A DIV (DSE):

INCLUDED PRACTICALS: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,17,18,19,21,22,24

1. Create a db called company consist of the following tables.

1.Emp (eno,ename, job,hiredate,salary,commission,deptno,)

2.dept(deptno,deptname,location)

eno is primary key in emp

deptno is primary key in dept

create table Emp(eno int(10),ename varchar(10),job varchar(10), hiredate date,salary varchar(10),commision varchar(10),deptno varchar(20));

create table dept(deptno varchar(20),deptname varchar(20),location varchar(20));

ALTER TABLE Emp ADD PRIMARY KEY (eno);

ALTER TABLE dept ADD PRIMARY KEY (deptno);

insert into Emp(eno,ename,job,hiredate,salary,commision,deptno) values (01,'ABC','manager',2022/01/02,'5000','2000','10');

insert into Emp(eno,ename,job,hiredate,salary,commision,deptno) values (02,'PQR','salesman',2022/01/02,'1001','500','20');

insert into Emp(eno,ename,job,hiredate,salary,commision,deptno) values (03,'XYZ','manager',2022/01/02,'1000','2500','10');

insert into Emp(eno,ename,job,hiredate,salary,commision,deptno) values (04,'LMN','salesman',2022/01/02,'500','2500','20');

insert into dept (deptno,deptname,location) values ('10','production','Pune');

insert into dept (deptno,deptname,location) values ('20','Marketing','Mumbai');

Solve Queries by SQL

1. List the maximum salary paid to salesman

SELECT MAX(salary)FROM Emp where job = 'salesman' ;

2. List name of emp whose name start with ‘I’

select * from Emp where ename like 'I%'

3. List details of emp who have joined before ’30-sept-81’

select * from Emp where hiredate < 30/09/1981;

4. List the emp details in the descending order of their basic salary

select * from Emp order by salary desc;

5. List of no. of emp & avg salary for emp in the dept no ‘20’

SELECT COUNT(ename)from Emp;

SELECT AVG(salary)from Emp where deptno = '20'

6. List the avg salary, minimum salary of the emp hiredatewise for dept no ‘10’.

SELECT AVG(salary) from Emp where deptno = '10' ;

SELECT MIN(salary) from Emp where deptno = '10' ;

7. List emp name and its department

select Emp.ename,dept.deptno from Emp inner join dept on Emp.deptno = dept.deptno;

8. List total salary paid to each department

SELECT SUM(salary) from Emp where deptno = '10';

SELECT SUM(salary) from Emp where deptno = '20';

9. List details of employee working in ‘Dev’ department

SELECT Emp.ename, dept.deptname from Emp inner join dept on Emp.deptno = dept. deptno where deptname = 'Dev';

10. Update salary of all employees in deptno 10 by 5 %.

update Emp set salary = salary + 5 where deptno = '10';

select * from Emp;

Q.2

1. employee (employee name, street, city) ,employee name is primary key

2. works (employee name, company name, salary)

3. company (company name, city) ,company name is primary key

4. manages (employee name, manager name)

create table employee(employeename varchar(20) primary key,street varchar(20),city varchar(20));

insert into employee(employeename, street,city) values ('Neha','A street','A city');

insert into employee(employeename, street,city) values ('Reesha','B street','B city');

insert into employee(employeename, street,city) values ('Ritika','C street','C city');

insert into employee(employeename, street,city) values ('Ritu','C street','C city');

insert into employee(employeename, street,city) values ('Ryan','A street','A city');

insert into employee(employeename, street,city) values ('Kelly','B street','B city');

create table company(companyname varchar(20) primary key,city varchar(20));

insert into company (companyname , city)values ('First Bank Corporation','A city');

insert into company (companyname , city)values('Small Bank Corporation','B city');

insert into company (companyname , city)values('No Bank Corporation','C city');

insert into company (companyname , city)values('Yes Bank Corporation','A city');

insert into company (companyname , city)values('More Bank Corporation','B city');

```
create table works(employeename varchar(20),companyname varchar(20),salary double);

insert into works (employeename,companyname, salary)values('Neha','First Bank
Corporation',40000);

insert into works (employeename,companyname, salary)values('Reesha','Small Bank
Corporation',30000);

insert into works (employeename,companyname, salary)values('Ritika','No Bank
Corporation',35000);

insert into works (employeename,companyname, salary)values('Ritu','Small Bank
Corporation',25000);

insert into works (employeename,companyname, salary)values('Ryan','First Bank
Corporation',15000);

insert into works (employeename,companyname, salary)values('Kelly','First Bank
Corporation',10000);


create table manages(employeename varchar(20),managername varchar(20));

insert into manages (employeename,managername )values ('Neha','Ryan');

insert into manages (employeename,managername )values('Neha','Kelly');

insert into manages (employeename,managername )values('Reesha','Ritu');
```

Give an expression in SQL for each of the following queries.

1. Find the names of all employees who work for First Bank Corporation.

```
select employeename from works where companyname='First Bank Corporation';
```

2. Find all employees who do not work for First Bank Coorporation

```
select employeename from works where companyname<>'First Bank Corporation';
```

3. Find the company that has most employees.

4. Find all companies located in every in which small bank corporation is located

5. Find details of employee having salary greater than 10,000.

select * from works where salary>10000;

6. Update salary of all employees who work for First Bank Corporation by 10%.

update works set salary=salary+10 where companyname ='First Bank Corporation';

select * from works;

7. Find employee and their managers.

Select * from manages;

8. Find the names, street and cities of all employees who work for First Bank Corporation and earn more than 10,000.

select e.employeename,e.street,e.cityfrom employee e, works w where e.employeename=w.employeename and companyname="First Bank Corporation"and salary > 10000 **;**

9. Find those companies whose employees earn a higher salary,on average, than the average salary at First Bank Corporation

select AVG(salary) from works where companyname='First Bank Corporation';

Q.3

The following tables form part of a database held in a relational DBMS:

Hotel (HotelNo, Name, City) HotelNo is the primary key

Room (RoomNo, HotelNo, Type, Price)

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

Guest (GuestNo, GuestName, GuestAddress) GuestNo is primary key

Room contains room details for each hotel and (HotelNo, RoomNo) forms the primary key.

Booking contains details of the bookings and the primary key comprises (HotelNo, GuestNo and DateFrom)

```
create table Hotel(hotelNo varchar(20) primary key , name varchar(40), city varchar (40));

create table Room(roomno varchar(20)primary key,hotelno varchar (20),type varchar(20),price varchar(20));

create table Booking(hotelNo varchar(20),guestno varchar(20),dateFrom varchar(20),dateTo varchar(20),roomno varchar(20));

create table Guest(guestno varchar(20)primary key,guestname varchar(20),guestaddress varchar(50));


insert into Hotel(hotelNo,name,city)values ('01','Grosvenor','Newyork');

insert into Hotel(hotelNo,name,city)values ('02','Indigo','Delhi');

insert into Hotel(hotelNo,name,city)values ('03','Zen','London');

insert into Hotel(hotelNo,name,city)values ('04','Italia','Chikago');

insert into Hotel(hotelNo,name,city)values ('05','Bukhara','Los Angeles');


insert into Room(roomno,hotelNo,type,price)values('11','01','suit','12000');

insert into Room(roomno,hotelNo,type,price)values('13','01','presedential suit','100000');

insert into Room(roomno,hotelNo,type,price)values('14','03','deluxe','8000');

insert into Room(roomno,hotelNo,type,price)values('15','04','studio','15000');

insert into Room(roomno,hotelNo,type,price)values('16','05','super deluxe','14000');


insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','22',2022/08/02,2022/09/03,'11');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','23',2021/10/04,2021/10/05,'13');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('03','24',2020/07/08,2020/07/09,'14');
```

```
insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('05','25',2022/08/07,2022/08/08,'16');


insert into Guest(guestno,guestname,guestaddress) values ('23','ABC','Newyork');

insert into Guest(guestno,guestname,guestaddress) values ('24','ABC','London');

insert into Guest(guestno,guestname,guestaddress) values ('25','ABC','Delhi');

insert into Guest(guestno,guestname,guestaddress) values ('22','ABC','Mumbai');
```

Solve following queries by SQL

1. List full details of all hotels.

```
SELECT * FROM Hotel;
```

2. How many hotels are there?

```
SELECT COUNT(*) FROM Hotel;
```

3. List the price and type of all rooms at the Grosvenor Hotel.

```
SELECT price, type FROM Room WHERE hotelNo = (SELECT hotelNo FROM Hotel
WHERE name= ‘Grosvenor Hotel’);
```

4. List the number of rooms in each hotel.

```
SELECT hotelNo, COUNT(roomNo) AS count FROM Room GROUP BY hotelNo;
```

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

```
Update Room set price=price+5;
```

6. List full details of all hotels in London.

```
SELECT * FROM Hotel WHERE city = ‘London’;
```

7. What is the average price of a room?

```
SELECT AVG(price) FROM Room;
```

8. List all guests currently staying at the Grosvenor Hotel.

```
SELECT * FROM Guest WHERE guestno = (SELECT guestNo FROM Booking WHERE
dateFrom <= CURRENT_DATE AND dateTo >= CURRENT_DATE AND hotelNo = (SELECT
hotelNo FROM Hotel WHERE name = ‘Grosvenor’));
```

9. List the number of rooms in each hotel in London.

SELECT hotelNo, COUNT(roomNo) AS count FROM Room r, Hotel h WHERE r.hotelNo = h.hotelNo AND city = ‘London’ GROUP BY hotelNo;

10.Create one view on above database and query it.

create view show as select hotelno,name from Hotel;

if it gives error then put show (i.e view_name in square brackets [ ])

Q4. **The following tables form part of a database held in a relational DBMS:**

Hotel (HotelNo, Name, City) HotelNo is primary key

Room (RoomNo, HotelNo, Type, Price)

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

Guest (GuestNo, GuestName, GuestAddress) GuestNo is primary key

create table Hotel(hotelno varchar(20) primary key , name varchar(40), city varchar (40));

create table Room(roomno varchar(20)primary key,hotelno varchar (20),type varchar(20),price varchar(20));

create table Booking(hotelno varchar(20),guestno varchar(20),datefrom varchar(20),dateto varchar(20),roomno varchar(20));

create table Guest(guestno varchar(20)primary key,guestname varchar(20),guestaddress varchar(50));

insert into Hotel(hotelno,name,city)values ('01','Grosvenor','Newyork');

insert into Hotel(hotelno,name,city)values ('02','Indigo','Delhi');

insert into Hotel(hotelno,name,city)values ('03','Zen','London');

insert into Hotel(hotelno,name,city)values ('04','Italia','Chikago');

insert into Hotel(hotelno,name,city)values ('05','Bukhara','Los Angeles');

insert into Room(roomno,hotelno,type1,price)values('11','01','double','12000');

insert into Room(roomno,hotelno,type1,price)values('13','01','presedential suit','100000');

```
insert into Room(roomno,hotelno,type1,price)values('14','03','deluxe','8000');

insert into Room(roomno,hotelno,type1,price)values('15','04','studio','15000');

insert into Room(roomno,hotelno,type1,price)values('16','05','family','14000');


insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','22','2022/08/02','2022/08/03','11');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','23','2021/10/04','2021/10/05','13');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('03','24','2020/07/08','2020/07/09','14');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('05','25','2022/08/07','2022/08/08','16');


insert into Guest(guestno,guestname,guestaddress) values ('23','ABC','Newyork');

insert into Guest(guestno,guestname,guestaddress) values ('24','ABC','London');

insert into Guest(guestno,guestname,guestaddress) values ('25','ABC','Delhi');

insert into Guest(guestno,guestname,guestaddress) values ('22','ABC','Mumbai');
```

Solve following queries by SQL

1. What is the total revenue per night from all double rooms?

```
select SUM(price)from Room where type1 = 'double';
```

2. List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

```
SELECT r.* FROM Room r LEFT JOIN (SELECT g.guestname, h.hotelno, b.roomno FROM
Guest g, Booking b, Hotel h WHERE g.guestno = b.guestno AND b.hotelno = h.hotelno AND
name='Grosvenor' AND datefrom <= CURRENT_DATE AND dateto >= CURRENT_DATE)
AS XXX ON r.hotelno = XXX.hotelno AND r.roomno = XXX.roomno;
```

3. What is the average number of bookings for each hotel in April?

```
SELECT COUNT(DISTINCT guestNo) FROM BookingWHERE (datefrom <='2022-08-01'
AND dateto>='2022-08-01') OR (datefrom >='2022-08-01' AND datefrom <= '2022-08-31');
```

4. Create index on one of the field and show is performance in query.

CREATE INDEX showON Hotel (hotelno, name);

5. List full details of all hotels.

select h.hotelno,h.name,h.city,r.type1,r.price from Hotel h, Room r ;

6. List full details of all hotels in London.

SELECT * FROM Hotel WHERE city = 'London';

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

update Room set price = price + 5;

select * from Room;

8. List the number of rooms in each hotel in London.

SELECT h.hotelno ,COUNT(roomNo) AS count FROM Room r, Hotel h WHERE r.hotelno = h.hotelno AND city = 'London' GROUP BY hotelno;

9. List all double or family rooms with a price below £40.00 per night, in ascending order of price

SELECT * FROM Room WHERE price < '40' AND type IN ('double', 'family')

ORDER BY price;

Q.5The following tables form part of a database held in a relational DBMS:

Hotel (HotelNo, Name, City) HotelNo is the primary key

Room (RoomNo, HotelNo, Type, Price)

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

Guest (GuestNo, GuestName, GuestAddress)

create table Hotel(hotelno varchar(20) primary key , name varchar(40), city varchar (40));

create table Room(roomno varchar(20)primary key,hotelno varchar (20),type varchar(20),price varchar(20));

create table Booking(hotelno varchar(20),guestno varchar(20),datefrom varchar(20),dateto varchar(20),roomno varchar(20));

create table Guest(guestno varchar(20)primary key,guestname varchar(20),guestaddress varchar(50));

```
insert into Hotel(hotelno,name,city)values ('01','Grosvenor','Newyork');
insert into Hotel(hotelno,name,city)values ('02','Indigo','Delhi');
insert into Hotel(hotelno,name,city)values ('03','Zen','London');
insert into Hotel(hotelno,name,city)values ('04','Italia','Chikago');
insert into Hotel(hotelno,name,city)values ('05','Bukhara','Los Angeles');

insert into Room(roomno,hotelno,type1,price)values('11','01','double','12000');
insert into Room(roomno,hotelno,type1,price)values('13','01','presedential suit','100000');
insert into Room(roomno,hotelno,type1,price)values('14','03','deluxe','8000');
insert into Room(roomno,hotelno,type1,price)values('15','04','studio','15000');
insert into Room(roomno,hotelno,type1,price)values('16','05','family','14000');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','22','2022/08/02','2022/08/03','11');
insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','23','2021/10/04','2021/10/05','13');
insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('03','24','2020/07/08','2020/07/09','14');
insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('05','25','2022/08/07','2022/08/08','16');

insert into Guest(guestno,guestname,guestaddress) values ('23','ABC','Newyork');
insert into Guest(guestno,guestname,guestaddress) values ('24','ABC','London');
insert into Guest(guestno,guestname,guestaddress) values ('25','ABC','Delhi');
insert into Guest(guestno,guestname,guestaddress) values ('22','ABC','Mumbai');
```

Solve following queries by SQL

1. List full details of all hotels.

select h.hotelno,h.name,h.city,r.type1,r.price from Hotel h, Room r ;

2. How many hotels are there?

select count(name) from Hotel;

3. List the price and type of all rooms at the Grosvenor Hotel.

select type,price from Room where name= '**Grosvenor'**

4. List the number of rooms in each hotel

select count(room_no) as noofrooms,hotelno from room group by hotel_no;

5. List all guests currently staying at the Grosvenor Hotel.

6. List all double or family rooms with a price below £40.00 per night, in ascending order of price.

SELECT * FROM Room WHERE price < '40' AND type1 IN ('double', 'family')

ORDER BY price;

7. How many different guests have made bookings for August?

select guestno from Booking where datefrom between '2022/08/01' and '2022/08/31';

8. What is the total income from bookings for the Grosvenor Hotel today?

SELECT SUM(PRICE) FROM ROOM WHERE HOTELNO=(SELECT HOTELNO FROM HOTEL WHERE HOTELNAME='Grosvenor') and

9. What is the most commonly booked room type for each hotel in London?

select MAX(type) from Room where hotelno =(select hotelno from Hotel where city='London');

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

Update Room set price=price+5;

Q.6 The following tables form part of a database held in a relational DBMS:

Hotel (HotelNo, Name, City)

Room (RoomNo, HotelNo, Type, Price)

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

Guest (GuestNo, GuestName, GuestAddress)

```
create table Hotel(hotelno varchar(20) primary key , name varchar(40), city varchar (40));

create table Room(roomno varchar(20)primary key,hotelno varchar (20),type varchar(20),price
varchar(20));

create table Booking(hotelno varchar(20),guestno varchar(20),datefrom varchar(20),dateto
varchar(20),roomno varchar(20));

create table Guest(guestno varchar(20)primary key,guestname varchar(20),guestaddress
varchar(50));


insert into Hotel(hotelno,name,city)values ('01','Grosvenor','Newyork');

insert into Hotel(hotelno,name,city)values ('02','Indigo','Delhi');

insert into Hotel(hotelno,name,city)values ('03','Zen','London');

insert into Hotel(hotelno,name,city)values ('04','Italia','Chikago');

insert into Hotel(hotelno,name,city)values ('05','Bukhara','Los Angeles');


insert into Room(roomno,hotelno,type1,price)values('11','01','double','12000');

insert into Room(roomno,hotelno,type1,price)values('13','01','presedential suit','100000');

insert into Room(roomno,hotelno,type1,price)values('14','03','deluxe','8000');

insert into Room(roomno,hotelno,type1,price)values('15','04','studio','15000');

insert into Room(roomno,hotelno,type1,price)values('16','05','family','14000');


insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','22','2022/08/02','2022/08/03','11');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('01','23','2021/10/04','2021/10/05','13');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('03','24','2020/07/08','2020/07/09','14');

insert into Booking
(hotelno,guestno,datefrom,dateto,roomno)values('05','25','2022/08/07','2022/08/08','16');
```

insert into Guest(guestno,guestname,guestaddress) values ('23','ABC','Newyork');

insert into Guest(guestno,guestname,guestaddress) values ('24','ABC','London');

insert into Guest(guestno,guestname,guestaddress) values ('25','ABC','Delhi');

insert into Guest(guestno,guestname,guestaddress) values ('22','ABC','Mumbai');

Solve following queries by SQL

1. List full details of all hotels.

select h.hotelno,h.name,h.city,r.type1,r.price from Hotel h, Room r ;

2. List full details of all hotels in London.

SELECT * FROM Hotel WHERE city = 'London';

3. List all guests currently staying at the Grosvenor Hotel.

SELECT * FROM Guest WHERE guestno = (SELECT guestNo FROM Booking WHERE dateFrom <= CURRENT_DATE AND dateTo >= CURRENT_DATE AND hotelNo = (SELECT hotelNo FROM Hotel WHERE name = ‘Grosvenor’));

4. List the names and addresses of all guests in London, alphabetically ordered by name.

select guestname , guestaddress from Guest where guestaddress = 'London' order by guestname;

5. List the bookings for which no date_to has been specified.

select * from Booking where dateto = 'null';

6. How many hotels are there?

select count(name) from Hotel;

7. List the rooms that are currently unoccupied at the Grosvenor Hotel.

Select roomno from room where hotelno=(select hotelno from hotel where name=’grosnevor’)

8. What is the lost income from unoccupied rooms at each hotel today?

9. Create index on one of the field and show is performance in query.

CREATE INDEX showON Hotel (hotelno, name);

10. Create one view on above database and query it

CREATE VIEW hotel_view ASSELECT name, cityFROM Hotel;

UPDATE hotel_view SET name = 'India meal' WHERE name = 'Indigo';   (query on view)

select * from hotel_view;

7. Consider the following database
Project(project_id,proj_name,chief_arch) , project_id is primary key
Employee(Emp_id,Emp_name) , Emp_id is primary key
Assigned-To(Project_id,Emp_id)

---

create table Project(project_id varchar(10),proj_name varchar(20),chief_arch varchar(20));

create table Employee(Emp_id int,Emp_name varchar(20));

alter table Project add primary key(project_id);

alter table Emp add primary key(Emp_id);

create table Assigned_To(project_id varchar(5),Emp_id int);

//create table Assigned_To(project_id int, foreign key(project_id) references Project(project_id), Emp_id int , foreign key (Emp_id) references Employee(Emp_id));

insert into Project
Values('C353','Database','MYSQL'),('C354','JAVA','Ecplise'),('C453','PYTHON','Pycharm');

insert into Employee Values(123,'Swapnil'),(124,'Akshay'),(125,'Ritul');

insert into Assigned_To values('C353',123),('C353',124),('C354',125);

1.Get the details of employees working on project C353
select emp_id from Assigned_To where projectid = 'C353';

2. Get employee number of employees working on project C353

```
select A.emp_id, emp_name from Assigned_To A , Employee where project_id =
'C353' ;
//select count(*) from Assigned_To  , Employee where project_id = 'C353' ;
```

3. Obtain details of employees working on Database project

```
select Emp_name, A. Emp_id from A. Assigned_To A, Employee where project_id
in (select P.project_id from P. project where P. proj_name = 'Database');
```

4. Get details of employees working on both C353 and C354

```
select Emp_name, A.emp_id from Assigned_to A, Employee where A.Project_id =
'C354' union select Emp_name, A.emp_id from Assigned_to A, Employee where
A.Project_id = 'C353';
```

5. Get employee numbers of employees who do not work on project C453

8. Consider the following database
Employee(emp_no,name,skill,pay-rate) eno primary key
Position(posting_no,skill) posting_no primary key
Duty_allocation(posting_no,emp_no,day,shift)
Find the SQL queries for the following:

```
create table Employee(emp_no int, primary key(emp_no),name text,skill
text,pay_rate int);

create table Positions(posting_no int, primary key(posting_no),skill text);

create table Duty_allocation(posting_no int ,foreign key(posting_no) references
Positions(posting_no),emp_no int ,foreign key(emp_no) references
Employee(emp_no),day date,shift text);
```

1. Get the duty allocation details for emp_no 123461 for the month of April 1986.

```
select posting_no., shift, day
from Duty_allocation
where emp_no = 123461 and
Day ≥ 1986-04-01 and Day ≤ 1986-04-30 ;
```

2. Find the shift details for Employee ‘xyz’

```
select posting_no., shift, day
from Duty_allocation, Employee
where Duty allocation.emp_no. = Employee.emp_no and
Name = 'XYZ';
```

3. Get employees whose rate of pay is more than or equal to the rate of pay of employee ‘xyz’

```
select S.name, S.pay_rate from Employee as S, Employee as T where S.pay_rate >
T.pay_rate and T.name = 'XYZ';
```

4. Get the names and pay rates of employees with emp_no less than 123460 whose rate of pay is more than the rate of pay of at least one employee with emp_no greater than or equal to 123460.

```
Select name, pay_rate  from Employee where emp_no < 123460 and pay_rate >
some (select pay_rate from Employee where emp_no ≥ 123460);
```

5. Find the names of employees who are assigned to all positions that require a Chef’s skill

```
select S.Name from Employee S where (select posting_no from Duty_allocation D
where S.emp_no = D.emp_no) contains (select P.posting_no from position P where
P.skill = 'Chef');
```

6 .Find the employees with the lowest pay rate

```
select emp_no, Name, Pay_rate from Employee where pay_rate ≤ all (select
pay_rate from Employee)
```

7 .Get the employee numbers of all employees working on at least two dates.

```
select emp_no from Duty_allocation group by emp_no having (count;*) > 1
```

8 .Get a list of names of employees with the skill of Chef who are assigned a duty

select Name from Employee where emp_no in ((select emp_no from Employee where skill = 'Chef') intersect (select emp_no from Duty_allocation));

9 .Get a list of employees not assigned a duty

(select emp_no from Employee) minus (select emp_no from Duty_allocation)

10.Get a count of different employees on each shift

select shift, count (distinct emp_no) from Duty_allocation group by shift;

---

9. **Create the following tables. And Solve following queries by SQL**

- **Deposit (actno,cname,bname,amount,adate)**
- **Branch (bname,city)**
- **Customers (cname, city)**
- **Borrow(loanno,cname,bname, amount) Add primary key and foreign key wherever applicable. Insert data into the above created tables.**

create table deposit (actno varchar(5) ,cname varchar(18) , bname varchar(18) , amount int ,adate date);

create table branch(bname varchar(18),city varchar(18));

create table customers(cname varchar(19) ,city varchar(18));

create table borrow(loanno varchar(5), cname varchar(18), bname varchar(18), amount int);

deposit:

insert into deposit values('100',’anil’,'vrce',1000,'1995-03-01');
insert into deposit values('101','sunil','ajni',5000,'1996-01-04');
insert into deposit values('102','mehul','karolbagh',3500,'1995-11-17');
insert into deposit values('104','madhuri','chandi',1200,'1995-12-17');
insert into deposit values('105','prmod','m.g.road',3000,'1996-03-27');
insert into deposit values('106','sandip','andheri',2000,'1996-03-31');

```
insert into deposit values('107','shivani','virar',1000,'1995-07-05');
insert into deposit values('108','kranti','nehruplace',5000,'1996-06-02');
insert into deposit values('109','minu','powai',7000,'1997-12-02');
```

branch:

```
insert into branch values('vrce','nagpur');
insert into branch values('ajni','nagpur');
insert into branch values('karolbagh','delhi');
insert into branch values('chandi','delhi');
insert into branch values('dharampeth','nagpur');
insert into branch values('m.g.road','banglore');
insert into branch values('andheri','bombay');
insert into branch values('vihar','bombay');
insert into branch values('nehru place','delhi');
insert into branch values('powai','bombay');
```

customer:

```
insert into customers values ('anil','calcutta');
insert into customers values ('sunil','delhi');
insert into customers values ('mehul','baroda');
insert into customers values ('mandar','patna');
insert into customers values ('madhuri','nagpur');
insert into customers values ('pramod','nagpur');
insert into customers values ('sandip','surat');
insert into customers values ('shivani','bombay');
insert into customers values ('kranti','bombay');
insert into customers values ('naren','bombay');
```

borrow:

```
insert into borrow values ('201','anil','vrce',1000);
insert into borrow values ('206','mehul','vrce',5000);
insert into borrow values ('311','sunil','dharampeth',3000);
insert into borrow values ('321','madhuri','andheri',2000);
insert into borrow values ('375','prmod','vihar',8000);
insert into borrow values ('481','kranti','nehru place',3000);
```

1. Display names of depositors having amount greater than 4000.
SELECT CNAME FROM DEPOSIT WHERE AMOUNT >4000;

2. Display account date of customers Anil
Select adate from Deposit where cname=’Anil’;

3. Display account no. and deposit amount of customers having account opened between dates 1-12-96 and 1-5-97
SELECT act_no, AMOUNT FROM DEPOSIT WHERE ADATE BETWEEN ‘1996-12-01’ AND ’1997-05-01’;

4. Find the average account balance at the Perryridge branch.
select avg (balance) from account where branch-name = “Perryridge”
5. Find the names of all branches where the average account balance is more than $1,200.
select branch-name, avg-balance from (select branch-name, avg (balance) from account group by branch-name) as result (branch-name, avg-balance) where avg-balance > 1200

6. Delete depositors having deposit less than 5000
Delete from deposit where amount <5000;

7. Create a view on deposit table.
create View deposit_view as select actno,cname,bname,amount,adate from deposit;
select * from deposit_view;

10. **Create the following tables. And Solve following queries by SQL**
1. Deposit (actno,cname,bname,amount,adate)
2. Branch (bname,city)
3. Customers (cname, city)
4. Borrow(loanno,cname,bname, amount)
Add primary key and foreign key wherever applicable.
Insert data into the above created tables.

Use Question 9 Structure

a. **Display names of all branches located in city Bombay.**

Select * from Branch where city=’Bombay’

b. **Display account no. and amount of depositors.**

Select actno, amount from deposit

c. **Update the city of customers Anil from Pune to Mumbai**

Update Customers set city=’Mumbai’ where city=’Pune’

d. **Find the number of depositors in the bank**

select count (distinct cname) from deposit

e. **Calculate Min,Max amount of customers.**

f. **Create an index on deposit table**

create index deposit_index on deposit(actno);

g. Create View on Borrow table.

Create view borrow_view as select bname,city from borrow;

Select * from borrow_view;

11. Create the following tables. Solve queries by SQL

- **Deposit (actno,cname,bname,amount,adate)**
- **Branch (bname,city)**
- **Customers (cname, city)**
- **Borrow(loanno,cname,bname, amount)**

Add primary key and foreign key wherever applicable. Insert data into the above created tables.

Use Question 9 structure

a. **Display account date of customers Anil.**

Select adate form deposit where cname=’Anil’;

b. **Modify the size of attribute of amount in deposit**

c. **Display names of customers living in city pune.**
Select cname form customers where city=’Pune’

d. **Display name of the city where branch KAROLBAGH is located.**
Select city from branch where bname=’KAROLBAGH’

e. **Find the number of tuples in the customer relation**
select count (*) from customer

f. **Delete all the record of customers Sunil**
delete * from customer where cname=’Sunil’

g. **Create a view on deposit table**
create View deposit_view as select actno,cname,bname,amount,adate from deposit;
select * from deposit_view;

12. Create the following tables. Solve queries by SQL
- **Deposit (actno,cname,bname,amount,adate)**
- **Branch (bname,city)**
- **Customers (cname, city)**
- **Borrow(loanno,cname,bname, amount)**

Add primary key and foreign key wherever applicable. Insert data into the above created tables. Solve following queries by SQL

Use question 9 Structure

1. **Display customer name having living city Bombay and branch city Nagpur**
   select c.city from customer c, branch b where c.city=’bombay’ and b.city=’nagpur’ ;

2. **Display customer name having same living city as their branch city**
   select c.city from customer c, branch b where c.city=b.city ;

3. **Display customer name who are borrowers as well as depositors and having living city Nagpur.**

Select cname form deposit d , borrow b, customers c where d.cname=b.name, d.cname=c.cname and c.city=’Nagpur’

4. **Display borrower names having deposit amount greater than 1000 and loan amount greater than 2000**

select br1.cname, br1.amount, d1.cname, d1.amount from borrow br1,deposit d1 where d1.cname = br1.cname and d1.amount > 1000 and br1.amount >2000;

5. **Display customer name living in the city where branch of depositor sunil is located.**

select c.cname from customer c where c.city in (select b.city from branch b where b.bname in (select d.bname from deposit d where d.cname='sunil'));

6. **6. Create an index on deposit table**

create index deposit_index on deposit(actno);

13) Create the following tables.
1)PUBLISHER(PID , PNAME ,ADDRESS ,STATE ,PHONE ,EMAILID);
2)BOOK(ISBN ,BOOK_TITLE , CATEGORY , PRICE , COPYRIGHT_DATE , YEAR ,PAGE_COUNT ,PID);
3) AUTHOR(AID,ANAME,STATE,CITY ,ZIP,PHONE,URL)
4) AUTHOR_BOOK(AID,ISBN);
5) REVIEW(RID,ISBN,RATING);
Solve following queries by SQL

create table publisher(pid int, pname varchar(50), address varchar(50), state varchar(50), phone varchar(50), emailid varchar(50));

create table book(isbn varchar(50),book_title varchar(50), category varchar(50), price int, copyright_date int , year int,page_count int ,pid int);

create table author(aid int,aname varchar(50),state varchar(50),city varchar(50),zip int,phone varchar(50),url varchar(50));

create table author_book(aid int,isbn varchar(50));

create table review(rid int,isbn varchar(50),rating int);

Publisher

insert into publisher values(1, 'sunrise', 'mumbai', 'maharashtra', '9098765432', 'sunrise12@gmail.com');
insert into publisher values (2, 'mehta','pune', 'maharashtra', '9128765432', 'addison 12@gmail.com');
insert into publisher values (3,'morgan kaufmann', 'korth', 'maharashtra', '9548765432', 'morgan12@gmail.com');

Book:

insert into book values ('0321228383', 'database systems', 'a', 255, 12, 2007, 86, 1);
insert into book values ('0321228384', 'computer science', 'b', 205, 12, 2007, 80, 2);
insert into book values ('0321228385', 'out of their minds', 'c', 145, 12, 2007, 70, 3);

Author

insert into author values (10, 'chetan bhagat', 'maharashtra', 'mumbai', 401205, '9098765432', 'www.k10.com');
insert into author values (20, 'lewis', 'maharashtra', 'pune',410501, '9128765432', 'www.lewis20.com');
insert into author values (30, 'bernstein', 'maharashtra', 'korth', 402501, '9548765432', 'www.bern30.com');

Author_book

insert into author_book values (10,'0321228383');
insert into author_book values (20,'0321228384');
insert into author_book values (30,'0321228385');

Review

insert into review values(201, '0321228383', 4);
insert into review values(202, '0321228384', 3);

insert into review values(203, '0321228385', 4);

1. **Retrieve city, phone, url of author whose name is ‘CHETAN BHAGAT’.**
   select city,phone,url from author where aname='Chetan Bhagat';

2. **Retrieve book title, reviewable id and rating of all books.**
   select book_title,rid,rating from review r,book b where b.isbn=r.isbn;

3. **Retrieve book title, price, author name and url for publishers ‘MEHTA’.**
   select book_title,price,aname,url from book b,author a,publisher p where b.pid=p.pid and p.pname = 'MEHTA';

4. In a PUBLISHER relation change the phone number of ‘MEHTA’ to 123456

update publisher set phone='123456' where pname='mehta';

5. Calculate and display the average, maximum, minimum price of each publisher.

select avg(price),min(price),max(price) from book, publisher where book.pid=publisher.pid;

6. Delete details of all books having a page count less than 100.

delete from book where page_count < 100;

7. Retrieve details of all authors residing in city Pune and whose name begins with character ‘C’.

select * from author where city='Pune' and aname like 'C%';

8. Retrieve details of authors residing in same city as ‘Korth’.

select * from author where city='Korth';

9. Create a procedure to update the value of page count of a book of given ISBN.

10. Create a function that returns the price of book with a given ISBN.

14.A14. a) Consider table Stud(Roll, Att,Status) Write a PL/SQL block for following requirement and handle the exceptions. Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as “ND”

Write a PL/SQL block for following requirement and handle the exceptions. Roll no. of student

will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If

attendance is less than 75% then display the message “Term not granted” and set the status in

stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as

“ND” ---------->>>>>>>>>>>>

```
create table stud(RollNo int primary key, attendance int,status varchar(5));

insert into stud values(1,150, NULL),(2,200, NULL),(3,80, NULL),(4,70, NULL),(5,180, NULL);

select * from stud;

delimiter //

create procedure check_att(in roll int)

begin

declare att int;

declare total int;

declare exit handler for not found select 'Data not found!!!' message;

set total=200;

select attendance into att from stud where RollNo=roll;

if ((att/total)*100)>=75 then
```

```
update stud set status='ND' where RollNo=roll;
select 'Term Granted' Message;
else
update stud set status='D' where RollNo=roll;
select 'Term Not Granted' Message;
end if;
end;
//

call check_att(1);
call check_att(2);
call check_att(3);
select * from stud;
```

14.B b) Write a PL/SQL block for following requirement using user defined exception handling. The account_master table records the current balance for an account, which is updated whenever, any deposits or withdrawals takes place. If the withdrawal attempted is more than the current balance held in the account. The user defined exception is raised, displaying an appropriate message. Write a PL/SQL block for above requirement using user defined exception handling.

Write a PL/SQL block for following requirement using user defined exception handling. The

account_master table records the current balance for an account, which is updated whenever, any

deposits or withdrawals takes place. If the withdrawal attempted is more than the current balance

held in the account. The user defined exception is raised, displaying an appropriate message.

Write a PL/SQL block for above requirement using user defined exception handling.

---------->>>>>>

```
create table account_master(ID int primary key,Current_balance int);

insert into account_master values(1,10000),(2,5000),(3,60000);

select*from account_master;


delimiter //

create procedure withdraw(in acc_id int,in amt int)

begin

declare bal int;

declare sp condition for sqlstate '45000';

select Current_balance into bal from account_master where ID=acc_id;

if bal<amt then
```

```
signal sqlstate '45000'
set message_text='NotEnoughBalance';
else
set bal = bal-amt;
update account_master set Current_balance=bal where ID=acc_id;
end if;
end;
//

create procedure deposit(in acc_id int,in amt int)
begin
declare bal int;
select current_balance into bal from account_master where ID=acc_id;
update account_master set current_balance=bal+amt where ID=acc_id;
end;
//


call withdraw(3,40000);
select*from account_master;

call deposit(2,2000);
select*from account_master;

call withdraw(1,75000);
```

15A Write an SQL code block these raise a user defined exception where business rule is
voilated. BR for client_ master table specifies when the value of bal_due field is less than 0
handle the exception.

```
delimiter //
create procedure check_br(in uid int)
begin
declare temp_bal int;
declare sp condition for sqlstate'45000';
select bal_due into temp_bal from client_master where id=uid;
if temp_bal<0 then
signal sqlstate '45000'
 set message_text='BR violated';
 else
 select 'BR not violated' Message;
 end if;
 end
 //
```

q15 b ---------->

Write an SQL code block

Borrow(Roll_no, Name, DateofIssue, NameofBook, Status)

Fine(Roll_no,Date,Amt)

Accept roll_no & name of book from user. Check the number of days (from date of issue), if

days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine

will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status

will change from I to R. If condition of fine is true, then details will be stored into fine table.

Also handles the exception by named exception handler or user define exception handler.

create table borrow(

```
roll_no int primary key,
name varchar(50),
dateofissue date,
nameofbook varchar(50),
status varchar(50));

create table fine(
roll_no int primary key,
dateofreturn date,
amt int);

insert into borrow values(1,'A','2022-08-15','java','I');
insert into borrow values(2,'A','2022-08-05','cns','I');
insert into borrow values(3,'A','2022-08-01','dbms','I');
insert into borrow values(4,'A','2022-08-01','spos','I');

delimiter $
create procedure fine_calculation(in rno int,bookname varchar(20))
begin
declare issuedate date;
declare diff int;
declare fine_amt int;
declare exit handler for sqlexception select 'Table not Found';
select dateofissue into issuedate from borrow where roll_no=rno and nameofbook=bookname;
select datediff (curdate(),issuedate) into diff;
if (diff>15 and diff<30) then
set fine_amt = diff*5;
```

```
insert into fine values(rno, curdate(),fine_amt);
elseif(diff>30) then
set fine_amt = diff*50;
insert into fine values(rno, curdate(),fine_amt);
elseif(diff<15) then
set fine_amt = 0;
insert into fine values(rno, curdate(),fine_amt);
end if;
update borrow set status ='R' where roll_no=rno and nameofbook=bookname;
end;
$


call fine_calculation(3,'dbms');
call fine_calculation(4,'spos');
call fine_calculation(2,'cns');
call fine_calculation(1,'java');

select*from fine;
select*from borrow;
```

16. Cursor (Any Two) a) The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update. (Use of %FOUND, %NOTFOUND, %ROWCOUNT)

SQL> create table bank_manager(

2 id number(3) not null primary key,

3 inactive_days number(3)

4);

Table created.

SQL> insert into bank_manager (id, inactive_days) values (01,256);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (02,456);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (03,545);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (04,222);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (05,120);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (06,03);

1 row created.

SQL> select * from bank_manager;

ID	INACTIVE_DAYS
1	256
2	456
3	545
4	222
5	120
6	3

6 rows selected.

SQL> alter table bank_manager add status number(2) ;

Table altered.

```
SQL> select * from bank_manager;

        ID INACTIVE_DAYS     STATUS
---------- ------------ ----------
         1          256
         2          456
         3          545
         4          222
         5          120
         6            3

6 rows selected.

SQL> edit
Wrote file afiedt.buf

  1  declare
  2     total_rows number(3);
  3  begin
  4     update bank_manager set status = 1 where inactive_days>356;
  5     if sql%notfound then
  6           dbms_output.put_line('No Record Found');
  7     elsifsql%found then
  8           total_rows := sql%rowcount;
  9           dbms_output.put_line('Account Updated: '||total_rows);
 10     end if;
```

```
 11* end;
SQL> /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> /
Account Updated: 2

PL/SQL procedure successfully completed.

SQL> select * from bank_manager;

        ID INACTIVE_DAYS     STATUS
---------- ------------ ----------
         1          256
         2          456          1
         3          545          1
         4          222
         5          120
         6            3

6 rows selected.

SQL>
```

b)Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization, Whenever such salary updates takes place, a record for the same is maintained in the increment_salary table.

```
SQL> create table employee2(
  2    id number not null primary key,
  3    name varchar2(20),
  4    salary number(10,2) not null
  5  );

Table created.

SQL> insert into employee2(id,name,salary) values (1,'Rushikesh',20000);

1 row created.

SQL> insert into employee2(id,name,salary) values (2,'Ritul',30000);

1 row created.

SQL> insert into employee2(id,name,salary) values (3,'Sanket',35000);
```

1 row created.

SQL> insert into employee2(id,name,salary) values (4,'Isha',40000);

1 row created.

SQL> insert into employee2(id,name,salary) values (5,'Kunal',25000);

1 row created.

SQL> insert into employee2(id,name,salary) values (6,'Ranjit',18000);

1 row created.

SQL> select * from employee2;

ID	NAME	SALARY
1	Rushikesh	20000
2	Ritul	30000
3	Sanket	35000
4	Isha	40000
5	Kunal	25000
6	Ranjit	18000

```
6 rows selected.


SQL> edit
Wrote file afiedt.buf


  1  declare
  2    av_salary number(10,2);
  3  begin
  4    av_salary := &av_salary;
  5    update employee2 set salary = salary*0.10 where salary <av_salary;
  6    if sql%found then
  7          dbms_output.put_line('Rows Updated: '||sql%rowcount);
  8    elsifsql%notfound then
  9          dbms_output.put_line('No Record Found');
 10    end if;
 11* end;
SQL> /
Enter value for av_salary: 28000
old   4:       av_salary := &av_salary;
new   4:       av_salary := 28000;
PL/SQL procedure successfully completed.
SQL> set serveroutput on;
SQL> /
Enter value for av_salary: 28000
old   4:       av_salary := &av_salary;
```

new 4: av_salary := 28000;

Rows Updated: 3

PL/SQL procedure successfully completed.

c) Write PL/SQL block using explicit cursor for following requirements: College has decided to mark all those students detained (D) who are having attendance less than 75%. Whenever such update takes place, a record for the same is maintained in the D_Stud table. create table stud21(roll number(4), att number(4), status varchar(1));

```
SQL> create table stud21(
  2    roll number(4) not null primary key,
  3    att number(4) not null,
  4    status varchar(1)
  5  );
```

Table created.

```
SQL> insert into stud21 (roll,att) values (1,78);
```

1 row created.

SQL> insert into stud21 (roll,att) values (2,58);

1 row created.

SQL> insert into stud21 (roll,att) values (3,76);

1 row created.

SQL> insert into stud21 (roll,att) values (4,66);

1 row created.

SQL> insert into stud21 (roll,att) values (5,56);

1 row created.

SQL> insert into stud21 (roll,att) values (6,88);

1 row created.

SQL> create table d_stud(

2 roll number(4) not null,

3 att number(4) not null,

```
  4     status varchar(1)
  5  );

Table created.

SQL> set linesize 160;
SQL> select * from stud21;

     ROLL       ATT S
---------- ---------- -
         1         78
         2         58
         3         76
         4         66
         5         56
         6         88

6 rows selected.

SQL> declare
  2     cursor stu_cursor is
  3     select roll,att from stud21 where att<75;
  4     stud_recordstu_cursor%rowtype;
  5  begin
  6     open stu_cursor;
```

```
 7    loop
 8          fetch stu_cursor into stud_record;
 9          exit when stu_cursor%notfound;
10           insert into d_stud (roll,att) values (stud_record.roll,stud_record.att);
11           update stud21 set status = 'D' where roll = stud_record.roll;
12    end loop;
13  end;
14  /
```

PL/SQL procedure successfully completed.

SQL> select * from stud21;

ROLL	ATT	S
1	78	
2	58	D
3	76	
4	66	D
5	56	D
6	88	

6 rows selected.

SQL> select * from d_stud;

ROLL ATT S

--------- --------- -

2 58

4 66

5 56

SQL>

17. Cursor (Any Two) a) The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update. (Use of %FOUND, %NOTFOUND, %ROWCOUNT)

```
SQL> create table bank_manager(
  2     id number(3) not null primary key,
  3     inactive_days number(3)
  4  );
```

Table created.

```
SQL> insert into bank_manager (id, inactive_days) values (01,256);
```

1 row created.

```
SQL> insert into bank_manager (id, inactive_days) values (02,456);
```

1 row created.

```
SQL> insert into bank_manager (id, inactive_days) values (03,545);
```

1 row created.

```
SQL> insert into bank_manager (id, inactive_days) values (04,222);
```

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (05,120);

1 row created.

SQL> insert into bank_manager (id, inactive_days) values (06,03);

1 row created.

SQL> select * from bank_manager;

ID	INACTIVE_DAYS
1	256
2	456
3	545
4	222
5	120
6	3

6 rows selected.

SQL> alter table bank_manager add status number(2) ;

Table altered.

```
SQL> select * from bank_manager;

        ID INACTIVE_DAYS     STATUS
---------- ------------ ----------
         1          256
         2          456
         3          545
         4          222
         5          120
         6            3

6 rows selected.

SQL> edit
Wrote file afiedt.buf

  1  declare
  2     total_rows number(3);
  3  begin
  4     update bank_manager set status = 1 where inactive_days>356;
  5     if sql%notfound then
  6           dbms_output.put_line('No Record Found');
  7     elsifsql%found then
  8           total_rows := sql%rowcount;
  9           dbms_output.put_line('Account Updated: '||total_rows);
 10     end if;
 11* end;
```

```
SQL> /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> /
Account Updated: 2

PL/SQL procedure successfully completed.

SQL> select * from bank_manager;

        ID INACTIVE_DAYS     STATUS
---------- ------------- ----------
         1           256
         2           456          1
         3           545          1
         4           222
         5           120
         6             3

6 rows selected.
```

17 b) ..Write a PL/SQL block of code using parameterized Cursor, that will merge the data available

in the newly created table N_RollCall with the data available in the table O_RollCall. If the

data in the first table already exist in the second table then that data should be skipped. output:

-- Write a PL/SQL block of code using parameterized Cursor, that will merge the data available

-- in the newly created table N_RollCall with the data available in the table O_RollCall. If the

-- data in the first table already exist in the second table then that data should be skipped. output:

—----------_???/>>>>>>>>>>>>>>>>>>

```
create table n_rollcall (roll int, name varchar(10));
insert into n_rollcall values (2,'vishal'), (5,'pratik'), (6,'parth');

create table o_rollcall (roll int, name varchar(10));
insert into o_rollcall values (2,'vishal'), (4,'hettik'), (3,'kartik'), (1,'deepak'), (5,'pratik');

delimiter $
create procedure p3(in r1 int)
begin
declare r2 int;
declare exit_loop boolean;
declare c1 cursor for select roll from o_rollcall where roll>r1;
declare continue handler for not found set exit_loop=true;
```

```
open c1;
loop1:loop
fetch c1 into r2;
if not exists(select * from n_rollcall where roll=r2)
then
insert into n_rollcall select * from o_rollcall where roll=r2;
end if;
if exit_loop
then
close c1;
leave loop1;
end if;
end loop loop1;
end;
$

call p3(2);
select*from n_rollcall;
```

17c)--------->

17c—-------->Write the PL/SQL block for following requirements using parameterized Cursor: Consider

table EMP(e_no, d_no, Salary), department wise average salary should be inserted into new

table dept_salary(d_no, Avg_salary)

```
mysql> delimiter //
mysql> create procedure check_salary()
    -> begin
    -> declare temp_emp int;
    -> declare temp_dno int;
    -> declare temp_salary int;
    -> declare avg_salary int;
    -> declare temp_dno_dept_salary int;
    -> declare ec boolean;
    -> declare cur1 cursor for select avg(salary),dno from emp group
by dno;
    -> declare continue handler for not found set ec=true;
    -> open cur1;
    -> l1:loop
    -> fetch cur1 into temp_salary,temp_dno;
    -> insert into dept_salary values(temp_salary,temp_dno);
    -> if ec then
    -> close cur1;
    -> leave l1;
    -> end if;
    -> end loop l1;
    -> end
    -> //
```

18. TRIGGER: a

Write a update, delete trigger on clientmstr table. The System should keep track of the

records that ARE BEING updated or deleted. The old value of updated or deleted records

should be added in audit_trade table. (separate implementation using both row and statement

triggers).

```
mysql> CREATE TABLE LIB_AUDIT(RNO INT,
-> B_TITLE VARCHAR(20),
-> ACTION VARCHAR(20));^C
mysql> CREATE TABLE BOOKS(RNO INT,
-> B_TITLE VARCHAR(20));
Query OK, 0 rows affected (0.04 sec)
mysql>
mysql> CREATE TABLE LIB_AUDIT(RNO INT,
-> B_TITLE VARCHAR(20),
-> ACTION VARCHAR(20));
Query OK, 0 rows affected (0.03 sec)
mysql> DESC LIB_AUDIT;
+---------+-------------+------+-----+---------+-------+
| Field
```

| Type

| Null | Key | Default | Extra |

+---------+-------------+------+-----+---------+-------+

| RNO

| int

| YES |

| NULL

|

|

| B_TITLE | varchar(20) | YES |

| NULL

|

|

| ACTION | varchar(20) | YES |

| NULL

|

|

+---------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

mysql> DESC BOOKS;

+---------+-------------+------+-----+---------+-------+

| Field

| Type

| Null | Key | Default | Extra |

+---------+-------------+------+-----+---------+-------+

| RNO

| int

| YES |

| NULL

|

|

| B_TITLE | varchar(20) | YES |

| NULL

|

|

+---------+-------------+------+-----+---------+-------+

2 rows in set (0.00 sec)

mysql> INSERT INTO BOOKS VALUES(1, 'ABC');

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO BOOKS VALUES(2, 'DEF');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO BOOKS VALUES(3, 'GHI');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO BOOKS VALUES(4, 'JKL');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO BOOKS VALUES(5, 'MNO');

Query OK, 1 row affected (0.01 sec)mysql> SELECT *FROM BOOKS;

+------+---------+

| RNO | B_TITLE |

+------+---------+

|

1 | ABC

|

|

2 | DEF

|

|

3 | GHI

|

|

4 | JKL

|

|

5 | MNO

|

+------+---------+

5 rows in set (0.00 sec)

mysql> SELECT *FROM LIB_AUDIT;

Empty set (0.00 sec)

mysql> DELIMITER $

mysql> CREATE TRIGGER before_book_delete

-> AFTER DELETE

-> ON books

-> FOR EACH ROW

-> BEGIN

```
-> INSERT INTO LIB_AUDIT
-> SET action ='DELETE',
-> RNO=OLD.RNO,
-> B_TITLE=OLD.B_TITLE;
-> END;
-> $
Query OK, 0 rows affected (0.01 sec)
mysql> DELIMITER ;
mysql> DELETE FROM BOOKS WHERE RNO = 1;
Query OK, 1 row affected (0.01 sec)
mysql> SELECT *FROM BOOKS;
+------+---------+
| RNO | B_TITLE |
+------+---------+
|
2 | DEF
|
|
3 | GHI
|
|
4 | JKL
|
|
5 | MNO
```

|

+------+---------+

4 rows in set (0.00 sec)

mysql> SELECT *FROM LIB_AUDIT;+------+---------+--------+

| RNO | B_TITLE | ACTION |

+------+---------+--------+

|

1 | ABC

| DELETE |

+------+---------+--------+

1 row in set (0.00 sec)

mysql> DELIMITER $

mysql> CREATE TRIGGER before_book_update

-> BEFORE UPDATE

-> ON BOOKS

-> FOR EACH ROW

-> BEGIN

-> INSERT INTO LIB_AUDIT

-> SET action ='UPDATE',

-> RNO=NEW.RNO,

-> B_TITLE=NEW.B_TITLE;

-> END;

-> $

Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;

```
mysql> UPDATE BOOKS SET B_TITLE = 'XYZ' WHERE RNO = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> SELECT* FROM BOOKS;
+------+---------+
| RNO | B_TITLE |
+------+---------+
|
2 | XYZ
|
|
3 | GHI
|
|
4 | JKL
|
|
5 | MNO
|
+------+---------+
4 rows in set (0.00 sec)
mysql> SELECT* FROM LIB_AUDIT;
+------+---------+--------+
| RNO | B_TITLE | ACTION |
+------+---------+--------+
```

|

1 | ABC

| DELETE |

|

2 | XYZ

| UPDATE |

+------+---------+--------+

2 rows in set (0.00 sec)

18 a or —->

```
18A
delimiter //
create trigger after_delete
after delete on client_master
for each row
begin
insert into audit_table
set action='DELETE',
id=old.id,
data=old.data;
end
//

delimiter //
create trigger after_update
after update on client_master
for each row
begin
insert into audit_table
set action='UPDATE',
id=old.id,
data=old.data;
end
//
```

18B

```
delimiter //
create trigger after_insert
after insert
on emp
for each row
begin
if(new.salary<50000) then
signal sqlstate '45000' set message_text='Rejected!!!';
end if;
insert into tracking
set eno=new.eno,
salary=new.salary;
end
//
```

18._____>Write a before trigger for Insert, update event considering following requirement:

Emp(e_no, e_name, salary) I) Trigger action should be initiated when salary is tried to be

inserted is less than Rs. 50,000/- II) Trigger action should be initiated when salary is tried to be

updated for value less than Rs. 50,000/- Action should be rejection of update or Insert

operation by displaying appropriate error message. Also the new values expected to be inserted

will be stored in new table Tracking(e_no, salary)

```
CREATE TABLE Employee
(
  Id INT PRIMARY KEY,
  Name VARCHAR(45),
  Salary INT,
  Gender VARCHAR(12),
  DepartmentId INT
)
CREATE TABLE Audit2
(
  Salary INT
```

```
) ;
INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3);
INSERT INTO Employee VALUES (2,'XYZ', 79000, 'Female', 4);



CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON Employee
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  dbms_output.put_line('Old salary: ' || :OLD.salary);
sal_diff:= :OLD.salary;
  dbms_output.put_line('New salary: ' || :NEW.salary);
  insert into Audit2 values(sal_diff);
END;

update Employee set salary=85080 where id=2;

select * from Audit2;
```

19. Create Database DYPIT using MongoDB

Use DYPIT

Create following Collections Teachers(Tname,dno,dname,experience,salary,date_of_joining)

```
db.createCollection('Teachers')
db.Teachers.insertMany([{
    'Tname': 'Sojwal',
    'dno': 1,
    'dname': 'Computer',
    'experience':11,
    'salary':10001,
    'date_of_joining':'1/1/2001'
    },
    {
    'Tname': 'Omkar',
    'dno': 2,
    'dname': 'IT',
    'experience':5,
    'salary':100011,
```

```
    'date_of_joining':'2/2/2012'
    },
{
    'Tname': 'Arshad',
    'dno': 3,
    'dname': 'E&TC',
    'experience':17,
    'salary':200001,
    'date_of_joining':'9/6/1996'
    },
{
    'Tname': 'Akshay',
    'dno': 2,
    'dname': 'IT',
    'experience':7,
    'salary':10002,
    'date_of_joining':'1/1/2011'
    }])
```

Students(Sname,roll_no,class)

```
db.createCollection(‘Students’)
db.Students.insertMany([{
    'Sname': 'Rupesh',
    'roll_no': 1,
    'class': 'Computer'
    },
    {
```

'Sname': 'Ramdas',

'roll_no': 2,

'class': 'E&TC'

},

{

'Sname': 'Chetan',

'roll_no': 3,

'class': 'IT'

}])

1. Find the information about all teachers
   db.Teachers.find().pretty()
2. Find the information about all teachers of computer department
   db.Teachers.find({'dname':'Computer'}).pretty()
3. Find the information about all teachers of computer,IT,ande&TC department
   db.Teachers.find().pretty()
4. Find the information about all teachers of computer,IT,and E&TC department having salary greate than or equl to 10000/-

   db.Teachers.find({'salary':{$gte:10000}}).pretty()
5. Find the student information having roll_no = 2 or Sname=xyz
   db.Students.find({$or:[{'roll_no':2},{'Sname':'xyz'}]}).pretty()
6. Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.

   db.Teachers.insert({
   ... 'Tname': 'Praveen',
   ... 'dno': 3,
   ... 'dname': 'E&TC',
   ... 'experience':11,
   ... 'salary':5001,
   ... 'date_of_joining':'1/1/2021'
   ... })

   db.Teachers.updateOne({Tname:'Praveen'}, {$set:{experience:10}})

7. Update the deparment of all the teachers working in IT deprtment to COMP
   db.Teachers.updateMany({dname:'IT'}, {$set:{dname:'Computer'}})
8. find the teachers name and their experience from teachers collection
   db.Teachers.find({},{dname:0,dno:0,salary:0,date_of_joining:0}).pretty()

db.Teachers.find({},{dno:0,dname:0,salary:0,date_of_joining:0})

9. Using Save() method insert one entry in department collection

```
db.Teachers.save({
'Tname': 'Rajesh',
...     'dno': 1,
...     'dname': 'Computer',
...     'experience':8,
...     'salary':50001,
...     'date_of_joining':'1/1/2019'
})
```

10. Using Save() method change the dept of teacher Rajesh to IT

11. Delete all the doccuments from teachers collection having IT dept

    db.Teachers.deleteMany({“dname”:”IT”})

12. display with pretty() method, the first 3 doccuments in teachers collection in ascending order

    db.Teachers.find().sort({dno:1}).limit(3).pretty()

20 1.Create Database DYPIT
2. Create following Collections
Teachers(Tname,dno,dname,experience,salary,date_of_joining)
Students(Sname,roll_no,class)

3. Find the information about two teachers
db.Teachers.find().limit(2).pretty()
4. Find the information about all teachers of computer department

    db.Teachers.find({dname:'Computer'}).pretty()

5. Find the information about all teachers of computer,IT,ande&TC department

    Same as question 19

6.. Find the information about all teachers of computer,IT,and E&TC department having salary greate than or equl to 25000/-

db.Teachers.find({'salary':{$gte:25000}}).pretty()

7. Find the student information having roll_no = 25 or Sname=xyz

8. Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.

Same as 19

9. Update the deparment of all the teachers working in IT deprtment to COMP

Same as 19

10. find the teachers name and their experience from teachers collection

db.Teachers.find({},{dname:0,dno:0,salary:0,date_of_joining:0}).pretty()11. Using Save() method insert one entry in department collection

Same as 19

13. Delete all the doccuments from teachers collection having IT dept.
    Same as 19

14. display with pretty() method, the first 5 documents in teachers collection in ascending order

db.Teachers.find().sort({dno:1}).limit(5).pretty()

21. Create Database DYPIT using MongoDB Create following Collections
Teachers(Tname,dno,dname,experience,salary,date_of_joining)
Students(Sname,roll_no,class)

1. Find the information about all teachers
   db.Teachers.find().pretty()
2. Find the average salary teachers of computer department

db.Teachers.aggregate([{$match:{"dname":"Computer"}},{$group : {_id : "$dname", salary_maximum : {$avg : "$salary"}}}])

3. Find the minimum and maximum salary of e&TC department teachers
   db.Teachers.aggregate([{$match:{"dname":"E&TC"}},{$group : {_id : "$dname", salary_maximum : {$max : "$salary"}, salary_minimum:{$min : "$salary"}}}])

4. Find the information about all teachers of computer,IT,and E&TC department having salary greate than or equl to 10000/-

   db.Teachers.find({'salary':{$gte:10000}}).pretty()

5. Find the student information having roll_no = 2 or Sname=xyz
   Same as above questions
6. Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.
   Same s above questions.
7. Update the deparment of all the teachers working in IT deprtment to COMP
   Same as above
8. find the teachers name and their experience from teachers collection
9. db.Teachers.find({},{dname:0,dno:0,salary:0,date_of_joining:0}).pretty()Using Save() method insert one entry in department collection
   Same as above

10. Find the total salary all teachers.
    db.Teachers.aggregate([{$group : {_id : "", total_salary : {$sum : "$salary"}}}])

22. Create Database DYPIT using MongoDB Create following Collections
Teachers(Tname,dno,dname,experience,salary,date_of_joining)
Students(Sname,roll_no,class)

1. Display the department wise average salary

```
db.Teachers.aggregate([{$group : {_id : "$dname", salary_avarage : {$avg : "$salary"}}}])
```

2. display the no. Of employees working in each department

```
db.Teachers.aggregate( [ { $unwind: "$dname" },  { $sortByCount: "$dname" } ] )
```

3. Display the department wise total salary of departments having total salary greater than or equals to 50000/-

4. Write the queries using the different operators like max, min. Etc.

Refer above quetion

5. Create unique index on any field for above given collections

```
db.Teachers.createIndex({Tname:1}, {unique:true})
```

6. Create compound index on any fields for above given collections

7. Show all the indexes created in the database DYPIT

```
db.Teachers.getIndexes()
```

8. Show all the indexes created in above collections.

```
db.Teachers.getIndexes()
```

24. Design and Implement following query using MongoDB

1. Create a collection called ‘games’.

2. Add 5 games to the database. Give each document the following properties: name, gametype, rating (out of 100)

```
db.games.insertMany([{
    'name': 'life',
    'gametype': 'joke',
```

'rating': 100

},

{

'name': 'Crypto',

'gametype': 'Luck',

'rating': 10

},

{

'name': 'Solitare',

'gametype': 'card',

'rating': 80

},

{

'name': 'Pubg',

'gametype': 'FPS',

'rating': 80

},

{

'name': 'GTA',

'gametype': 'open_world',

'rating': 75

}])

3. Write a query that returns all the games

db.games.find().pretty()

4. Write a query that returns the 3 highest rated games.

db.games.find().sort({rating:-1}).limit(3).pretty()

5. Update your two favourite games to have two achievements called ‘Game Master’ and ‘Speed Demon’.

db.games.updateOne({name:"GTA"}, {$set:{achievements:"Game-master,Speed-daemon"}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

db.games.updateOne({name:"life"},

... {$set:{achievements:"Game-master","Speed-daemon"}})

6. Write a query that returns all the games that have both the ‘Game Maser’ . the ‘Speed Demon’ achievements.

db.games.find({"achievements":"Game-master,Speed-daemon"}).pretty()

8. Write a query that returns only games that have achievements

26. Using MapReduce in mongodb solve following queries on given below collection.

1. Import zip.json.

mongoimport --dbsai --collection zip --file C:\Users\OMKAR\Desktop\zips.json

2. Find total population in each state.

db.zip.mapReduce(function() {emit(this.state,this.pop);}, function(key,value){return Array.sum(value)}, { query:{state:"MA"},out:"state_pop_totals"});

db.state_pop_totals.find();

27.