
22	456	large
24	503	small
25	089	medium
26	890	large
28	023	small
27	098	medium
23	012	large

```
sql>insert into faculty values
        ( 02,”rajesh”,8976534567,”physics”),
        ( 03,”akshara”,9876534567,”chemistry”),
        ( 04,”kamala”,8976534578,”maths”),
        ( 05,”anuj”,8976598567,”sanskrit”),
        ( 06,”alekya”,8976534523,”java”),
        ( 07,”prerna”,8976534590,”c++”),
```

```
sql>select * from faculty;
```

f_id	**fname**	**fmobile**	**subject**
02	rajesh	8976534567	physics
03	akshara	9876534567	chemistry
04	kamala	8976534578	maths
05	anuj	8976598567	sanskrit
06	alekya	8976534523	java
07	prerna	8976534590	c++

```
sql>insert into labs values
        (“chemlab”,03,”chemical”),(“physicslab”,02,”eee”),
        (“javalab”,06,”cse”),(“c++lab”,07,”ece”),(“pspclab”,01,”ece”);
```

```
sql>select * from labs;
```

Labname	**f_id**	**Branch`**
Chemlab	03	chemical
physicslab	02	eee
javalab	06	cse
c++lab	07	ece
pspclab	01	ece

```
sql>insert into library values
        (12,”cse”,45,023),(13,”cse”,46,015),(14,”ece”,60,023),
        (15,”chem”,43,012),(16,”civil”,34,456),(17,”mech”,23,503),
```

```
sql>select * from library;
```

rackno	**branch**	**no_of_books**	**b_id**
12	cse	45	023
13	cse	46	015
14	ece	60	023
15	chem	43	012
16	civil	34	456
17	mech	23	503

```
sql>insert into classrooms values
     ( “w101”,”cse”,”cse4”,015),(“w102”,”cse”,”cse3”,503),
     ( “w106”,”cse”,”cse2”,456),(“w107”,”cse”,”cse1”,023),
     ( “w104”,”cse”,”lab1”,012),(“w108”,”cse”,”lab2”,098);
```

```
sql>select * from classrooms;
```

room_no	**branch**	**sectionno**	**b_id**
w101	cse	cse4	015
w102	cse	cse3	503
w106	cse	cse2	456
w107	cse	cse1	023
w104	cse	lab1	012
w108	cse	lab2	098

USING JOIN COMMANDS:

SQL> select faculty.fname as Facultyname,labs.labname Lab from faculty inner join lab on faculty.f_id=labs.f_id;

Facultyname	**Lab**
02	physicslab
03	Chemlab
06	javalab
07	c++lab

RESULT: College management database created successfully.

AIM: To create a database for Hospital with 5 related tables.

DESCRIPTION: Creating a database with related tables in it using SQL commands and joining all the tables using join operations.

CREATING DATABASE:

```
sql>create database Hospital;
sql>use Hospital;
```

CREATING TABLES:

<u>HOSPITAL TABLE:</u>

```
sql>create table Hospital(
        Hospital_ID int primary key,Hospital_name char(30),City
        char(20));
```

<u>DOCTOR TABLE:</u>

```
sql>create table Doctor(
         Doctor_ID int ,D_name char(30),Hospital_ID int,
         Qualification char(20),Salary int,
         foreign key(Hospital_ID) references
         Hospital(Hospital_ID));
```

<u>PATIENT TABLE:</u>

```
sql>create table Patient(
        Patient_ID int primary key,Patient_name char(20),Age int,
        Hospital_ID int,foreign key(Hospital_ID) references
        Hospital(Hospital_ID));
```

<u>PHARMACY TABLE:</u>

```
sql>create table Pharmacy(
      P_ID int primary key,P_name char(30),Hospital_ID int,foreign
      key(Hospital_ID) references Hospital(Hospital_ID));
```

REPORTS TABLE:

```
sql>create table Reports(
        Report_ID int primary key,Problem char(20),
        Date_of_admission varchar(10),Patient_ID int,foreign
        key(Patient_ID) references Patient(Patient_ID));
```

DISPLAYING TABLES IN DATABASE:

```
sql>show tables;
```

```
Tables_in_Hospital
    Doctor
    Hospital
    Patient
    Pharmacy
    Reports
```

INSERTING DATA INTO TABLES:

```
sql>insert into Hospital values(214,’Sangamitra’,’Ongole’),
(220,’SKY Dental’,’Vijayawada’),(221,’KIMS’,’Hyderabad’),
(224,’Apollo’,’Guntur’),(225,’Nayana’,’Kakinada’),(227,’Gandhi
Hospital’,’Hyderabad’),(228,’Medicover’,’Vizag’),(229,’Homeocare
International’,’Kakinada’),(230,’Srinidhi Nursing
Home’,’Amalapuram’),(231,’Aayush Hospital’,’Vijayawada’);
```

```
sql>select * from Hospital;
```

Hospital_ID	**Hospital_name**	**City**
214	Sangamitra	Ongole
220	SKY Dental	Vijayawada
221	KIMS	Hyderabad
224	Apollo	Guntur
225	Nayana	Kakinada
227	Gandhi Hospital	Hyderabad
228	Medicover	Vizag
229	Homeocare International	Kakinada
230	Srinidhi Nursing Home	Amalapuram
231	Aayush Hospital	Vijayawada

```
sql>insert into Doctor values(101,’Akshaya’,214,’MBBS’,70000),
   (102,’Navya’,220,’BDS’,80000),(103,’Priya’,221,’Gynaecology’,90000),
   (104,’Charan’,224,’MBBS’,70000),(105,’Rajesh’,225,’BOPTM’,95000),
   (106,’Anuhya’,227,’Orthopedic surgeon’,90000),
   (107,’Sandeep’,’Cardiology’100000),(108,’Mohammad
   Ali’,’BHMS’,90000),(109,’Meera’,’ENT’,80000),(110,’Harish
   Kumar’,’Endocrinology’,95000);
```

```
sql>select * from Doctor;
```

Doctor_ID	**Doctor_name**	**Hospital_ID**	**Qualification**	**Salary**
101	Akshaya	214	MBBS	70000
102	Navya	220	BDS	80000
103	Priya	221	Gynaecology	90000
104	Charan	224	MBBS	70000
105	Rajesh	225	BOPTM	95000
106	Anuhya	227	Orthopedic surgeon	90000
107	Sandeep	228	Cardiology	100000
108	Mohammad Ali	229	BHMS	90000
109	Meera	230	ENT	80000
110	Harish Kumar	231	Endocrinology	95000

```
sql>insert into Patient values(11,’Siri’,24,214),(12,’Balu’,22,220),
    (13,’Mahitha’,21,221),(14,’Ramya’,23,224),(15,’Rajeswari’,54,225),
    (16,’Raghava’,36,227),(17,’Ananthalakshmi’,54,228),(18,’Satya’,40,229),
    (19,’Venkatarao’,49,230),(20,’Subramanyam’,50,231);
```

sql>select * from Patient;

Patient_ID	**Patient_name**	**Age**	**Hospital_ID**
11	Siri	24	214
12	Balu	22	220
13	Mahitha	21	221
14	Ramya	23	224
15	Rajeswari	54	225
16	Raghava	36	227
17	Ananthalakshmi	54	228
18	Satya	40	229
19	Venkatarao	49	230
20	Subramanyam	50	231

sql>insert into Pharmacy values(10,’Medicox’,214),(20,’Dentalcare’,220),
(30,’Pharma’,221),(40,’Apollo’,224),(50,’Medicore’,225),(60,’Gandhi
medicals’,227),(70,’Medicals’,228),(80,’Homeocare Pharmacy’,229),
(89,’Srinidhi medicals’,230),(90,’Aayush Pharma’,231);

sql>select * from Pharmacy;

P_ID	**P_name**	**Hospital_ID**
10	Medicox	214
20	Dentalcare	220
30	Pharma	221
40	Apollo	224
50	Medicore	225
60	Gandhi medicals	227
70	Medicals	228
80	Homeocare Pharmacy	229
89	Srinidhi medicals	230
90	Aayush Pharma	231

sql>insert into Reports values(91,’Fever’,’21dec’,11),
(92,’Rootcanal’,’13jan’,12),(93,’PCOD’,’22jan’,13),
(94,’Foodpoison’,’2feb’,14),(95,’Eyeinfection’,’5feb’,15),(96,’Bone fracture’,’6feb’,16),(97,’Heartpain’,’21feb’,17),
(98,’Diabetes’,’22feb’,18),(99,’Hearing problem’,’4jan’,19),
(100,’Diabetes’,’30jan’,20);

sql>select * from Reports;

Report_ID	**Problem**	**Date_of_admission**	**Patient_ID**
91	Fever	21dec	11
92	Rootcanal	13jan	12
93	PCOD	22jan	13
94	Foodpoison	2feb	14
95	Eyeinfection	5feb	15
96	Bone fracture	6feb	16
97	Heartpain	21feb	17
98	Diabetes	22feb	18
99	Hearing problem	4jan	19
100	Diabetes	30jan	20

DISPLAYING TABLE AFTER USING JOINS:

sql>select
Hospital.Hospital_name,Doctor.Hospital_ID,Doctor.Qualification,
Patient.Patient_name,Pharmacy.P_name,Reports.Problem from Hospital left join Doctor on
Hospital.Hospital_ID=Doctor.Hospital_ID left join Patient on
Patient.Hospital_ID=Hospital.Hospital_ID left join Pharmacy on
Pharmacy.Hospital_ID=Patient.Hospital_ID left join Reports on
Reports.Hospital_ID=Hospital.Hospital_ID;

Hospital_name	**Hospital_ID**	**Qualification**	**Patient_name**	**P_name**	**Problem**
Sangamitra	214	MBBS	Siri	Medicox	Fever
SKY Dental	220	BDS	Balu	Dentalcare	Rootcanal
KIMS	221	Gynaecology	Mahitha	Pharma	PCOD
Apollo	224	MBBS	Ramya	Apollo	Foodpoison
Nayana	225	BOPTM	Rajeswari	Medicore	Eyeinfection
Gandhi Hospital	227	Orthopedic sugeon	Raghava	Gandhi medicals	Bone fracture
Medicover	228	Cardiology	Ananthalakshmi	Medicals	Heartpain
Homeocare International	229	BHMS	Satya	Homeocare Pharmacy	Diabetes
Srinidhi Nursing Home	230	ENT	Venkatarao	Srinidhi medicals	Hearing problem
Aayush Hospital	231	Endocrinology	Subramanyam	Aayush Pharma	Diabetes

Result: Thus Hospital database is created.

AIM: To create data base for Sports manage management with 7 Relation Tables.

DESCRIPTION: Creating a database with related tables in it using SQL commands and joining all the tables using join operations.

CREATING DATABASE:

```
    sql>create database Sports_management;
    sql>use Sports_management;
```

CREATING TABLES:

<u>PLAYER TABLE:</u>

```
sql>create table player(
                player_id int,p_name char(20),age integer,team_id int,primary
                key(player_id),foreign key(team_id)references team(team_id);
```

<u>MATCHES TABLE:</u>

```
sql>create table matches(
                 match_id int,time varchar(10),date varchar(20),winner char(20),
                 primary key(match_id));
```

<u>TEAM TABLE:</u>

```
sql>create table team(
                team_id int,t_name varchar(20),match_id integer,
                foreign key(match_id)references matches(match_id),
                primary key(team_id));
```

<u>COMPETITION TABLE:</u>

```
sql>creae table compitition(
                 team_id inr,sports_name char(20),type char(10));
```

<u>SPORTS_CLUB TABLE:</u>

```
sql>create table sports_club(
                 clubname varchar(10),location char(20),contact_no int);
```

<u>STADIUM TABLE:</u>

```
sql>create table stadium(
                 capacity varchar(10),city char(20),name char(20));
```

<u>COACH TABLE:</u>

```
sql> create table coach(
            cid integer,name char(20),email varchar(20),nationality
            char(10),player_id integer));
```

DISPLAYING TABLES IN DATABASE:

sql>show tables;

```
Tables_in_Hospital
-------------------
    Player
     matches
     team
     compitition
     sports_club
     stadium
```

INSERTING DATA INTO TABLES:

sql>insert into player values

```
( 1000,”shiva”,19,789),(1001,”tharun”,20,100),
(1002,”ram”19,310),(1003,”nandhu”,20,189),
(1004,”geetha”,19,810),(1005”chandu,20,789);
```

sql>select * from player;

player_id	**p_name**	**age**	**team_id**
1000	shiva	19	789
1001	tharun	20	100
1002	ram	19	310
1003	nandu	20	189
1004	geetha	19	810
1005	chandu	20	789

sql>insert into matches values

```
(101,”5:00pm”,”2021-11-11”,”ravi”),
(102,”5.00pm”,”2021-12-11”,”tharun”),
(311,”1.00am”,”2021-07-13”,”shiva”),
(342”11.00am”,”2022-07-13”,”ram”),
(567,”10.00am”,”2022-05-13”,”shiva”),
```

sql>select * from matches;

match_id	**time**	**date**	**winner**
101	5:00pm	2021-11-11	ravi
102	5.00pm	2021-12-11	tharun
311	1.00am	2021-07-13	shiva
342	11.00am	2022-07-13	ram
567	10.00am	2022-05-13	shiva

sql>insert into team values
(100,”tiger”,311),(189,”champions”,101),(234,”lions”,567),
(310,”7stars”,789),(789,”stars”,567),(810,”hard workers”,102),
(647,”champions”,789);

team_id	**t_name**	**match_id**
100	Tigers	311
189	champions	101
234	lions	567
310	7stars	789
789	stars	567
810	hardworkers	102
647	champions	789

sql>insert into competition values
(“throwball”,”singles”,100),(“shuttle”,”doubles”,789),
(“running”,”distance”,310),(“hocky”,”outdoor”,810),
(“chess”,”indoor”,189),(“skipping”,”jumping”,810);

sport_name	**type**	**team_id**
throwball	singles	100
shuttle	doubles	789
running	distance	310
hocky	outdoor	810
chess	indoor	189
skipping	jumping	810

```
sql>insert into sports_club
            ("club1","west_block",9780107290),
            ("club2","south_block",8701234589),
            ("club3","near first floor",8700123456),
            ("club4","near library",6300175767),
```

clubname	location	contact_no
club1	west_block	9780107290
club2	south_block	8701234589
club3	near first floor	8700123456
club4	near library	6300175767

```
sql>insert into stadium values
        ("30 mem","vizag","Wembley stadium"),
        ("50 mem","Delhi","Narendramodi stadium"),
        ("15 mem","Russian","Rose Bowl"),
        ("20 mem","japan","Estadio Azteca");
```

capacity	city	name
30 mem	vizag	Wembley stadium
50 mem	Delhi	Narendramodi stadium
15 mem	Russian	Rose Bowl
20 mem	japan	Estadio Azteca

```
sql>insert into coach values
           (121,"ajay","ajay@gmail.com","india",1004),
           (135,"shiva","shiva@gmail.com","india",1002),
          (990,"ramesh","ramesh@gmail.com","england",1000),
          (700,"ram","ram@gmail.com","england",1001),
          (789,"ravi","ravi@gmail.com","russian",1005);
```

cid	name	email	nationality	Player_id
121	ajay	ajay@gmail.com	india	1004
135	shiva	shiva@gmail.com	india	1002
990	ramesh	ramesh@gmail.com	england	1000
700	ram	ram@gmail.com	england	1001
789	ravi	ravi@gmail.com	russian	1005

DISPLAYING TABLE AFTER USING JOINS:

```
sql>select date,time,t_name from matches leftjoin team on
    matches.match_id=team.match_id;
```

date	**time**	**t_name**
2021-11-11	5:00pm	champions
2021-11-11	5:00pm	hardworkers
2021-07-13	1:00pm	tigers
2022-07-13	11:00pm	Null
2022-05-13	10:00pm	lions
2121-12-23	4:00pm	7stars
2022-05-13	10:00pm	stars

RESULT:

The sports management database is successfully created.

PL/SQL

PL/SQL

NOTE: For some of the programs which are based on tables, you may need to create related tables before execution of the respective programs.

Introduction(Basic Program):

SQL>Write a program to display welcome message.

```
      BEGIN  DBMS_OUTPUT.PUT_LINE('HAI');
      DBMS_OUTPUT.PUT_LINE('WELCOME');
      DBMS_OUTPUT.PUT_LINE('PL/SQL PROGRAMS');  END;
```

OUTPUT:

```
           HAI WELCOME
           PL/SQL PROGRAMS

           PL/SQL procedure successfully completed
```

1) PL/SQL Code using Basic Variable, Anchored Declarations, and Usage of Assignment Operation

a) Write a program to find sum of two integer numbers.

SQL>

```
     DECLARE  A
     NUMBER; B
     NUMBER; C
     NUMBER; BEGIN
     A:=100; B:=200;
     C:=A+B;
     DBMS_OUTPUT.PUT_LINE('THE SUM OF TWO INTEGERS IS:  '||C);  END;
```

OUTPUT:

```
          THE SUM OF TWO INTEGERS IS:  300
```

PL/SQL procedure successfully completed.

b) Write a program to accept empno,ename,sal & calculate bonus on the following condition 20% on ann_sal.

SQL>

```
            DECLARE  EMPNO NUMBER;
            ENAME VARCHAR2(20); SAL
            NUMBER(7,2); ANU_SAL
            NUMBER(10,2); BONUS NUMBER(10,2);
            BEGIN
            EMPNO:=1234; ENAME:='Ravi';
            SAL:=18000; ANU_SAL:=SAL*12;
            BONUS:=ANU_SAL*20/100;
            DBMS_OUTPUT.PUT_LINE('EMPNO: '||EMPNO);
          DBMS_OUTPUT.PUT_LINE('ENAME: '||ENAME);
          DBMS_OUTPUT.PUT_LINE('SAL:  '||SAL);
          DBMS_OUTPUT.PUT_LINE('BONUS: '||BONUS);  END;
```

OUTPUT:

```
                 EMPNO: 1234
                 ENAME: ravi SAL:

                 18000

                 BONUS: 43200

Statement processed.
```

c) Write a program to accept product no,pname,quantity,price & calculate total,discount(20% on total),net bill.

SQL>

```
            DECLARE  PRODNO NUMBER;
          PNAME VARCHAR2(20); QUAN
          NUMBER(3); PRICE NUMBER(7,2);
          TOTAL NUMBER(7,2); DISCOUNT
          NUMBER(7,2); NET NUMBER(7,2);
          BEGIN
            PRODNO:=1234;
          PNAME:='Chocolates'; QUAN:=10;
          PRICE:=100; TOTAL:=QUAN*PRICE;
            DISCOUNT:=TOTAL*20/100;
```

```
NET:=TOTAL-DISCOUNT; DBMS_OUTPUT.PUT_LINE('PRODNO: '||
PRODNO);  DBMS_OUTPUT.PUT_LINE('PNAME: '||PNAME);
DBMS_OUTPUT.PUT_LINE('QUANTITY: '||QUAN);
DBMS_OUTPUT.PUT_LINE('PRICE: '||PRICE);
DBMS_OUTPUT.PUT_LINE('TOTAL: '||TOTAL);
DBMS_OUTPUT.PUT_LINE('DISCOUNT: '||DISCOUNT);
DBMS_OUTPUT.PUT_LINE('NET BALANCE: '||NET);  END;
```

OUTPUT:

```
PRODNO: 1234

PNAME: Chocolates QUANTITY: 10

PRICE: 100

TOTAL: 1000

DISCOUNT: 200

NET BALANCE: 800
```

2. Write a PL/SQL block using SQL and Control Structures in PL/SQL

a)Write a program to accept empno,sal,calculate bonus based on the following conditions

Salary	**Bonus**
>=10000	**20% on ann_sal**
>=5000&<10000	**15% on ann_sal**
>=3000&<5000	**12% on ann_sal**
>=1500&<3000	**10% on ann_sal**
>1500	**8% on ann_sal**

SQL>

```
DECLARE    EMPNO NUMBER;
 SAL NUMBER(7,2);  ANU_SAL
 NUMBER(7,2); BONUS NUMBER(7,2);
 BEGIN
 EMPNO:=1234; SAL:=8000;
 ANU_SAL:=SAL*12; IF SAL>=10000
 THEN
 BONUS:=ANU_SAL*20/100;
 ELSIF SAL>=5000 AND SAL<10000 THEN  BONUS:=ANU_SAL*15/100;
ELSIF SAL>=3000 AND SAL<5000 THEN BONUS:=ANU_SAL*12/100;
ELSIF SAL>=1500 AND SAL<3000 THEN BONUS:=ANU_SAL*10/100;
ELSE BONUS:=ANU_SAL*8/100; END IF;
DBMS_OUTPUT.PUT_LINE('EMPNO: '||EMPNO);
DBMS_OUTPUT.PUT_LINE('SAL: '||SAL);
DBMS_OUTPUT.PUT_LINE('ANU_SAL: '||ANU_SAL);
DBMS_OUTPUT.PUT_LINE('BONUS: '||BONUS); END;
```

OUTPUT:

```
EMPNO: 1234
SAL: 8000

ANU_SAL: 96000

BONUS: 14400

Statement processed.
```

b)Write a Program to print numbers from 10-1.

SQL>

```
DECLARE I NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('THE NUMBERS ARE'); FOR I IN
REVERSE 1..10 LOOP DBMS_OUTPUT.PUT_LINE(I);
END LOOP; END;
```

OUTPUT:

```
THE NUMBERS ARE 10
9
8
7
6
5
4
3
2
1

PL/SQL procedure successfully completed.
```

c) Write a Program to accept a date & print next 7 days along with day.

SQL>

```
        DECLARE DA DATE; I
        NUMBER; BEGIN
        DA:='10-04-2010'; FOR I IN 1..7
        LOOP
        DBMS_OUTPUT.PUT_LINE('THE DATE IS:'||DA); DA:=DA+1;
        END LOOP; END;
```

OUTPUT:

```
            THE DATE IS:10/04/2010 THE DATE
IS:10/05/2010
THE  DATE  IS:10/06/2010  THE  DATE
IS:10/07/2010       THE       DATE
IS:10/08/2010       THE       DATE
IS:10/09/2010       THE       DATE
IS:10/10/2010
Statement processed
```

d) Write a Program to display dept details

Note: Create a table with name DEPT and columns DEPTNO,DNAME,LOC with data inserted into it before running the below program and write ouptut according to the data inserted.

```
SQL> CREATE TABLE DEPT(
          deptno    varchar(20),    dname
          varchar(20), loc varchar(20)
    );
```

```
SQL> INSERT INTO DEPT values(101,”Kakinada”,”Development”);  INSERT
     INTO DEPT values(102,’Ongole’,’Designing’);  INSERT INTO DEPT
     values(103,’Guntur’,’sales’);
);
```

SQL>

```
DECLARE
CURSOR EC IS SELECT * FROM DEPT; BEGIN
FOR V_EC IN EC LOOP
DBMS_OUTPUT.PUT_LINE('DEPTNO='||V_EC.DEPTNO); DBMS_OUTPUT.PUT_LINE('DNAME='||
V_EC.DNAME); DBMS_OUTPUT.PUT_LINE('LOC='||V_EC.LOC);
END LOOP; END;
```

Output:

```
DEPTNO=101
DNAME=Ongole
LOC=Designing
DEPTNO=103
DNAME=Guntur LOC=Sales
DEPTNO=102
DNAME=Kakinada
LOC=Development
```

3. Write a PL/SQL Code using Cursors, Exceptions and Composite Data Types

a)Write a Program to calc bonus for all emps insert into bonus table

```
SQL>CREATE TABLE EMP(EMPNO NUMBER(5) PRIMARY KEY,ESAL NUMBER(5)); SQL>  INSERT

INTO EMP(EMPNO,ESAL) VALUES(7698,2850)
 SQL>   INSERT INTO EMP(EMPNO,ESAL) VALUES(7839,5000)
 SQL>  INSERT INTO EMP(EMPNO,ESAL) VALUES(7499,1760)
  SQL>  INSERT INTO EMP(EMPNO,ESAL) VALUES(7782,2450)
 SQL>  INSERT INTO EMP(EMPNO,ESAL) VALUES(7566,2975)
 SQL>  INSERT INTO EMP(EMPNO,ESAL) VALUES(7654,1375)
SQL>  CREATE TABLE BONUS1(EMPNO NUMBER(5) PRIMARY KEY,BONUS_AMT
      NUMBER(10,3),ADD_AMT NUMBER(10,3),ISS_DATE DATE);
```

SQL>

```
        DECLARE
        CURSOR EC IS SELECT EMPNO,ESAL FROM EMP; V_EC EC
        %ROWTYPE;
        ANN_SAL NUMBER(10,2); B
        BONUS1%ROWTYPE; BEGIN
        OPEN EC; LOOP
        FETCH EC INTO V_EC;
        EXIT WHEN EC%NOTFOUND; ANN_SAL
        :=V_EC.ESAL*12; B.BONUS_AMT := ANN_SAL*0.2;
        INSERT INTO BONUS1(EMPNO,BONUS_AMT,ADD_AMT,ISS_DATE)
       VALUES(V_EC.EMPNO,B.BONUS_AMT,1000,SYSDATE);
        END LOOP; CLOSE EC;
        END;
```

```
SQL>  SELECT *FROM BONUS1;
```

OUTPUT:

EMPNO	BONUS_AMT	ADD_AMT	ISS_DATE
7698	6840	1000	02/07/2023
7654	3300	1000	02/07/2023
7499	4224	1000	02/07/2023
7782	5880	1000	02/07/2023
7839	12000	1000	02/07/2023
7566	7140	1000	02/07/2023

4. Write a PL/SQL Code using Procedures, Functions, and Packages FORMS

a)Write a program with Procedure to print Min of two numbers in PL/SQL

NOTE: Run both sql1,sql2 commands in each section one after another,not at a time.

```
SQL1>
CREATE OR REPLACE PROCEDURE min(x IN number, y IN number, z OUT number)
IS BEGIN
IF x<y THEN
z:=x; ELSE
z:=y; END IF;
END;

SQL2>
DECLARE
a number;
b number;
c number;
BEGIN
a:=125;
 b:=40;
min(a,b,c);
dbms_output.put_line(c);
END;


OUTPUT:40
Statement processed.
```

b)Write a program with Procedure to print Square of a number in PL/SQL.

```
SQL1>

CREATE OR REPLACE PROCEDURE square(x IN OUT number)
IS
BEGIN
x:=x*x;
END;
```

```
SQL2>

DECLARE
a number;
BEGIN a:=10;
square(a);
dbms_output.put_line(‘Square of 10 is: ’||a);
END;
```

```
OUTPUT:
Square of 10 is: 100 Statement processed.
```