Practicle 1:

**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**

**Solve Queries by SQL**

**1. List the maximum salary paid to salesman**

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

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

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

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

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

**7. List emp name and its department**

**8. List total salary paid to each department**

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

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

1. List the maximum salary paid to salesman

SELECT MAX(salary) FROM Emp WHERE job = 'Salesman';

--> SELECT MAX(salary) FROM Emp WHERE job = 'Salesman';

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

SELECT ename FROM Emp WHERE ename LIKE 'I%' ;

--> SELECT ename FROM Emp WHERE ename LIKE 'I%' ;

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

SELECT \* FROM Emp WHERE hiredate<('1991-09-30');

--> SELECT \* FROM Emp WHERE hiredate<('1991-09-30');

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

SELECT \* FROM Emp ORDER BY salary DESC;

--> SELECT \* FROM Emp ORDER BY salary DESC;

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

SELECT count(\*), avg(salary) From Emp Where deptno = 20;

--> SELECT count(\*), avg(salary) From Emp Where deptno = 20;

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

//Doubt : SELECT AVG(salary),Min(salary) FROM Emp WHERE deptno = 10;

--> //Doubt : SELECT AVG(salary),Min(salary) FROM Emp WHERE deptno = 10;

7. List emp name and its department

SELECT e.ename,d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

--> SELECT e.ename,d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

8. List total salary paid to each department

SELECT e.salary, d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

--> SELECT e.salary, d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

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

SELECT \* FROM Emp WHERE deptno IN

--> SELECT \* FROM Emp WHERE deptno IN

(SELECT deptno FROM dept WHERE deptname = 'Dev') ;

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

UPDATE Emp SET salary = (select salary+(select salary\*.5))

--> UPDATE Emp SET salary = (select salary+(select salary\*.5))

WHERE deptno = 10;

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

Practicle 2:

**Create a database**

**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)**

SQL Queries:

CREAE DATABASE Bank;

USE Bank;

CREATE TABLE employee(emp\_name VARCHAR(30), street VARCHAR(30), city VARCHAR(20), PRIMARY KEY (emp\_name));

CREATE TABLE works(emp\_name VARCHAR(30), com\_name VARCHAR(30), salary INT);

CREATE TABLE company(com\_name VARCHAR(30) PRIMARY KEY, city VARCHAR(30));

CREATE TABLE manages(emp\_name VARCHAR(30), man\_name VARCHAR(30));

insert into employee values ("Neha","A street","Pune"),("Reesha","B street","Mumbai"),("Ritika","C street","Delhi"),("Ritu","C street","Delhi"),("Ryan","A street","Pune"),("Kelly","B street","Mumbai");

insert into company values ("First Bank Corporation","Pune"),("Small Bank Corporation","Mumbai"),("No Bank Corporation","Delhi"),("Yes Bank Corporation","Pune"),("More Bank Corporation","Mumbai");

insert into works values ("Neha","First Bank Corporation",40000),("Reesha","Small Bank Corporation",30000),("Ritika","No Bank Corporation",35000),("Ritu","Small Bank Corporation",25000),("Ryan","First Bank Corporation",15000),("Kelly","First Bank Corporation",10000);

insert into manages values ("Neha","Ryan"),("Ritika","Kelly"),("Reesha","Ritu");

Solve Queries by SQL:

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

--> SELECT emp\_name FROM works WHERE com\_name = 'First Bank Corporation';

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

--> SELECT emp\_name FROM works WHERE com\_name <>'First Bank Corporation';

3. Find the company that has most employees.

--> SELECT com\_name FROM works GROUP BY com\_name ORDER BY COUNT(\*) DESC LIMIT 1;

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

--> 4th not done

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

--> SELECT e.emp\_name, e.street, e.city, w.com\_name, w.salary FROM works w,employee e WHERE salary > 10000 AND w.emp\_name = e.emp\_name;

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

--> UPDATE works SET salary = (salary + (salary\*.10)) WHERE com\_name = 'First Bank Corporation';

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.emp\_name, e.street, e.city FROM works w, employee e WHERE w.com\_name = 'First Bank Corporation' AND salary > 10000;

9. Find those companies whose employees earn a higher salary,on average, than th

average salary at First Bank Corporation

--> 9th not done

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

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

Practicle 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)**

SQL queries:

CREATE DATABASE HotelBooking;

USE HotelBooking;.

CREATE TABLE Hotel( hotel\_no INT, hotel\_name VARCHAR(20), city VARCHAR(20), PRIMARY KEY (hotel\_no));

CREATE TABLE Room(room\_no INT, hotel\_no INT, type VARCHAR(20), price DECIMAL, PRIMARY KEY(room\_no), FOREIGN KEY(hotel\_no) REFERENCES Hotel(hotel\_no));

CREATE TABLE Guest(guest\_no INT, guest\_name VARCHAR(20), g\_address VARCHAR(20), PRIMARY KEY(guest\_no));

CREATE TABLE Booking(hotel\_no INT, guest\_no INT, dateFrom DATE, dateTo DATE, room\_no INT, FOREIGN KEY(guest\_no) REFERENCES Guest(guest\_no), FOREIGN KEY(room\_no) REFERENCES Room(room\_no));

INSERT INTO Hotel( hotel\_no, hotel\_name, city) VALUES( 1, 'Grosvenor', 'London'),(2, 'RajPlaza', 'Pune'), (3, 'Alpine', 'Mumbai')( 4, 'Athens', 'London');

INSERT INTO Room( room\_no, hotel\_no, type, price) VALUES

(101, 1, 'single', 30),

(102, 1, 'double', 40),

(103, 1, 'family', 50),

(201, 2, 'single', 35),

(301, 3, 'double', 45),

(401, 4, 'double', 55),

(402, 4, 'family', 50);

INSERT INTO Guest( guest\_no, guest\_name, g\_address) VALUES

(11, 'Shriam', 'Kakslauttanen'),

(12, 'Druva', 'Jaipur'),

(13, 'Pravin', 'Latur'),

(14, 'Mahesh', 'Pune'),

(15, 'Atharva', 'Mumbai'),

(16, 'Krishna', 'London'),

(17, 'Sanket', 'Kolhapur'),

(18, 'Dhaval', 'Surat');

INSERT INTO Booking(hotel\_no, guest\_no, dateFrom, dateTo, room\_no) VALUES

(1, 11, '2022-08-25', '2022-11-25', 102),

(1, 12, '2022-04-24', '2022-12-02', 103),

(2, 16, '2022-05-23', '2022-08-13', 201),

(4, 17, '2022-09-22', '2022-10-11', 401),

(4, 13, '2022-09-21', '2022-12-18', 402);

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 type, price FROM Room WHERE hotel\_no = (SELECT hotel\_no FROM Hotel WHERE hotel\_name = 'Grosvenor');

4. List the number of rooms in each hotel.

--> SELECT hotel\_no, count(room\_no) AS count FROM Room GROUP BY (hotel\_no);

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

--> UPDATE Room SET price = price + price\*.05;

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) AS AvgPrice FROM Room;

8. List all guests currently staying at the Grosvenor Hotel

--> SELECT \* FROM Guest WHERE guest\_no IN (SELECT guest\_no FROM Booking WHERE dateFrom<=CURRENT\_DATE AND dateTo>=CURRENT\_DATE AND hotel\_no IN (SELECT hotel\_no FROM Hotel WHERE hotel\_name = 'Grosvenor'));

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

--> SELECT r.hotel\_no, count(room\_no) AS count FROM Room r, Hotel h WHERE r.hotel\_no = h.hotel\_no AND city = 'London' GROUP BY (r.hotel\_no);

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

--> CREATE VIEW room\_view AS

SELECT room\_no, type, price

FROM Room;

SELECT \* FROM room\_view;

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

**Practicle 4:**

**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**

SQL QUERIES:

CREATE DATABASE HotelBooking;

USE HotelBooking;.

CREATE TABLE Hotel( hotel\_no INT, hotel\_name VARCHAR(20), city VARCHAR(20), PRIMARY KEY (hotel\_no));

CREATE TABLE Room(room\_no INT, hotel\_no INT, type VARCHAR(20), price DECIMAL, PRIMARY KEY(room\_no), FOREIGN KEY(hotel\_no) REFERENCES Hotel(hotel\_no));

CREATE TABLE Guest(guest\_no INT, guest\_name VARCHAR(20), g\_address VARCHAR(20), PRIMARY KEY(guest\_no));

CREATE TABLE Booking(hotel\_no INT, guest\_no INT, dateFrom DATE, dateTo DATE, room\_no INT, FOREIGN KEY(guest\_no) REFERENCES Guest(guest\_no), FOREIGN KEY(room\_no) REFERENCES Room(room\_no));

INSERT INTO Hotel( hotel\_no, hotel\_name, city) VALUES( 1, 'Grosvenor', 'London'),(2, 'RajPlaza', 'Pune'), (3, 'Alpine', 'Mumbai')( 4, 'Athens', 'London');

INSERT INTO Room( room\_no, hotel\_no, type, price) VALUES

(101, 1, 'single', 30),

(102, 1, 'double', 40),

(103, 1, 'family', 50),

(201, 2, 'single', 35),

(301, 3, 'double', 45),

(401, 4, 'double', 55),

(402, 4, 'family', 50);

INSERT INTO Guest( guest\_no, guest\_name, g\_address) VALUES

(11, 'Shriam', 'Kakslauttanen'),

(12, 'Druva', 'Jaipur'),

(13, 'Pravin', 'Latur'),

(14, 'Mahesh', 'Pune'),

(15, 'Atharva', 'Mumbai'),

(16, 'Krishna', 'London'),

(17, 'Sanket', 'Kolhapur'),

(18, 'Dhaval', 'Surat');

INSERT INTO Booking(hotel\_no, guest\_no, dateFrom, dateTo, room\_no) VALUES

(1, 11, '2022-08-25', '2022-11-25', 102),

(1, 12, '2022-04-24', '2022-12-02', 103),

(2, 16, '2022-05-23', '2022-08-13', 201),

(4, 17, '2022-09-22', '2022-10-11', 401),

(4, 13, '2022-09-21', '2022-12-18', 402);

Solve following queries by SQL

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

--> SELECT SUM(price) AS TotalRevenue FROM Room WHERE type = '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 newt.guest\_name,r.\* FROM Room r JOIN(SELECT g.guest\_name, h.hotel\_no, b.room\_no FROM Hotel h, Booking b, Guest g WHERE h.hotel\_no = b.hotel\_no AND b.guest\_no = g.guest\_no AND hotel\_name = 'Grosvenor' AND dateFrom<=CURRENT\_DATE AND dateTo>=CURRENT\_DATE) AS newt ON r.hotel\_no = newt.hotel\_no AND r.room\_no = newt.room\_no;

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

--> SELECT hotel\_no FROM Booking WHERE dateFrom>=DATE'2022-04-01' AND dateTo<=DATE'2022-04-30' GROUP BY (hotel\_no);

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

--> CREATE INDEX guest\_name\_idx ON Guest(guest\_name);

5. List full details of all hotels.

--> SELECT \* FROM Hotel;

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 + price\*.05;

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

--> SELECT r.hotel\_no, COUNT(\*) AS count FROM Hotel h, Room r WHERE h.hotel\_no = r.hotel\_no AND city = 'London' GROUP BY(r.hotel\_no);

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

ascending order of price

--> SELECT type, price FROM Room WHERE price<40 AND (type='double' OR type='family') ORDER BY price ASC;

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

**Practicle 5:**

**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)**

SQL QUERIES:

CREATE DATABASE HotelBooking;

USE HotelBooking;.

CREATE TABLE Hotel( hotel\_no INT, hotel\_name VARCHAR(20), city VARCHAR(20), PRIMARY KEY (hotel\_no));

CREATE TABLE Room(room\_no INT, hotel\_no INT, type VARCHAR(20), price DECIMAL, PRIMARY KEY(room\_no), FOREIGN KEY(hotel\_no) REFERENCES Hotel(hotel\_no));

CREATE TABLE Guest(guest\_no INT, guest\_name VARCHAR(20), g\_address VARCHAR(20), PRIMARY KEY(guest\_no));

CREATE TABLE Booking(hotel\_no INT, guest\_no INT, dateFrom DATE, dateTo DATE, room\_no INT, FOREIGN KEY(guest\_no) REFERENCES Guest(guest\_no), FOREIGN KEY(room\_no) REFERENCES Room(room\_no));

INSERT INTO Hotel( hotel\_no, hotel\_name, city) VALUES( 1, 'Grosvenor', 'London'),(2, 'RajPlaza', 'Pune'), (3, 'Alpine', 'Mumbai')( 4, 'Athens', 'London');

INSERT INTO Room( room\_no, hotel\_no, type, price) VALUES

(101, 1, 'single', 30),

(102, 1, 'double', 40),

(103, 1, 'family', 50),

(201, 2, 'single', 35),

(301, 3, 'double', 45),

(401, 4, 'double', 55),

(402, 4, 'family', 50);

INSERT INTO Guest( guest\_no, guest\_name, g\_address) VALUES

(11, 'Shriam', 'Kakslauttanen'),

(12, 'Druva', 'Jaipur'),

(13, 'Pravin', 'Latur'),

(14, 'Mahesh', 'Pune'),

(15, 'Atharva', 'Mumbai'),

(16, 'Krishna', 'London'),

(17, 'Sanket', 'Kolhapur'),

(18, 'Dhaval', 'Surat');

INSERT INTO Booking(hotel\_no, guest\_no, dateFrom, dateTo, room\_no) VALUES

(1, 11, '2022-08-25', '2022-11-25', 102),

(1, 12, '2022-04-24', '2022-12-02', 103),

(2, 16, '2022-05-23', '2022-08-13', 201),

(4, 17, '2022-09-22', '2022-10-11', 401),

(4, 13, '2022-09-21', '2022-12-18', 402);

Solve following queries by SQL:

1. List full details of all hotels.

--> SELECT \* FROM Hotel;

2. How many hotels are there?

--> SELECT COUNT(hotel\_no) AS No\_Of\_Hotels FROM Hotel ;

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

--> SELECT type, price FROM Room WHERE hotel\_no = (SELECT hotel\_no FROM Hotel WHERE hotel\_name = 'Grosvenor');

4. List the number of rooms in each hotel

--> SELECT hotel\_no, count(room\_no) AS count FROM Room GROUP BY (hotel\_no);

5. List all guests currently staying at the Grosvenor Hotel

--> SELECT \* FROM Guest WHERE guest\_no IN (SELECT guest\_no FROM Booking WHERE dateFrom<=CURRENT\_DATE AND dateTo>=CURRENT\_DATE AND hotel\_no IN (SELECT hotel\_no FROM Hotel WHERE hotel\_name = 'Grosvenor'));

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

price.

--> SELECT type, price FROM Room WHERE price<40 AND (type='double' OR type='family') ORDER BY price ASC;

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

--> SELECT COUNT(guest\_no) FROM Booking WHERE (dateFrom <= DATE'2022-08-01' AND dateTo >= DATE'2022-08-31') OR (dateFrom >= DATE'2022-08-01' AND dateTo <= DATE'2022-08-31');

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

--> SELECT SUM(price) FROM Booking b, Room r, Hotel h WHERE (b.dateFrom <=CURRENT\_DATE AND b.dateTo >=CURRENT\_DATE) AND r.hotel\_no = h.hotel\_no AND r.hotel\_no = b.hotel\_no AND r.room\_no = b.room\_no AND h.hotel\_name = 'Grosvenor';

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

--> //not working//SELECT hotel\_no, type, MAX(count\_Type) FROM (SELECT hotel\_no,type, COUNT(type) AS count\_Type FROM Booking b, Hotel h, Room r, WHERE r.room\_no = b.room\_no AND r.hotel\_no = b.hotel\_no AND b.hotel\_no = h.hotel\_no AND city = 'London' GROUP BY hotel\_no,type) GROUP BY hotel\_no,type;

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

--> UPDATE Room SET price = price + price\*.05;

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

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

Practicle 6:

**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)**

**Solve following queries by SQL**

1. List full details of all hotels.

-> select \* from hotel;

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 hotelName = ‘Grosvenor Hotel’));

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

-> SELECT guestName, guestAddress FROM Guest WHERE address LIKE ‘%London%’

ORDER BY guestName;

5. List the bookings for which no date\_to has been specified.

-> SELECT \* FROM Booking WHERE dateTo IS NULL;

6. How many hotels are there?

-> SELECT COUNT(\*) FROM Hotel;

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

-> SELECT \* FROM room r

WHERE roomno NOT IN

(SELECT roomno FROM booking b, hotel h

WHERE (datefrom <= CURRENT\_DATE AND

dateto >= CURRENT\_DATE) AND

b.hotelno = h.hotelno AND hotelname = 'Grosvenor');

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

-> SELECT hotelno, SUM(price) FROM room r

WHERE roomno NOT IN

(SELECT roomno FROM booking b, hotel h

WHERE (datefrom <= CURRENT\_DATE AND

dateto >= CURRENT\_DATE) AND

b.hotelno = h.hotelno)

GROUP BY hotelno;

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

-> CREATE INDEX index\_hotel ON Hotel (hotelno,Name,City);

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

-> CREATE VIEW [Hotel Detail] AS SELECT Name,City FROM Hotel WHERE hotelno = 101 ;

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

Practicle 7:

**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)**

Find the SQL queries for the following:

1. Get the details of employees working on project C353

-> select A.empid, emp\_name from A.Assigned\_To A, Employee where project\_id = 'C353' ;

2. Get employee number of employees working on project C353

->select emp\_id from Assigned\_To where projectid = '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. project\_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)

intersect

(select emp\_name, A.empid from A.Assigned\_To A, Employee where project\_id = 'C354');

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

-> (select emp\_id from Employee)

minus

(select emp\_id from assigned\_to where project\_id = 'C453');

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

Practicle 8:

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:

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 ≥ 19860401 and Day ≤ 19860430 ;

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;

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

Practicle 9:

**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.**

Tables :

1) create table deposit(actno varchar2(5),cname varchar2(18),bname varchar2(18),amount number(8,2),adate

date);

2) create table branch(bname varchar2(18),city varchar2(18));

3) create table customer(cname varchar2(18), city varchar2(18));

4) create table borrow(loanno varchar2(5),cname varchar2(18),bname varchar2(18),amount number(8,2));

DEPOSIT TABLE:

1) insert into deposit values('100','Anil','wakad',1000.00,'1-mar-95');

2) insert into deposit values('101','Sunil','Pimpri',5000.00,'4-jan-96');

3) insert into deposit values('102','Aniket','KAROLBAGH',3500.00,'17-nov-95');

4) insert into deposit values('104','Krishna','bahgya nagar',1200.00,'17-dec-95');

5) insert into deposit values('105','Sanket','nehru nagar',3000.00,'27-mar-96');

6) insert into deposit values('106','Anuj','shivaji nagar',2000.00,'31-mar-96');

7) insert into deposit values('107','Shriram','Karvenagar',1000.00,'5-sep-95');

8) insert into deposit values('108','Bhosle','chinchwad',5000.00,'2-jul-95');

9) insert into deposit values('109','Anuj','akhurdi',7000.00,'10-aug-95');

BRANCH TABLE:

1) insert into branch values('wakad','Pune');

2) insert into branch values('Pimpri','Nagpur');

3) insert into branch values('KAROLBAGH','Delhi');

4) insert into branch values('bahgya nagar','Delhi');

5) insert into branch values('nehru nagar','Nagpur');

6) insert into branch values('shivaji nagar','Solapur');

7) insert into branch values('Karvenagar','Bombay');

8) insert into branch values('chinchwad','Umarkhed');

9) insert into branch values('akhurdi,'Delhi');

10) insert into branch values('POWAI','Bombay');

CUSTOMER TABLE:

1) insert into customer values('Anil','Calcutta');

2) insert into customer values('Sunil','Delhi');

3) insert into customer values('Aniket','Pune');

4) insert into customer values('Krishna','Nanded');

5) insert into customer values('Sanket','Solapur');

6) insert into customer values('Anuj','Nagpur');

7) insert into customer values('Shriram','Surat');

8) insert into customer values('Bhosle','Bombay');

9) insert into customer values('Om','Bombay');

10) insert into customer values('Deva','Nashik');

BORROW TABLE:

1) insert into borrow values('201','Anil','Calcutta',1000);

2) insert into borrow values('206','Shriram','Nashik',5000);

3) insert into borrow values('311','Sunil','DHARAMPETH',3000);

4) insert into borrow values('321','Sanket','Solapur',2000);

5) insert into borrow values('375','Krishna','Nanded',8000);

6) insert into borrow values('481','Bhosle','Mumbai',3000);

Insert data into the above created tables.

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 actno,amount from deposit where adate between '01-DEC-95' and '01-MAY-96';

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 account group by branch-name having 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 [Customers Detail] AS SELECT cname,bname FROM deposit WHERE actno = 101 ;

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

**Practicle 10:**

**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.**

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

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

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

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

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

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

**g. Create View on Borrow table.**

Insert data into the above created tables.

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

-> select bname 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

->

d. Find the number of depositors in the bank

-> select count (distinct customer-name)

from depositor

e. Calculate Min,Max amount of customers.

->

f. Create an index on deposit table

-> CREATE INDEX index\_customer\_branch ON deposit (cname,bname);

g. Create View on Borrow table.

-> CREATE VIEW [Customers Detail] AS SELECT cname, loanno , bname FROM borrow WHERE bname = 'ANDHERI';

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

Practicle 11:

**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.**

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

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

**c. Display names of customers living in city pune.**

**d. Display name of the city where branch KAROLBAGH is located.**

**e. Find the number of tuples in the customer relation**

**f. Delete all the record of customers Sunil**

**g. Create a view on deposit table.**

a. Display account date of customers Anil.

-> select adate from deposit where cname='Anil';

b. Modify the size of attribute of amount in deposit

-> alter table deposit modify amount number(10,2);

c. Display names of customers living in city pune.

-> select cname from customer 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 deposit where cname='Sunil';

g. Create a view on deposit table.

CREATE VIEW [KAROLBAGH Branch] AS SELECT cname FROM deposit WHERE bname = 'KAROLBAGH';

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

Practicle 12:

**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**

**1. Display customer name having living city Bombay and branch city Nagpur**

**2. Display customer name having same living city as their branch city**

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

**living city Nagpur.**

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

**amount greater than 2000**

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

**located.**

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

create Database DBMS\_Lab;

use DBMS\_Lab;

Create table Deposit(actno int,cname varchar(20),bname varchar(20),amount int,adate date);

Create table Branch(bname varchar(20),city varchar(20));

create table Customers(cname varchar(20),city varchar(20))

create table Borrow(loanno int,cname varchar(20),bname varchar(20),amount int);

1.for first question insert this info to respective table.

insert into Customers values("Krishna","Bombay");

insert into Branch values("Kothrud","Nagpur");

Ans:SELECT cname from Customers Inner join Branch On Customers.city="Bombay" AND Branch.city="Nagpur";

output:

+---------+

| cname |

+---------+

| Krishna |

+---------+

2.for second question

insert into Customers values("Darshan","Pune");

insert into Customers values("Kartik","Pimpri");

Ans:SELECT cname from Customers Inner join Branch On Customers.city=Branch.city;

select \*from Customers;

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

| cname | city |

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

| Darshan | Pune |

| Kartik | Pimpri |

| Krishna | Bombay |

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

3 rows in set (0.00 sec)

mysql> Select \*from Branch;

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

| bname | city |

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

| Pune | Pune |

| Pimpri | Pimpri |

| Kothrud | Nagpur |

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

3.

select \*from Deposit;

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

| actno | cname | bname | amount | adate |

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

| 100 | Krishna | Pune | 2500 | 2022-10-11 |

| 101 | Aniket | Pimpri | 4200 | 2022-10-01 |

| 102 | Sanket | Pune | 4200 | 2022-10-01 |

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

Select \*from Customers;

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

| cname | city |

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

| Darshan | Pune |

| Kartik | Pimpri |

| Krishna | Bombay |

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

Ans:

select Customers.cname from Customers Inner join Borrow On Borrow.cname=Customers.cname Inner Join Deposit On Borrow.bname=Deposit.bname;

4.Ans:Select Borrow.cname from Borrow Inner join Deposit ON Borrow.bname=Deposit.bname where Borrow.amount > 2000 AND Deposit.amount > 1000;

5.Ans: Select Customers.cname from Customers Inner join Deposit On Customers.cname=Deposit.cname Inner join Branch On Deposit.bname=Branch.bname AND Deposit.cname="Aniket";

+--------+

| cname |

+--------+

| Aniket |

+--------+

6.Ans:Create INDEX indx ON Deposit(actno,cname,bname,amount,adate);

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

Practicle 13:

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**

**1. Retrieve city, phone, url of author whose name is ‘CHETAN BHAGAT’.**

**2. Retrieve book title, reviewable id and rating of all books.**

**3. Retrieve book title, price, author name and url for publishers ‘MEHTA’.**

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

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

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

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

**‘C’.**

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

Create table PUBLISHER(PID int,PNAME varchar(20),ADDRESS varchar(20),STATE varchar(20),PHONE varchar(20),EMAILID varchar(20));

Query OK, 0 rows affected (0.03 sec)

mysql> Create table BOOK(ISBN int,BOOK\_TITLE varchar(20),CATEGORY varchar(20),PRICE int,COPYRIGHT\_DATE date,YEAR varchar(20),PAGE\_COUNT int,PID int);

Query OK, 0 rows affected (0.01 sec)

mysql> Create table AUTHOR(AID int,ANAME varchar(20),STATE varchar(20),CITY varchar(20),ZIP int,PHONE BIGINT,URL varchar(20));

Query OK, 0 rows affected (0.02 sec)

mysql> Create table AUTHOR\_BOOK(AID int,ISBN int);

Query OK, 0 rows affected (0.02 sec)

mysql> Create table REVIEW (RID int,ISBN int,RATING int);

Query OK, 0 rows affected (0.02 sec)

mysql> insert into AUTHOR values(1,"CHETAN BHAGAT","DELHI","DELHI",411,9833928209,"www.chetan.com");

Query OK, 1 row affected (0.01 sec)

mysql> 1.Ans: Select CITY,PHONE,URL from AUTHOR WHERE ANAME="CHETAN BHAGAT";

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

| CITY | PHONE | URL |

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

| DELHI | 9833928209 | [www.chetan.com](file:///C:\Users\91904\OneDrive\Desktop\www.chetan.com) |

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

1 row in set (0.00 sec)

2.Select BOOK.BOOK\_TITLE,REVIEW.RID from BOOK INNER JOIN REVIEW ON BOOK.ISBN=REVIEW.ISBN;

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

| BOOK\_TITLE | RID |

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

| DSA | 13 |

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

3.SELECT BOOK.BOOK\_TITLE,BOOK.PRICE,AUTHOR.ANAME,AUTHOR.URL FROM BOOK INNER JOIN PUBLISHER On AUTHOR.STATE=PUBLISHER.STATE;

4.UPDATE PUBLISHER SET PHONE=123456 WHERE PNAME="MEHTA";

5.1.SELECT avg(BOOK.PRICE) from BOOK INNER JOIN PUBLISHER ON PUBLISHER.PID=BOOK.PID;

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

| avg(BOOK.PRICE) |

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

| 450.0000 |

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

2. SELECT MAX(BOOK.PRICE) from BOOK INNER JOIN PUBLISHER ON PUBLISHER.PID=BOOK.PID;

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

| MAX(BOOK.PRICE) |

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

| 500 |

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

3.SELECT MIN(BOOK.PRICE) from BOOK INNER JOIN PUBLISHER ON PUBLISHER.PID=BOOK.PID;

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

| MIN(BOOK.PRICE) |

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

| 400 |

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

6.DELETE FROM BOOK WHERE PAGE\_COUNT < 100;

7. SELECT \*from AUTHOR WHERE CITY="PUNE" AND ANAME LIKE "C%";

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

| AID | ANAME | STATE | CITY | ZIP | PHONE | URL |

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

| 10 | Chaman | MH | PUNE | 41101 | 1733 | [www.sql.in](file:///C:\Users\91904\OneDrive\Desktop\www.sql.in) |

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

8.SELECT \*from AUTHOR WHERE CITY="Kurth";

9.

Ans:

CREATE PROCEDURE updatePage(ISBN int)

-> BEGIN

-> UPDATE BOOK SET PAGE\_COUNT=100;

-> end

-> //

insert into BOOK values(12,"DSA","SPECIAL",123,"2022-02-23",2022,23,89);

-> //

Query OK, 1 row affected (0.00 sec)

mysql> call updatePage(12);

-> //

Query OK, 1 row affected (0.01 sec)

mysql> select \*from BOOK;

-> //

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

| ISBN | BOOK\_TITLE | CATEGORY | PRICE | COPYRIGHT\_DATE | YEAR | PAGE\_COUNT | PID |

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

| 12 | DSA | SPECIAL | 123 | 2022-02-23 | 2022 | 100 | 89 |

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

10 CREATE FUNCTION retrievePrice(ISBN int)

-> BEGIN

-> DECLARE

-> pg\_cnt int;

-> select pg\_cnt into PRICE from BOOK where BOOK.ISBN=ISBN;

-> RETURN pg\_cnt;

-> end

-> //

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

Practicle 14:

14.

**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”**

**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**

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);

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;

14 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.

Answer:

create table account\_master(ID int primary key,Current\_balance int);

insert into account\_master values(1,10000),(2,5000),(3,60000);

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;

//

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

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

Practicle 15:A)

**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.**

create table client\_master (c\_id int, bal\_due int);

insert into client\_master values (1,-250), (2,200);

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 c\_id=uid;

if temp\_bal<0 then

signal sqlstate '45000'

set message\_text='BR violated';

else

select 'BR not violated' Message;

end if;

end

//

call check\_br(2);

//

call check\_br(1);

//

15.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.**

Delimiter //

create procedure Calculate\_fine(in roll int)

begin

declare fine\_amt int;

declare no\_of\_days int;

declare issue\_date date;

select DateofIssue into issue\_date from borrower where Roll\_no=roll;

select datediff(curdate(),issue\_date) into no\_of\_days;

if no\_of\_days>15 and no\_of\_days<=30 then

set fine\_amt=no\_of\_days\*5;

elseif no\_of\_days>30 then

set fine\_amt=(no\_of\_days-30)\*50+30\*5;

else

set fine\_amt=0;

end if;

insert into fine values(roll,curdate(),fine\_amt);

update borrower set Status='R' where Roll\_no=roll;

end;

//

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

Practicle 16 A)

**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)**

delimiter //

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

//

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

Practicle 16 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.**

**create table salary (emp\_id int, salary int);**

**insert into salary values (1,1000), (2,5000), (3,1500);**

**create table incr\_salary (id int, salary int);**

delimiter //

create procedure inc\_salary()

begin

declare temp\_salary int;

declare temp\_id int;

declare avg\_salary int;

declare exitcond boolean;

declare cur cursor for select emp\_id from salary;

declare cur2 cursor for select salary from salary;

declare continue handler for not found set exitcond=true;

select avg(salary) into avg\_salary from salary;

open cur;

open cur2;

l1:loop

fetch cur into temp\_id;

fetch cur2 into temp\_salary;

if(temp\_salary<avg\_salary) then

set temp\_salary=temp\_salary+temp\_salary\*0.1;

insert into incr\_salary values(temp\_id,temp\_salary);

end if;

if exitcond then

close cur;

close cur2;

leave l1;

end if;

end loop l1;

end

//

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

Q 17 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)**

create table inactive\_mstr(

acc\_no int primary key,

status varchar(30));

insert into inactive\_mstr values(101, 'active');

insert into inactive\_mstr values(102, 'inactive');

insert into inactive\_mstr values(103, 'inactive');

insert into inactive\_mstr values(104, 'active');

insert into inactive\_mstr values(105, 'inactive');

create table new\_active(

acc\_no int primary key,

new\_status varchar(30));

delimiter $

create procedure activate()

begin

declare accno int;

declare status2 varchar(30);

declare exitcondition boolean;

declare cur1 CURSOR for select acc\_no, status from inactive\_mstr;

declare continue handler for not found set exitcondition = TRUE;

open cur1;

L1: LOOP

fetch cur1 into accno, status2;

if status2 = 'inactive' then

insert into new\_active values(accno, status2);

end if;

update new\_active set new\_status = 'active';

if exitcondition then

leave L1;

end if;

end LOOP;

close cur1;

end;

$

Q17 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:**

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)

**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)**

create table Emp(

e\_no int Primary Key,

d\_no int ,

salary int);

create table dept\_salary(

d\_no int primary key,

Avg\_Salary int);

insert into Emp values (1,10,10000),(2,10,30000),(3,20,10400),(4,20,10000),(5,20,3000),(6,30,1500);

DELIMITER $

create procedure prod()

BEGIN

DECLARE TEMP\_no int;

DECLARE EXITCONDITION BOOLEAN;

DECLARE cur1 cursor for select d\_no from Emp;

DECLARE continue handler for not found set EXITCONDITION = TRUE;

open cur1;

L1:Loop

Fetch cur1 into TEMP\_no;

if not exists (Select \* from dept\_salary where d\_no=TEMP\_no) then

insert into dept\_salary select d\_no,avg(salary) FROM Emp group by(d\_no)

having( d\_no=Temp\_no) ;

END if;

IF EXITCONDITION THEN

close cur1;

LEAVE L1;

end if;

end Loop;

end;

$

call prod;

select \* from dept\_salary;

$

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

18 **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).**

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

//

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

18 **b) 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 Emp(e\_no integer primary key not null, e\_name text, salary integer);

create table tracking(e\_no integer, salary integer);

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 e\_no = new.e\_no, salary = new.salary;

end;

//

insert into Emp values(2001, 'sushant',40000)//;

ERROR 1644 (45000): Rejected!!!

insert into Emp values(2002, 'shweta',55000)//;

Query OK, 1 row affected (0.01 sec)

insert into Emp values(2004, 'sriya',54000)//;

Query OK, 1 row affected (0.01 sec)

insert into Emp values(2006, 'kim',15000)//;

ERROR 1644 (45000): Rejected!!!

select \* from Emp;

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

19.

**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**

**2. Find the information about all teachers of computer department**

**3. Find the information about all teachers of computer,IT,and e&TC department**

**4. Find the information about all teachers of computer,IT,and E&TC department having**

**salary greate than or equl to 10000/-**

**5. Find the student information having roll\_no = 2 or Sname=xyz**

**6. Update the experience of teacher-praveen to 10years, if the entry is not available in**

**database consider the entry as new entry.**

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

**8. find the teachers name and their experience from teachers collection**

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

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

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

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

**order**

db.teachers.find().pretty();

db.students.insertMany([{

... 'sname': 'krishna',

... 'roll\_no': 71 ,

... 'class': 'TCOD' ,

...

... },

... {

... 'sname': 'jayesh',

... 'roll\_no': 77 ,

... 'class': 'TCOD'

... },

... {

... 'sname': 'rupesh',

... 'roll\_no': 32 ,

... 'class': 'TCOD'

...

... },

... {

... 'sname': 'akash',

... 'roll\_no': 73 ,

... 'class': 'TCOD'

... },

... {

... 'sname': 'dinesh',

... 'roll\_no': 79 ,

... 'class': 'TCOC'

... }

... ]);

db.teachers.find({dname : 'COMP'}).pretty();

db.teachers.find({dname : 'COMP'}, {dname : 'ENTC'}, {dname :'IT' }).pretty();

db.teachers.find({salary :{$gt : 10000} }).pretty();

db.students.find({roll\_no : 71}).pretty();

db.teachers.insertOne( { 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 5, 'salary': 9200, 'date\_of\_joining': ISODate("2012-10-11T06:31:15.000Z") } );

db.teachers.updateOne({tname : 'praveen'}, { $set : {experience: 10} });

db.teachers.updateMany({dname : 'IT'}, { $set : {dname: 'COMP'} });

db.teachers.deleteMany({dname : 'IT'});

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

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

Q 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**

**4. Find the information about all teachers of computer department**

**5. Find the information about all teachers of computer,IT,and e&TC department**

**6. Find the information about all teachers of computer,IT,and E&TC department having salary**

**greate than or equl to 25000/-**

**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.**

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

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

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

**13. Delete all the doccuments from teachers collection having IT dept.**

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

1.Create Database DYPIT

use DYPIT;

2. Create following Collections

Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )

Students(Sname,roll\_no,class)

db.createCollection('teachers');

db.createCollection('students');

db.teachers.insertOne({ 'tname': 'Rahul',

'dno': 12, 'dname': '',

'experience': 5, 'salary':76000,

'date\_of\_joining': ISODate("2017-11-10T06:31:15.000Z") });

db.teachers.insertMany([

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 5, 'salary': 23000, 'date\_of\_joining': ISODate("2017-10-12T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 4, 'salary': 92000, 'date\_of\_joining': ISODate("2018-11-21T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 2, 'salary': 42000, 'date\_of\_joining': ISODate("2020-10-02T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 6, 'salary': 25000, 'date\_of\_joining': ISODate("2016-12-22T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 2, 'salary': 54000, 'date\_of\_joining': ISODate("2020-09-17T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 1, 'salary': 56000, 'date\_of\_joining': ISODate("2021-06-13T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 9, 'salary': 30000, 'date\_of\_joining': ISODate("2013-03-23T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 7, 'salary': 60000, 'date\_of\_joining': ISODate("2015-03-14T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 3, 'salary': 30000, 'date\_of\_joining': ISODate("2019-11-10T06:31:15.000Z") }

] );

db.teachers.find().pretty();

db.students.insertMany([{

'sname': 'krishna',

'roll\_no': 71 ,

'class': 'TCOD' ,

},

{

'sname': 'jayesh',

'roll\_no': 77 ,

'class': 'TCOD'

},

{

'sname': 'rupesh',

'roll\_no': 32 ,

'class': 'TCOD'

},

{

'sname': 'akash',

'roll\_no': 73 ,

'class': 'TCOD'

},

{

'sname': 'dinesh',

'roll\_no': 79 ,

'class': 'TCOC'

}

]);

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 : 'COMP'}).pretty();

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

db.teachers.find( { dname: { $in: [ 'COMP','IT','ENTC'] } } );

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 : {$gte : 2500} , dname : {$in :['IT','COMP', 'ENTC']}});

7. Find the student information having roll\_no = 25 or Sname=xyz

db.students.find( { roll\_no : 71 } ).pretty();

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

db.teachers.updateMany( {tname : 'praveen'}, {$set : { experience : 10}});

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

db.teachers.updateMany (

{dname : 'IT'}, {$set : {dname : 'COMP'}}

);

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

db.teachers.find( {}, {tname :1, experience : 1});

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

//this is the syntax of the save method but this method is deprecated

db.teachers.save( { tname : "krishna", dno : 4 } )

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

db.teachers.deleteMany({dname : 'IT'});

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

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

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

Q 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**

**2. Find the average salary teachers of computer department**

**3. Find the minimum and maximum salary of e&TC department teachers**

**4. Find the information about all teachers of computer,IT,and E&TC department having**

**salary greate than or equl to 10000/-**

**5. Find the student information having roll\_no = 2 or Sname=xyz**

**6. Update the experience of teacher-praveen to 10years, if the entry is not available in**

**database consider the entry as new entry.**

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

**8. find the teachers name and their experience from teachers collection**

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

**10. Find the total salary all teachers.**

db.createCollection('teachers');

db.createCollection('students');

db.teachers.insertOne({ 'tname': 'Rahul',

'dno': 12, 'dname': '',

'experience': 5, 'salary':76000,

'date\_of\_joining': ISODate("2017-11-10T06:31:15.000Z") });

db.teachers.insertMany([

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 5, 'salary': 23000, 'date\_of\_joining': ISODate("2017-10-12T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 4, 'salary': 92000, 'date\_of\_joining': ISODate("2018-11-21T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 1, 'dname': 'COMP', 'experience': 2, 'salary': 42000, 'date\_of\_joining': ISODate("2020-10-02T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 6, 'salary': 25000, 'date\_of\_joining': ISODate("2016-12-22T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 2, 'salary': 54000, 'date\_of\_joining': ISODate("2020-09-17T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 2, 'dname': 'ENTC', 'experience': 1, 'salary': 56000, 'date\_of\_joining': ISODate("2021-06-13T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 9, 'salary': 30000, 'date\_of\_joining': ISODate("2013-03-23T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 7, 'salary': 60000, 'date\_of\_joining': ISODate("2015-03-14T06:31:15.000Z") },

{ 'tname': 'praveen', 'dno': 3, 'dname': 'IT', 'experience': 3, 'salary': 30000, 'date\_of\_joining': ISODate("2019-11-10T06:31:15.000Z") }

] );

db.teachers.find().pretty();

db.students.insertMany([{

'sname': 'krishna',

'roll\_no': 71 ,

'class': 'TCOD' ,

},

{

'sname': 'jayesh',

'roll\_no': 77 ,

'class': 'TCOD'

},

{

'sname': 'rupesh',

'roll\_no': 32 ,

'class': 'TCOD'

},

{

'sname': 'akash',

'roll\_no': 73 ,

'class': 'TCOD'

},

{

'sname': 'dinesh',

'roll\_no': 79 ,

'class': 'TCOC'

}

]);

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: 'COMP' }

},

{

$group: { \_id: "$name", AvgSalary: { $avg: "$salary" } }

}

] );

3.Find the minimum and maximum salary of e&TC department teachers

//minimum salary

db.teachers.aggregate( [

{

$match: { dname: 'ENTC' }

},

{

$group: { \_id: "$name", MinSalary: { $min: "$salary" } }

}

] )

//for maximum salary

db.teachers.aggregate( [

{

$match: { dname: 'ENTC' }

},

{

$group: { \_id: "$name", MaxSalary: { $max: "$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 : {$gte : 10000} , dname : {$in :['IT','COMP', 'ENTC']}})

5. Find the student information having roll\_no = 2 or Sname=xyz

db.students.find( { roll\_no : 2 } ).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.updateMany( {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 : 'COMP'}}

);

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

db.teachers.find( {}, {tname :1, experience : 1});

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

//this is the syntax of the save method but this method is deprecated

db.teachers.save( { tname : "krishna", dno : 4 } );

10. Find the total salary all teachers.

db.teachers.aggregate(

[

{ $group: { \_id: null, total: { $sum: "$salary" } } }

]

);

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

Q22

**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**

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

**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.**

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

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

**7. Show all the indexes created in the database DYPIT**

**8. Show all the indexes created in above collections.**

db.createCollection('teachers');

db.createCollection('students');

db.teachers.insertOne({ 'tname': 'Rahul',

'dno': 12, 'dname': '',

'experience': 5, 'salary':76000,

'date\_of\_joining': ISODate("2017-11-10T06:31:15.000Z") });

db.teachers.insertMany([

{ 'tname': 'patil', 'dno': 1, 'dname': 'COMP', 'experience': 5, 'salary': 500, 'date\_of\_joining': ISODate("2017-10-12T06:31:15.000Z") },

{ 'tname': 'jadhav', 'dno': 1, 'dname': 'COMP', 'experience': 4, 'salary': 300, 'date\_of\_joining': ISODate("2018-11-21T06:31:15.000Z") },

{ 'tname': 'chobe', 'dno': 1, 'dname': 'COMP', 'experience': 2, 'salary': 300, 'date\_of\_joining': ISODate("2020-10-02T06:31:15.000Z") },

{ 'tname': 'sakunde', 'dno': 2, 'dname': 'ENTC', 'experience': 6, 'salary': 900, 'date\_of\_joining': ISODate("2016-12-22T06:31:15.000Z") },

{ 'tname': 'bhosle', 'dno': 2, 'dname': 'ENTC', 'experience': 2, 'salary': 100, 'date\_of\_joining': ISODate("2020-09-17T06:31:15.000Z") },

{ 'tname': 'reddy', 'dno': 2, 'dname': 'ENTC', 'experience': 1, 'salary': 90, 'date\_of\_joining': ISODate("2021-06-13T06:31:15.000Z") },

{ 'tname': 'patel', 'dno': 3, 'dname': 'IT', 'experience': 9, 'salary': 100, 'date\_of\_joining': ISODate("2013-03-23T06:31:15.000Z") },

{ 'tname': 'desle', 'dno': 3, 'dname': 'IT', 'experience': 7, 'salary': 50, 'date\_of\_joining': ISODate("2015-03-14T06:31:15.000Z") },

{ 'tname': 'chavan', 'dno': 3, 'dname': 'IT', 'experience': 3, 'salary': 30, 'date\_of\_joining': ISODate("2019-11-10T06:31:15.000Z") }

] );

db.teachers.find().pretty();

db.students.insertMany([{

'sname': 'krishna',

'roll\_no': 71 ,

'class': 'TCOD' ,

},

{

'sname': 'jayesh',

'roll\_no': 77 ,

'class': 'TCOD'

},

{

'sname': 'rupesh',

'roll\_no': 32 ,

'class': 'TCOD'

},

{

'sname': 'akash',

'roll\_no': 73 ,

'class': 'TCOD'

},

{

'sname': 'dinesh',

'roll\_no': 79 ,

'class': 'TCOC'

}

]);

1. Display the department wise average salary

db.teachers.aggregate([

{ $match : {dname : {$in : ['IT', 'COMP', 'ENTC']}} },

{ $group : {\_id :'$dname', AverageSalary : {$avg : '$salary'} } }

]);

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

db.teachers.aggregate([

{ $match: { dname: { $in: ['IT', 'COMP', 'ENTC'] } } },

{ $group: { \_id: '$dname', deptCount:{ $sum: 1} } }

]);

3. Display the department wise total salary of departments having total salary greater than

or equals to 50000/-

db.teachers.aggregate([

{ $match: { dname: { $in: ['IT', 'COMP', 'ENTC'] }, salary : {$gte : 50000 } }},

{ $group: { \_id: '$dname', totalSalary:{ $sum: '$salary'} } }

]);

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

db.teachers.aggregate([

{$group : {\_id : 'dname' , minsalary : {$min : '$salary'}, maxSalary : {$max : '$salary'}}}

]);

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

db.teachers.createIndex( { 'index': 1 }, {expireAfterSeconds: 3600 });

db.teachers.createIndex( { 'ind': 3 });

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

db.teachers.createIndex( { 'krishna': 1, 'shriman': 1 });

7. Show all the indexes created in the database DYPIT

db.getCollectionNames().forEach(function(collection) {

indexes = db.getCollection(collection).getIndexes();

print("Indexes on " + collection + ":");

printjson(indexes);

});

8. Show all the indexes created in above collections.

db.teachers.getIndexes();

db.teachers.dropIndexes();

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

Q 23

**Create index and fire queries with MongoDB**

**1. Import zip.json.**

**2. Create single field, composite and multikey indexes.**

**3. Fire queries given below again and write your analysis.**

**1. Display all cities having population above 1600.**

**2. Display all cities in state “KS”.**

**3. Display location of city "TIMKEN"**

1. Import zip.json.

mongoimport --db <databaseName> --collection <collectionName> --file <filePath/jsonFile.json>

2. Create single field, composite and multikey indexes.

single -> db.<collectionName>.createIndex( {'ind' : 1 });

multikey -> db.<collectionName>.createIndex( {'index' : 2, 'name' : 4 });

composite -> db.<collectionName>.createIndex( {'index' : 2, 'name' : 4 },{'indw' : 3 });

3. Fire queries given below again and write your analysis.

1. Display all cities having population above 1600.

db.<collectionName>.find( { population : {$gt : 1600} });

2. Display all cities in state “KS”.

db.<collectionName>.find( {state : 'KS'}, {city :1});

3. Display location of city "TIMKEN"

db.<collectionName>.find( {city : 'TIMKEN'}, {location :1});

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

Q 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)**

**3. Write a query that returns all the games**

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

**5. Update your two favourite games to have two achievements called ‘Game**

**Master’ and ‘Speed Demon’.**

**6. Write a query that returns all the games that have both the ‘Game Maser’ and**

**7. the ‘Speed Demon’ achievements.**

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

1. Create a collection called ‘games’.

db.createCollection('games');

2. Add 5 games to the database. Give each document the following properties:

name, gametype, rating (out of 100)

db.games.insertMany([

... {name : 'temple run', gameType :'racing', rating : 70 },

... {name : 'freefire', gameType :'fighting', rating : 89 },

... {name : 'bull run', gameType :'racing', rating : 90 },

... {name : 'pubg', gameType :'fighting', rating : 80 },

... {name : 'red run', gameType :'racing', rating : 60 }

... ]);

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().pretty().limit(3);

5. Update your two favourite games to have two achievements called ‘Game

Master’ and ‘Speed Demon’.

db.games.updateMany(

{gameType : 'racing'}, {$set : {achivements : ['Game master','Speed Demon']}}

);

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

db.games.find( {achivements : ['Game master', 'Speed Demon'] });

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

db.games.find( { }, {achivements :true});

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

Q 25

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

**{**

**“id” : 0,**

**“name” : “Leanne Flinn”,**

**“email” : “leanne.flinn@unilogic.com”,**

**“work” :”Unilogic” ,**

**“age” :27**

**“gender” :”Male”**

**“Salary” :16660**

**“hobbies”:”Acrobatics,Photography,Papier-Mache”**

**}**

**1. Get the count of Males and Females**

**2. Count the number of users in each hobby**

States> db.gen.mapReduce(function () {emit(this.g, this.\_id);}, function (key, values) {return values.length }, {out: "count" })

{ result: 'count', ok: 1 }

States> db.count.find()

[ { \_id: 'F', value: 2 }, { \_id: 'M', value: 3 } ]

**test> use details**

**switched to db details**

**details> db.createCollection("records")**

**{ ok: 1 }**

**details> db.records.insert({ id:1,name:"Yash",email:"yash@123",work:"unilogic",age:27,gender:"male",salary:14444,hibbies:"Cricket"})**

**DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("637258c9deaac56a01501cf2") }**

**}**

**details> db.records.insert({ id:1,name:"Sarvesh",email:"sarv@123",work:"logic",age:21,gender:"male",salary:14444,hibbies:"Cricket"})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("637258f8deaac56a01501cf3") }**

**}**

**details> db.records.insert({ id:3,name:"Shruti",email:"Shruti@123",work:"logic",age:21,gender:"female",salary:14444,hibbies:"Gaming"})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("6372592adeaac56a01501cf4") }**

**}**

**details> db.records.insert({ id:4,name:"Snehal",email:"Snehal@123",work:"ulogic",age:21,gender:"female",salary:14444,hibbies:"Gaming"})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63725948deaac56a01501cf5") }**

**}**

**details> db.records.mapReduce(**

**... function()**

**... { emit(this.gender,1);},**

**... function(key,values)**

**... {return Array.sum(values)}, {out:"genders"}**

**... )**

**DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.**

**See** [**https://docs.mongodb.com/manual/core/map-reduce**](https://docs.mongodb.com/manual/core/map-reduce) **for details.**

**{ result: 'genders', ok: 1 }**

**details> db.genders.find()**

**[ { \_id: 'male', value: 2 }, { \_id: 'female', value: 2 } ]**

**details>**

**details>**

**details> db.records.mapReduce(**

**... function()**

**... {emit(this.hibbies,1);},**

**... function(key,values)**

**... {**

**... return Array.sum(values)},**

**... {out:"hobbie"})**

**{ result: 'hobbie', ok: 1 }**

**details> db.hobbie.find()**

**[ { \_id: 'Cricket', value: 2 }, { \_id: 'Gaming', value: 2 } ]**

**details>**

**======================================================================================**

**Q 26**

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

**1. Import zip.json.**

**2. Find total population in each state.**

1. Import zip.json.

mongoimport --db <databaseName> --collection <collectionName> --file <filePath/jsonFile.json>

2. Find total population in each state.

db.teachers.aggregate([

... { $match: { state: { $in: ['state1', 'state2', 'state2'] } } },

... { $group: { \_id: '$state', totalPopulation:{ $sum: '$population'} } }

... ]);

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

Q.27

27.Create a database called ‘library’, create a collection called ‘books’.find the number of books

having pages less 250 pages and consider ad small book and greater than 250 consider as Big

book using Map Reduce function.

**test> use details**

**switched to db details**

**details> db.createCollection("records")**

**{ ok: 1 }**

**details> db.records.insert({ id:1,name:"Yash",email:"yash@123",work:"unilogic",age:27,gender:"male",salary:14444,hibbies:"Cricket"})**

**DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("637258c9deaac56a01501cf2") }**

**}**

**details> db.records.insert({ id:1,name:"Sarvesh",email:"sarv@123",work:"logic",age:21,gender:"male",salary:14444,hibbies:"Cricket"})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("637258f8deaac56a01501cf3") }**

**}**

**details> db.records.insert({ id:3,name:"Shruti",email:"Shruti@123",work:"logic",age:21,gender:"female",salary:14444,hibbies:"Gaming"})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("6372592adeaac56a01501cf4") }**

**}**

**details> db.records.insert({ id:4,name:"Snehal",email:"Snehal@123",work:"ulogic",age:21,gender:"female",salary:14444,hibbies:"Gaming"})**

**books> db.books.insert({name : "JAVA", pages : 100})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63726b3adeaac56a01501cf6") }**

**}**

**books> db.books.insert({name : "JavaScript", pages : 200})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63726b4bdeaac56a01501cf7") }**

**}**

**books> db.books.insert({name : "Unity3D", pages : 300})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63726b66deaac56a01501cf8") }**

**}**

**books> db.books.insert({name : "RDR2", pages : 400})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63726b75deaac56a01501cf9") }**

**}**

**books> db.books.insert({name : "Days Gone", pages : 150})**

**{**

**acknowledged: true,**

**insertedIds: { '0': ObjectId("63726b86deaac56a01501cfa") }**

**}**

**books> var mapfunc = function()**

**... {**

**... var category;**

**... if(this.pages >= 250)**

**... category = "Big Books";**

**... else**

**... category = "Small Books";**

**... emit(category,{name:this.name});**

**... };**

**books> var redFunc = function(key,values)**

**... {**

**... var sum=0;**

**... values.forEach(function(doc) {**

**... sum += 1;**

**... });**

**... return {books:sum};**

**... };**

**books> db.books.mapReduce(mapfunc, redFunc, {out:"book\_results"});**

**{ result: 'book\_results', ok: 1 }**

**books> db.book\_results.find();**

**[**

**{ \_id: 'Big Books', value: { books: 2 } },**

**{ \_id: 'Small Books', value: { books: 3 } }**

**]**

**books>**