**All codes-**

**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 %.

create table Emp(

eno integer primary key,

ename varchar(20),

job varchar(20),

hiredate date,

salary integer,

commission integer,

deptno integer

);

insert into Emp values(1, "Samiksha", 'Manager','2020-12-02' , 76540, 3476, 10);

insert into Emp values(2, "Pranita", 'Engineer','1975-09-24' , 84350, 1240, 12);

insert into Emp values(3, "Komal", 'Salesman','2009-02-04' , 35948, 4716, 10);

insert into Emp values(4, "Sakshi", 'Manager','2014-12-06' , 65487, 1240, 13);

insert into Emp values(5, "Ishita", 'Salesman','1980-02-24' , 54840, 3120, 20);

insert into Emp values(6, "Isha", 'developer','2021-02-24' , 43840, 1320, 20);

select\*from Emp;

create table dept(

deptno integer primary key,

deptname varchar(20),

location varchar(20)

);

insert into dept values(10, "Computer", "Nashik");

insert into dept values(12, "IT", "Pune");

insert into dept values(13, "Dev", "Mumbai");

insert into dept values(20, "AIDS", "Pune");

select \* from dept;

-- 1. List the maximum salary paid to salesman

select max(salary) as maxprice from Emp where job="Salesman";

-- 2. List name of emp whose name start with ‘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<'1981-09-30';

-- 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(\*), 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), 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;

-- 8. List total salary paid to each department

select D.deptname, sum(E.salary) from Emp E, dept D where D.deptno=E.deptno group by deptname;

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

select \* from Emp E, dept D where D.deptno=E.deptno and D.deptname="Dev";

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

update Emp E, dept D set salary=salary+(salary\*5/100) where D.deptno=10;

select \* from Emp;

# 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)

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

* + 1. Find the names of all employees who work for First Bank Corporation.
    2. Find all employees who do not work for First Bank Coorporation
    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.
    6. Update salary of all employees who work for First Bank Corporation by 10%.
    7. Find employee and their managers.
    8. Find the names, street and cities of all employees who work for First Bank Corporation and earn more than 10,000.
    9. Find those companies whose employees earn a higher salary,on average, than th average salary at First Bank Corporation

create table employee(

ename varchar(20) primary key,

street varchar(20),

city varchar(20)

);

insert into employee values("Samiksha", "s1", "Nashik");

insert into employee values("Pranita", "s2", "Mumbai");

insert into employee values("Sakshi", "s3", "Pune");

insert into employee values("Komal", "s4", "Nashik");

select \* from employee;

create table works(

ename varchar(20) primary key,

cname varchar(20),

salary integer not null

);

insert into works values("Samiksha", "c1", 54420);

insert into works values("Pranita", "c2", 35770);

insert into works values("Sakshi", "c3", 24150);

insert into works values("Komal", "c1", 78120);

select \* from works;

create table company(

cname varchar(20) primary key,

city varchar(20)

);

insert into company values("c1", "Nashik");

insert into company values("c2", "Pune");

insert into company values("c3", "Mumbai");

select \* from company;

create table manages(

ename varchar(20),

mname varchar(20)

);

insert into manages values("Samiksha", "M1");

insert into manages values("Sakshi", "M2");

insert into manages values("Komal", "M3");

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

select E.ename from employee E, works W, company C where C.cname=W.cname and W.ename=E.ename and C.cname="c1";

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

select E.ename from employee E, works W, company C where C.cname=W.cname and W.ename=E.ename and C.cname!="c1";

-- 3. Find the company that has most employees.

SELECT cname,count(\*) as cnt FROM works GROUP BY cname ORDER BY cnt DESC limit 1;

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

SELECT cname,city FROM company WHERE city IN(SELECT city FROM company WHERE cname='c2');

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

select \* from employee E, works W having salary > 10000 and E.ename=W.ename;

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

update works set salary=salary+(salary\*10/100) where cname="c1";

select \* from works;

-- 7. Find employee and their managers.

select ename, mname 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.ename, E.street, E.city from employee E, works W where W.cname="c1" and W.salary>10000 and E.ename=W.ename;

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

select cname from works group by cname having avg (salary) > (select avg (salary) from works where cname = "c1");

1. **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)

# Solve following queries by SQL

* 1. List full details of all hotels.
  2. How many hotels are there?
  3. List the price and type of all rooms at the Grosvenor Hotel.
  4. List the number of rooms in each hotel.
  5. Update the price of all rooms by 5%.
  6. List full details of all hotels in London.
  7. What is the average price of a room?
  8. List all guests currently staying at the Grosvenor Hotel.
  9. List the number of rooms in each hotel in London. 10.Create one view on above database and query it.

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 type1 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 Room.Type,Room.Price from Room,Hotel where Hotel.Name='Grosvenor' and Room.HotelNo=Hotel.HotelNo;

**4. List the number of rooms in each hotel**

select Hotel.Name, count(RoomNo) as Roomcount from Room, Hotel where Hotel.City='London' and Room.HotelNo=Hotel.HotelNo group by Room.hotelno;

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

select Guest.GuestName from Guest,Hotel,Booking where Guest.GuestNo=Booking.GuestNo and Booking.HotelNo=Hotel.HotelNo and Hotel.Name='Grosvenor';

**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';

select Count(GuestNo) from Booking where DateFrom>'2022-08-01' and DateTo<'2022-08-31';

select Count(GuestNo) from Booking where DateFrom and DateTo between '2022-08-01' and '2022-08-31';

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

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

select MAX(type1) from Room where hotelno = '01';

select max(Type) from Room,Hotel where Hotel.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 Booking where dateto >= '2022/11/11';

**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.**

**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)**

# Find the SQL queries for the following:

1. Get the details of employees working on project C353
2. Get employee number of employees working on project C353
3. Obtain details of employees working on Database project
4. Get details of employees working on both C353 and C354
5. Get employee numbers of employees who do not work on project C453

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:

1. Get the duty allocation details for emp\_no 123461 for the month of April 1986.
2. Find the shift details for Employee ‘xyz’
3. Get employees whose rate of pay is more than or equal to the rate of pay of employee ‘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.
5. Find the names of employees who are assigned to all positions that require a Chef’s skill

6.Find the employees with the lowest pay rate

1. .Get the employee numbers of all employees working on at least two dates.
2. .Get a list of names of employees with the skill of Chef who are assigned a duty 9 .Get a list of employees not assigned a duty

10.Get a count of different employees on each shift

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.**

1. Display names of depositors having amount greater than 4000.
2. Display account date of customers Anil
3. Display account no. and deposit amount of customers having account opened between dates 1-12-96 and 1-5-97
4. Find the average account balance at the Perryridge branch.
5. Find the names of all branches where the average account balance is more than $1,200.
6. Delete depositors having deposit less than 5000
7. Create a view on deposit table.

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;

OR

mysql> create table Deposit( actno integer primary key, cname varchar(20), bname varchar(20), amount integer, adate date);

Query OK, 0 rows affected (0.28 sec)

mysql> create table Branch(bname varchar(20),city varchar(20));

Query OK, 0 rows affected (0.20 sec)

mysql> create table Customers(cname varchar(20),city varchar(20));

Query OK, 0 rows affected (0.21 sec)

mysql> create table Borrow(loanno integer primary key,cname varchar(20),bname varchar(20),amount integer);

Query OK, 0 rows affected (0.18 sec)

mysql> insert into Deposit values(101,'Anil','Balewadi',3500,'2000-08-11');

Query OK, 1 row affected (0.06 sec)

mysql> insert into Deposit values(102,'Sunil','Perryridge',5500,'1997-04-30');

Query OK, 1 row affected (0.10 sec)

mysql> insert into Deposit values(103,'Kunal','Aundh',7500,'1997-02-9');

Query OK, 1 row affected (0.08 sec)

mysql> insert into Deposit values(104,'Mrunal','Kasarwadi',2000,'1996-09-17');

Query OK, 1 row affected (0.10 sec)

mysql> insert into Branch values('Balewadi','Pune');

Query OK, 1 row affected (0.10 sec)

mysql> insert into Branch values('Perryridge','Mumbai');

Query OK, 1 row affected (0.09 sec)

mysql> insert into Branch values('Aundh','Pune');

Query OK, 1 row affected (0.11 sec)

mysql> insert into Branch values('Kasarwadi','Delhi');

Query OK, 1 row affected (0.07 sec)

mysql> insert into Customers values('Anil','Pune');

Query OK, 1 row affected (0.09 sec)

mysql> insert into Customers values('Sunil','Mumbai');

Query OK, 1 row affected (0.15 sec)

mysql> insert into Customers values('Kunal','Pune');

Query OK, 1 row affected (0.13 sec)

mysql> insert into Customers values('Mrunal','Delhi');

Query OK, 1 row affected (0.06 sec)

mysql> insert into Borrow values(1,'Pranita','Pune',20000);

Query OK, 1 row affected (0.11 sec)

mysql> insert into Borrow values(2,'Taksha','Perryridge',3000);

Query OK, 1 row affected (0.12 sec)

mysql> insert into Borrow values(2,'Taksha','Aundh',50000);

ERROR 1062 (23000): Duplicate entry '2' for key 'Borrow.PRIMARY'

mysql> insert into Borrow values(3,'jui','Aundh',50000);

Query OK, 1 row affected (0.10 sec)

mysql> delete from Borrow where loanno=1;

Query OK, 1 row affected (0.10 sec)

mysql> insert into Borrow values(1,'Pranita','Balewadi',20000);

Query OK, 1 row affected (0.08 sec)

mysql> insert into Borrow values(4,'Samiksha','Kasarwadi',1000);

Query OK, 1 row affected (0.02 sec)

mysql>

mysql> select \*from Deposit;

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

| actno | cname | bname | amount | adate |

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

| 101 | Anil | Balewadi | 3500 | 2000-08-11 |

| 102 | Sunil | Perryridge | 5500 | 1997-04-30 |

| 103 | Kunal | Aundh | 7500 | 1997-02-09 |

| 104 | Mrunal | Kasarwadi | 2000 | 1996-09-17 |

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

4 rows in set (0.00 sec)

mysql> select \*from Branch;

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

| bname | city |

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

| Balewadi | Pune |

| Perryridge | Mumbai |

| Aundh | Pune |

| Kasarwadi | Delhi |

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

4 rows in set (0.00 sec)

mysql> select \*from Customers;

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

| cname | city |

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

| Anil | Pune |

| Sunil | Mumbai |

| Kunal | Pune |

| Mrunal | Delhi |

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

4 rows in set (0.00 sec)

mysql> select \*from Borrow;

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

| loanno | cname | bname | amount |

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

| 1 | Pranita | Balewadi | 20000 |

| 2 | Taksha | Perryridge | 3000 |

| 3 | jui | Aundh | 50000 |

| 4 | Samiksha | Kasarwadi | 1000 |

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

1) mysql> select cname from Deposit where amount>4000;

+-------+

| cname |

+-------+

| Sunil |

| Kunal |

+-------+

2) mysql> select adate from Deposit where cname='Anil';

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

| adate |

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

| 2000-08-11 |

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

3) mysql> select actno,amount from Deposit where adate between '1996-12-1' and '1997-05-1';

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

| actno | amount |

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

| 102 | 5500 |

| 103 | 7500 |

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

4) mysql> select avg(amount) from Deposit where bname='Perryridge';

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

| avg(amount) |

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

| 5500.0000 |

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

5) mysql> select bname from Deposit group by bname having avg(amount)>1200;

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

| bname |

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

| Balewadi |

| Perryridge |

| Aundh |

| Kasarwadi |

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

6) mysql> delete from Deposit where amount<5000;

Query OK, 2 rows affected (0.17 sec)

mysql> select \* from Deposit;

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

| actno | cname | bname | amount | adate |

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

| 102 | Sunil | Perryridge | 5500 | 1997-04-30 |

| 103 | Kunal | Aundh | 7500 | 1997-02-09 |

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

7) mysql> create view D as select actno,cname,bname,amount,adate from Deposit;

Query OK, 0 rows affected (0.11 sec)

mysql> select \*from D;

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

| actno | cname | bname | amount | adate |

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

| 102 | Sunil | Perryridge | 5500 | 1997-04-30 |

| 103 | Kunal | Aundh | 7500 | 1997-02-09 |

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

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

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

Select \* from Branch where city=’Bombay’

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

Select actno, amount from deposit

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

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

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

select count (distinct cname) from deposit

1. **Calculate Min,Max amount of customers.**
2. **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;

OR

a) select bname from Branch where city='Bombay';

b) select actno,amount from Deposit;

c) update Customers set city='Mumbai' where cname='Anil';

d) select count(cname) from Deposit;

e)

f) create index I on Deposit (actno,cname,bname,amount,adate);

g) create view B as select loanno,cname,bname,amount from Borrow;

**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**

1. **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’ ;

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

select c.city from customer c, branch b where c.city=b.city ;

1. **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’

1. **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;

1. **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'));

1. **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';

1. **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;

1. **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.**

**PL/SQL**

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”

declare

--s\_roll students.roll%type;

s\_roll int;

s\_att students.attendance%type;

roll int;

myexception Exception;

begin

roll:=3;

select attendance into s\_att from students where roll=3;

if s\_att<100 then

raise myexception;

else

update students set status='P' where roll=3;

end if;

exception

when myexception then

dbms\_output.put\_line('Term NOT GRANTED');

end;

**OR**

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

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

select \* from stud;

mysql> delimiter //

mysql> 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;

-> //

mysql> delimiter ;

**OR**

CREATE TABLE stud1(roll\_no integer, attendance integer, status varchar(7));

insert into stud1 values(01,85,'NA');

insert into stud1 values(02,90,'NA');

insert into stud1 values(03,65,'NA');

insert into stud1 values(04,70,'NA');

insert into stud1 values(05,68,'NA');

insert into stud1 values(06,42,'NA');

insert into stud1 values(07,50,'NA');

insert into stud1 values(08,95,'NA');

SELECT \* FROM STUD1;

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

| roll\_no | attendance | status |

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

| 1 | 85 | NA |

| 2 | 90 | NA |

| 3 | 65 | NA |

| 4 | 70 | NA |

| 5 | 68 | NA |

| 6 | 42 | NA |

| 7 | 50 | NA |

| 8 | 95 | NA |

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

DELIMITER $

CREATE PROCEDURE ATTEND\_CAL(IN roll int)

begin

declare attend int;

select attendance into attend from stud1 where roll\_no = roll;

if attend<75 then

update stud1 set status ='ND' WHERE roll\_no=roll;

else

update stud1 set status ='D' where roll\_no=roll;

end if;

end;

$

CALL ATTEND\_CAL(01)$;

CALL ATTEND\_CAL(02)$;

CALL ATTEND\_CAL(03)$;

CALL ATTEND\_CAL(04)$;

CALL ATTEND\_CAL(05)$;

CALL ATTEND\_CAL(06)$;

CALL ATTEND\_CAL(07)$;

CALL ATTEND\_CAL(08)$;

SELECT \* FROM STUD1$;

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

| roll\_no | attendance | status |

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

| 1 | 85 | D |

| 2 | 90 | D |

| 3 | 65 | ND |

| 4 | 70 | ND |

| 5 | 68 | ND |

| 6 | 42 | ND |

| 7 | 50 | ND |

| 8 | 95 | D |

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

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.

create table acc\_master(id int,name varchar(20),balance number);

insert into acc\_master values(1,'MASTER',50000);

select \* from acc\_master;

declare

current number;

withdrawl number;

deposit number;

choice number;

overexception exception;

begin

dbms\_output.put\_line('1. WITHDRWAL 2. DEPOSIT');

dbms\_output.put\_line('Enter Choice');

choice:=2;

withdrawl:=1000;

select balance into current from acc\_master where id=1;

if choice=1 then

if withdrawl > current then

raise overexception;

else

current:=current-withdrawl;

update acc\_master set balance=balance-withdrawl where id=1;

end if;

else

deposit:=2000;

update acc\_master set balance=balance+deposit where id=1;

end if;

exception

when overexception then

dbms\_output.put\_line('Amount Exceeded');

end;

# 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(

-> id int,

-> bal\_due int

-> );

insert into client\_master(1 , 5000);

insert into client\_master values(2 , 000);

insert into client\_master values(3 , -4647);

create procedure check\_bal(in cid int)

begin

declare balance int;

declare exit handler for sqlexception select 'BR violated' message;

select bal\_due into balance from client\_master where id = cid;

if (balance<0) then select 'BR violated' message;

else

select 'Accepted'message;

end if;

end;

$

call check\_bal(3)

$

# 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.

mysql> CREATE TABLE BORROW(

-> ROLL\_NO INTEGER PRIMARY KEY,

-> NAME VARCHAR(50),

-> DATEOFISSUE DATE,

-> NAMEOFBOOK VARCHAR(50),

-> STATUS VARCHAR(50));

Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE FINE(

-> ROLL\_NO INTEGER PRIMARY KEY,

-> DATEOFRETURN DATE,

-> AMT INTEGER);

Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE FINE( ROLL\_NO INTEGER PRIMARY KEY, DATEOFRETURN DATE, AMT INTEGER);

Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO BORROW VALUES(1, 'Shreeya', '2022-08-20', 'TOC', 'I');

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO BORROW VALUES(2, 'Rutuja', '2022-08-15', 'JAVA', 'I');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO BORROW VALUES(3, 'Apurva', '2022-09-11', 'SPOS', 'I');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO BORROW VALUES(4, 'Shreya', '2022-09-08', 'CNS', 'I');

Query OK, 1 row affected (0.00 sec)

mysql> DELIMITER $

mysql> 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;

-> $

Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> CALL FINE\_CALCULATION(3, 'SPOS');

Query OK, 1 row affected (0.01 sec)

mysql> CALL FINE\_CALCULATION(2, 'JAVA');

Query OK, 1 row affected (0.01 sec)

mysql> CALL FINE\_CALCULATION(4, 'CNS');

Query OK, 1 row affected (0.01 sec)

mysql> CALL FINE\_CALCULATION(1, 'TOC');

Query OK, 1 row affected (0.01 sec)

mysql> SELECT \*FROM FINE;

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

| ROLL\_NO | DATEOFRETURN | AMT |

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

| 1 | 2022-09-20 | 1550 |

| 2 | 2022-09-20 | 1800 |

| 3 | 2022-09-20 | 0 |

| 4 | 2022-09-20 | 0 |

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

4 rows in set (0.00 sec)

mysql> SELECT \*FROM BORROW;

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

|ROLL\_NO | NAME | DATEOFISSUE| NAMEOFBOOK | STATUS |

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

| 1 | Shreeya| 2022-08-20 | TOC | R |

| 2 | Rutuja | 2022-08-15 | JAVA | R |

| 3 | Apurva | 2022-09-11 | SPOS | R |

| 4 | Shreya | 2022-09-08 | CNS | R |

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

4 rows in set (0.00 sec)

**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>**

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.

**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.**

16) 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.

**CREATE DATABASE cursors;**

**use cursors;**

**create table new\_roll(roll integer, name varchar(10));**

**create table old\_roll(roll integer, name varchar(10));**

**insert into new\_roll values(2,'shweta');**

**insert into new\_roll values(4,'nikita');**

**insert into new\_roll values(5,'vikrant');**

**insert into new\_roll values(1,'palak');**

**insert into old\_roll values(2,'shweta');**

**insert into old\_roll values(4,'nikita');**

**insert into old\_roll values(3,'pranita');**

**insert into old\_roll values(1,'palak');**

**insert into old\_roll values(5,'vikrant');**

**select \* from new\_roll;**

**+------+---------+**

**| roll | name |**

**+------+---------+**

**| 2 | shweta |**

**| 4 | nikita |**

**| 5 | vikrant |**

**| 1 | palak |**

**+------+---------+**

**select \* from old\_roll;**

**+------+---------+**

**| roll | name |**

**+------+---------+**

**| 2 | shweta |**

**| 4 | nikita |**

**| 3 | pranita |**

**| 1 | palak |**

**| 5 | vikrant |**

**+------+---------+**

**DELIMITER $**

**CREATE PROCEDURE roll\_list()**

**BEGIN**

**DECLARE a int;**

**DECLARE a1 varchar(10);**

**DECLARE b int;**

**DECLARE b1 varchar(10);**

**DECLARE done int default false;**

**DECLARE c1 cursor for select roll,name from old\_roll;**

**DECLARE c2 cursor for select roll,name from new\_roll;**

**DECLARE continue handler for not found set done=true;**

**OPEN c1;**

**OPEN c2;**

**loop1:loop**

**FETCH c1 into a,a1;**

**if done then**

**leave loop1;**

**end if;**

**loop2:loop**

**FETCH c2 into b,b1;**

**if done then**

**insert into new\_roll values(a,a1);**

**leave loop2;**

**end if;**

**if a=b then leave loop2;**

**end if;**

**end loop;**

**end loop;**

**CLOSE c1;**

**CLOSE c2;**

**end;**

**$;**

**CALL roll\_list()$;**

**SELECT \* FROM new\_roll$;**

**+------+---------+**

**| roll | name |**

**+------+---------+**

**| 2 | shweta |**

**| 4 | nikita |**

**| 5 | vikrant |**

**| 1 | palak |**

**| 3 | pranita |**

**+------+---------+**

**17) c)** 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\_s()

-> 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;

-> $

Query OK, 0 rows affected (0.01 sec)

**18. TRIGGER:**

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**

# Teachers(Tname,dno,dname,experience,salary,date\_of\_joining ) Students(Sname,roll\_no,class)

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()

1. Find the information about all teachers of computer department

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

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

db.Teachers.find().pretty()

1. 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()

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

db.Students.find({$or:[{'roll\_no':2},{'Sname':'xyz'}]}).pretty()

1. 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}})

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

db.Teachers.updateMany({dname:'IT'}, {$set:{dname:'Computer'}})

1. 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})

1. 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'

})

1. Using Save() method change the dept of teacher Rajesh to IT
2. Delete all the doccuments from teachers collection having IT dept

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

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

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

**OR**

1. Find the information about all teachers

db.Teachers.find()

2. Find the information about all teachers of computer department

db.Teachers.find({dname: "Computer"});

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

db.Teachers.find({$or:[{dname: "Computer"},{dname: "IT"},{dname: "ENTC"}]})

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

salary greater than or equal to 10000/-

db.Teachers.find({$and:[{$or:[{{dname: "Computer"},{dname: "IT"},{dname: "ENTC"}]}, {salary:{$gte:10000}}]}).pretty()

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

db.Student.find({$or:[{rollno: 2},{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.

db.Teachers.update({Tname:"Pqr"},{$set:{experience:10}})

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

db.Teachers.update({dname:"IT"},{$set:{dname:"Computer"}})

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

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

db.department.insert({dname:"Computer", dno:86})

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().limit(3)

**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

1. 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()

1. Find the average salary teachers of computer department

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

1. 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"}}}])

1. 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()

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

Same as above questions

1. 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.

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

Same as above

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

db.Teachers.find({},{dname:0,dno:0,salary:0,date\_of\_joining:0}).pretty()

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

Same as above

1. 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()

**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"

db.city.insertOne({city:'LA',state:'california',population:5000,location:'hollywood'});

db.city.insertOne({city:'Timkeen',state:'KS',population:1200,location:'atlantis'});

db.city.insertOne({city:'Timkeen',state:'KS',population:1000,location:'hillwood'});

db.city.insertOne({city:'las vegas',state:'nevada',population:4900,location:'las vegas'});

db.city.insertOne({city:'columbus',state:'georgia',population:3800,location:'atlanta'});

db.city.insertOne({city:'atlanta',state:'georgia',population:3200,location:'atlanta'});

db.city.updateOne({city:'atlanta'},{$set :{city:['atlanta','athens','savannah','columbus','milton']}});

db.city.updateOne({city:'LA'},{$set :{city:['LA','SAn Francisco','Okaland','San Bruno','Santa Monica']}});

db.city.find();

db.city.createIndex({city:1});

db.city.find({city:'atlanta'}).explain("executionStats");

db.city.createIndex({'location':-1,population:1});

db.city.find({location:'atlanta',population:{$gt:3000}}).explain("executionStats");

db.city.createIndex({'city.LA':1});

db.city.find({city:'LA'});

db.city.find({city:'LA'}).explain("executionStats");

db.city.find({population : {$gt : 1600}}, {city:1});

db.city.find({'population' : {$gt : 1600}}, {'city':1});

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

db.city.find({city : 'Timkeen'}, {location:1});

**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()

1. Write a query that returns only games that have achievements
2. Using MapReduce in mongodb solve following queries on given below collection.

{

“id” : 0,

“name” : “Leanne Flinn”,

“email” : “[leanne.flinn@unilogic.com”,](mailto: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

**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.** 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

**use library;**

**switched to db library**

**> show dbs;**

**DYPIT 0.000GB**

**admin 0.000GB**

**config 0.000GB**

**local 0.000GB**

**student 0.000GB**

**> book1 = {name : "Understanding JAVA", pages : 100}**

**{ "name" : "Understanding JAVA", "pages" : 100 }**

**> book2 = {name : "Understanding JSON", pages : 200}**

**{ "name" : "Understanding JSON", "pages" : 200 }**

**> db.books.save(book1)**

**WriteResult({ "nInserted" : 1 })**

**> db.books.save(book2)**

**WriteResult({ "nInserted" : 1 })**

**> db.books.find();**

**{ "\_id" : ObjectId("637081620f41a217fc02de5d"), "name" : "Understanding JAVA", "pages" : 100 }**

**{ "\_id" : ObjectId("6370816a0f41a217fc02de5e"), "name" : "Understanding JSON", "pages" : 200 }**

**> book = {name : "Understanding XML", pages : 300}**

**{ "name" : "Understanding XML", "pages" : 300 }**

**> db.books.save(book)**

**WriteResult({ "nInserted" : 1 })**

**> book = {name : "Understanding Web Services", pages : 400}**

**{ "name" : "Understanding Web Services", "pages" : 400 }**

**> db.books.save(book)**

**WriteResult({ "nInserted" : 1 })**

**> book = {name : "Understanding Axis2", pages : 150}**

**{ "name" : "Understanding Axis2", "pages" : 150 }**

**> db.books.save(book)**

**WriteResult({ "nInserted" : 1 })**

**> var map = function() {**

**... var category;**

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

**... category = 'Big Books';**

**... else**

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

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

**... };**

**> var reduce = function(key, values) {**

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

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

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

**... });**

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

**... };**

**> var count = db.books.mapReduce(map, reduce, {out: "book\_results"});**

**> db[count.result].find()**

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

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

**>**