# sudo apt update;

sudo apt upgrade;

sudo apt install mysql -server;

sudo apt install mysql -client;

mysql -u root -o

sudo mysql;

# sudo apt mysql;

sudo mysql;

ubuntu

# All queries by Isha Borgaonkar

# MYSQL

**Problem Statement 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

**Creation of table and Insertion**

CREATE TABLE dept

(deptno number(4) not null ,

deptname varchar(20) not null,

location varchar(20) ,

PRIMARY KEY (deptno)

);

insert into dept values(1,'Finance','Pune');

insert into dept values(2,'Accounts','Pune');

insert into dept values(3,'Finance','Delhi');

insert into dept values(4,'Sales','Chennai');

insert into dept values(5,'Traning','Mumbai');

insert into dept values(10,'Development','Mumbai');

insert into dept values(20,'Testing','Mumbai');

insert into dept values(6,'Dev','Mumbai');

create table emp(

eno number(3) not null,

ename varchar(20),

job varchar(20),

hiredate date,

salary number(30),

commission number,

dept\_no number,

primary key(eno),

FOREIGN KEY (dept\_no) REFERENCES dept(deptno)

);

insert into emp values(107,'Ojas','Trainer',date '1980-10-05',50000,6,5);

insert into emp values(108,'Avadhut','Trainer',date '1981-07-02',50000,6,5);

insert into emp values(103,'Rutuja','Tester',date '2022-10-30',50000,10,20);

insert into emp values(104,'Priyanka','Tester',date'20221030',40000,10,20);

insert into emp values(105,'Isha','Advisor',date '2022-01-23',150000,1,1);

insert into emp values(106,'Ishan','Assisteant',date '2022-01-30',100000,1,1);

**Solve Queries by SQL**

1)List the maximum salary paid to salesman

select max(salary) from emp where job='salesman';

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

select ename from emp where ename like 'I%';

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

select \* from emp where hiredate<date'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 dept\_no=20;

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

select avg(salary),min(salary) from emp where dept\_no=10 ORDER BY hiredate ;

7)List emp name and its department

SELECT e.dept\_no , e.ename, d.deptname from emp e,dept d WHERE e.dept\_no = d.deptno;

8)List total salary paid to each department

SELECT dept\_no, SUM(salary)FROM emp GROUP BY dept\_no;

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

select \* from dept where deptname='Dev';

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

UPDATE emp SET salary= salary + (salary \* 5 / 100) WHERE dept\_no = 10;

select \* from emp;

**Problem Statement 2:**

**Create a database**

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

works (employee name, company name, salary)

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

manages (employee name, manager name)

create table employee (

employeename varchar(20) primary key,

street varchar(20),

city varchar(20)

);

create table company(

companyname varchar(20) primary key,

city varchar(20)

);

create table works(

employeename varchar(20),

companyname varchar(20),

salary real,

primary key (employeename,companyname),

foreign key (employeename) references employee(employeename),

foreign key (companyname) references company(companyname)

);

create table manages(

employeename varchar(20),

managername varchar(20),

primary key (employeename,managername),

foreign key (employeename) references employee(employeename),

foreign key (managername) references employee(employeename)

);

insert into employee values ('Isha','NeheruNagar','Pune');

insert into employee values ('Avadhut','NeheruNagar','Pune');

insert into employee values ('Sanjana','BalajiNagar','Katraj');

insert into employee values ('Samu','BalajiNagar','Katraj');

insert into employee values ('Adi','GpRoad','Nashik');

insert into employee values ('Mansi','GpRoad','Nashik');

select \* from employee;

insert into company values('FirstBankCorporation','Pune');

insert into company values('StateBank','Pune');

insert into company values('YesBank','Nahsik');

insert into company values('HSBCBank ','Pune');

insert into company values('TJSBBank ','Nashik');

insert into company values('CenteralBank ','Nashik');

select \* from company;

insert into works values ('Isha','FirstBankCorporation',40000);

insert into works values ('Avadhut','StateBank',30000);

insert into works values ('Sanjana','YesBank',35000);

insert into works values ('Samu','HSBCBank ',25000);

insert into works values ('Adi','TJSBBank',15000);

insert into works values ('Mansi','CenteralBank',10000);

select \* from works;

insert into manages values ('Isha','Avadhut');

insert into manages values('Samu','Adi');

insert into manages values('Sanjana','Mansi');

select \* from manages;

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

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

select employeename from works where companyname='FirstBankCorporation';

2)Find all employees who do not work for First Bank Corporation

select employeename from works where companyname<>'FirstBankCorporation';

3)Find the company that has most employees.

select max(companyname) from company;

SELECT companyname FROM works GROUP BY companyname HAVING count(\*) = ( SELECT count(\*) FROM works GROUP BY company-name

ORDER BY count(\*) DESC LIMIT 1)

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

SELECT companyname, city FROM company UNION ALL SELECT companyname, city FROM company WHERE companyname = 'StateBank';

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

select \* from works where salary>10000;

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

UPDATE works SET salary= salary + (salary \* 10/ 100) WHERE companyname='FirstBankCorporation';

7)Find employee and their managers.

select \* from manages;

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

SELECT e.employeename , e.street, e.city from employee e,works w WHERE w.companyname='FirstBankCorporation' and w .salary>10000 and e.employeename=w.employeename;

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

select companyname from works group by companyname having avg(salary)>(select avg(salary) from works where companyname='First Bank Corporation');

select dname, count(\*) count\_of\_employees from dept, emp where dept.deptno = emp.deptno group by DNAME order by desc

**Problem Statement 3:**

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

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

Room (RoomNo, HotelNo, Type, Price)

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

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

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

(HotelNo, GuestNo and DateFrom)

create table hotel(hotelno number(3), hotelname varchar(20), city varchar(20), primary key(hotelno));

insert into hotel values(1,'GrosvenorHotel','London');

insert into hotel values(2,'SpringVilla','London');

insert into hotel values(3,'Jwmarriot','India');

insert into hotel values(4,'TajPalace','India');

create table room (roomno number(3), hotel\_num number(3),type varchar(20),price number(10), primary key(roomno), foreign key (hotel\_num) references hotel(hotelno));

alter table room modify price number(10);

insert into room values (101,1,'Suite',10000);

insert into room values (102,1,'Motel',20000);

insert into room values (103,2,'Resort',15000);

insert into room values (104,3,'Resort',15000);

insert into room values (105,4,'Motel',20000);

insert into room values (106,4,'Motel',20000);

create table guest (guestno number(3), guestname varchar(20), gaddress varchar(20), primary key(guestno) );

insert into guest values(11,'Isha','UK');

insert into guest values(12,'Avadhut','UK');

insert into guest values(13,'Sanjana','USA');

insert into guest values(14,'Adi','INDIA');

insert into guest values(15,'Samu','INDIA');

create table booking(

hotel\_no number(3),

guest\_no number(3),

room\_no number(3),

datefrom date,

dateto date,

foreign key (hotel\_no) references hotel(hotelno),

foreign key (guest\_no) references guest(guestno),

foreign key (room\_no) references room(roomno));

insert into booking values(1,11,101,date '2022-10-01',date '2022-10-05');

insert into booking values(1,12,101,date '2022-04-01',date '2022-04-05');

insert into booking values(2,13,102,date '2022-04-05',date '2022-04-09');

insert into booking values(3,14,103,date '2022-05-05',date '2022-06-09');

insert into booking values(4,15,104,date '2022-04-05',date '2022-04-09');

select \* from booking;

**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 hotel\_num = (SELECT hotelno FROM hotel WHERE hotelname = 'GrosvenorHotel');

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

SELECT hotel\_num, COUNT(roomno) AS count FROM room

GROUP BY hotel\_num;

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

UPDATE room SET price= price + (price \* 5 / 100);

select \* from room;

**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 hotel\_no =(SELECT hotelno FROM hotel WHERE hotelname = 'GrosvenorHotel'));

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

SELECT hotelno, COUNT(roomno) AS count FROM room r, hotel h WHERE r.hotel\_num = h.hotelno AND city = 'London' GROUP BY h.hotelno;

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

CREATE VIEW view1 AS SELECT hotelno, hotelname FROM hotel where city='London';

select \* from view1;

**Problem Statement 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

**Solve following queries by SQL**

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

SELECT SUM(price) 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 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

hotelname= 'GrosvenorHotel' AND

datefrom <= CURRENT\_DATE AND

dateto >= CURRENT\_DATE)

AS lmn

ON r.hotel\_num = lmn.hotelnum AND r.room\_num = lmn.roomnum;

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 hotelName= ‘Grosvenor Hotel’ 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 avg(hotel\_no) from booking where (datefrom>date'2022-04-01' and dateto<date'2022-04-30');

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

CREATE VIEW view1 AS

SELECT hotelno, hotelname FROM hotel where city='London';

select \* from view1;

**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 \* 5 / 100);

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

SELECT hotelno, COUNT(roomno) AS count FROM room r, hotel h

WHERE r.hotel\_num = h.hotelno AND city = 'London'

GROUP BY h.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 type='Double' and price<40000;

**Problem statement 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)

create table hotel(hotelno number(3), hotelname varchar(20), city varchar(20), primary key(hotelno));

insert into hotel values(1,'GrosvenorHotel','London');

insert into hotel values(2,'SpringVilla','London');

insert into hotel values(3,'Jwmarriot','India');

insert into hotel values(4,'TajPalace','India');

create table room (roomno number(3), hotel\_num number(3),type varchar(20),price number(10), primary key(roomno), foreign key (hotel\_num) references hotel(hotelno));

alter table room modify price number(10);

insert into room values (101,1,'Suite',10000);

insert into room values (102,1,'Motel',20000);

insert into room values (103,2,'Resort',15000);

insert into room values (104,3,'Resort',15000);

insert into room values (105,4,'Motel',20000);

insert into room values (106,4,'Motel',20000);

insert into room values (107,3,'Double',20000);

insert into room values (108,2,'Double',20000);

create table guest (guestno number(3), guestname varchar(20), gaddress varchar(20), primary key(guestno) );

insert into guest values(11,'Isha','UK');

insert into guest values(12,'Avadhut','UK');

insert into guest values(13,'Sanjana','USA');

insert into guest values(14,'Adi','INDIA');

insert into guest values(15,'Samu','INDIA');

create table booking(

hotel\_no number(3),

guest\_no number(3),

room\_no number(3),

datefrom date,

dateto date,

foreign key (hotel\_no) references hotel(hotelno),

foreign key (guest\_no) references guest(guestno),

foreign key (room\_no) references room(roomno));

insert into booking values(1,11,101,date '2022-10-01',date '2022-10-05');

insert into booking values(1,12,101,date '2022-04-01',date '2022-04-05');

insert into booking values(2,13,102,date '2022-04-05',date '2022-04-09');

insert into booking values(3,14,103,date '2022-05-05',date '2022-06-09');

insert into booking values(4,15,104,date '2022-04-05',date '2022-04-09');

**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 hotel\_num =

(SELECT hotelno FROM hotel

WHERE hotelname = 'GrosvenorHotel');

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

SELECT hotel\_num, COUNT(roomno) AS count FROM room

GROUP BY hotel\_num;

**5)List all guests currently staying at the Grosvenor Hotl.**

SELECT \* FROM guest

WHERE guestno =

(S

ELECT guestno FROM booking

WHERE datefrom <= CURRENT\_DATE AND

dateto >= CURRENT\_DATE AND

hotel\_no =

(SELECT hotel\_no FROM hotel

WHERE hotelname = 'GrosvenorHotel'));

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

select \* from room where type='Double' and price<40000;

**7)How many different guests have made bookings for August?**

SELECT COUNT(DISTINCT (guest\_no)) FROM booking

WHERE (datefrom <= DATE'2004-08-01' AND dateto >= DATE'2004-08-01') OR (datefrom >= DATE'2004-08-01' AND datefrom <= DATE'2004-08-31');

SELECT COUNT(DISTINCT guestno)FROM booking WHERE (datefrom <= ‘8/31/06’ AND dateto >= ‘8/1/06’);

**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?**

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

UPDATE room SET price= price + (price \* 5 / 100);

**Problem Statement 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**

create table hotel(hotelno number(3), hotelname varchar(20), city varchar(20), primary key(hotelno));

insert into hotel values(1,'GrosvenorHotel','London');

insert into hotel values(2,'SpringVilla','London');

insert into hotel values(3,'Jwmarriot','India');

insert into hotel values(4,'TajPalace','India');

create table room (roomno number(3), hotel\_num number(3),type varchar(20),price number(10), primary key(roomno), foreign key (hotel\_num) references hotel(hotelno));

alter table room modify price number(10);

insert into room values (101,1,'Suite',10000);

insert into room values (102,1,'Motel',20000);

insert into room values (103,2,'Resort',15000);

insert into room values (104,3,'Resort',15000);

insert into room values (105,4,'Motel',20000);

insert into room values (106,4,'Motel',20000);

insert into room values (107,3,'Double',20000);

insert into room values (108,2,'Double',20000);

create table guest (guestno number(3), guestname varchar(20), gaddress varchar(20), primary key(guestno) );

insert into guest values(11,'Isha','UK');

insert into guest values(12,'Avadhut','UK');

insert into guest values(13,'Sanjana','USA');

insert into guest values(14,'Adi','INDIA');

insert into guest values(15,'Samu','INDIA');

create table booking(

hotel\_no number(3),

guest\_no number(3),

room\_no number(3),

datefrom date,

dateto date,

foreign key (hotel\_no) references hotel(hotelno),

foreign key (guest\_no) references guest(guestno),

foreign key (room\_no) references room(roomno));

insert into booking values(1,11,101,date '2022-10-01',date '2022-10-05');

insert into booking values(1,12,101,date '2022-04-01',date '2022-04-05');

insert into booking values(2,13,102,date '2022-04-05',date '2022-04-09');

insert into booking values(3,14,103,date '2022-05-05',date '2022-06-09');

insert into booking values(4,15,104,date '2022-04-05',date '2022-04-09');

**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

hotel\_no =

(SELECT hotelno FROM hotel

WHERE hotelname = 'GrosvenorHotel'));

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

SELECT guestname, gaddress FROM guest WHERE gaddress 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.**

**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 show ON Hotel (hotelno, name);

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

CREATE VIEW view1 AS

SELECT hotelno, hotelname FROM hotel where city='London';

**Problem Statement 7:**

Consider the following database

Project(project\_id,proj\_name,chief\_arch) , project\_id is primary key

Employee(Emp\_id,Emp\_name) , Emp\_id is primary key

Assigned-To(Project\_id,Emp\_id)

create table project(project\_id varchar(10), proj\_name varchar(20), chief\_arch varchar(20), primary key(project\_id));

create table employeee(emp\_id number(20), emp\_name varchar(20),primary key(emp\_id));

create table assigned\_to(project\_id varchar(10), emp\_id number(3), foreign key (project\_id) references project(project\_id), foreign key (emp\_id)references employeee(emp\_id));

insert into project values('C353' ,'DatabaseProj','Premier');

insert into project values('C453' ,'DatabaseProj','Interiors');

insert into project values('C351' ,'TestingProj','Premier');

insert into project values('C458' ,'DevalopmentProj','Interiors');

select \* from project;

insert into employeee values(1,'Isha');

insert into employeee values(2,'Avadhut');

insert into employeee values(3,'Sanjana');

insert into employeee values(4,'Samu');

insert into assigned\_to values('C353',1);

insert into assigned\_to values('C453',2);

insert into assigned\_to values('C453',3);

insert into assigned\_to values('C453',4);

select \* from project

**Find the SQL queries for the following:**

**1)Get the details of employees working on project C353**

select \* from project where project\_id = 'C353';

**2)Get employee number of employees working on project C353**

select count(\*) from project where project\_id = 'C353';

**3)Obtain details of employees working on Database projet**

select \* from project where proj\_name='DatabaseProj';

**4)Get details of employees working on both C353 and C354**

select \* from project where project\_id = 'C353' and project\_id = 'C453';

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

select \* from project where project\_id <>'C453';

**Ans key:**

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)

where project\_id = 'C353';

select count(\*) from project where project\_id = 'C353';

select \* from project where proj\_name='DatabaseProj';

select \* from project where project\_id = 'C353' and project\_id = 'C453';

select \* from project where project\_id <>'C453';

second:

second:

create table Project (Project\_id varchar(10) primary key, proj\_name varchar(10) , chief\_arch varchar(20));

create table Employee (Emp\_id varchar(10) primary key, Emp\_name varchar(10));

create table Assigned\_To (Project\_id varchar(10), Emp\_id varchar(10));

insert into Project(Project\_id , proj\_name , chief\_arch) values ('C353','Database','ABC');

insert into Project(Project\_id , proj\_name , chief\_arch) values ('C354','Big data','PQR');

insert into Project(Project\_id , proj\_name , chief\_arch) values ('C453','Cloud','LMN');

insert into Project(Project\_id , proj\_name , chief\_arch) values ('C352','Networking','DEF');

insert into Project(Project\_id , proj\_name , chief\_arch) values ('C351','OS','XYZ');

insert into Employee(Emp\_id , Emp\_name) values ('01','Nobita');

insert into Employee(Emp\_id , Emp\_name) values ('02','Shijuka');

insert into Employee(Emp\_id , Emp\_name) values ('03','Jian');

insert into Employee(Emp\_id , Emp\_name) values ('04','Kenichi');

insert into Employee(Emp\_id , Emp\_name) values ('05','Yumiko');

insert into Assigned\_To(Project\_id , Emp\_id) values ('C353','01');

insert into Assigned\_To(Project\_id , Emp\_id) values ('C354','01');

insert into Assigned\_To(Project\_id , Emp\_id) values ('C353','02');

insert into Assigned\_To(Project\_id , Emp\_id) values ('C351','03');

insert into Assigned\_To(Project\_id , Emp\_id) values ('C453','04');

**Find the SQL queries for the following:**

**1. Get the details of employees working on project C353**

select Project\_id ,Emp\_name from Assigned\_To inner join Employee on Employee.Emp\_id = Assigned\_To.Emp\_id where Project\_id = 'C353';

**2. Get employee number of employees working on project C353**

select count(Emp\_id)from Assigned\_To where Project\_id = 'C353';

**3. Obtain details of employees working on Database project**

select \* from Project where proj\_name = 'Database';

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

select Emp\_name from Employee inner join Assigned\_To on Employee.Emp\_id = Assigned\_To. Emp\_id where Project\_id = ('C353''C354');

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

select \* from Assigned\_To inner join Employee on Employee.Emp\_id = Assigned\_To. Emp\_id where Project\_id <> 'C453';

**Problem Statement 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)

create table Employee (emp\_no varchar(10) primary key , name varchar(10), skill varchar(10),payrate varchar(10));

create table Position (posting\_no varchar(10) primary key , skill varchar(10));

create table Duty\_allocation(posting\_no varchar(10), emp\_no varchar(10), day varchar(10), shift varchar(10));

insert into Employee(emp\_no, name, skill, payrate) values ('123461', 'ABC','chef', '10000');

insert into Employee(emp\_no, name, skill, payrate) values ('123460', 'PQR','manager', '15000');

insert into Employee(emp\_no, name, skill, payrate) values ('123462', 'LMN','chef', '10000');

insert into Employee(emp\_no, name, skill, payrate) values ('123463', 'XYZ','waiter', '5000');

insert into Employee(emp\_no, name, skill, payrate) values ('123464', 'DEF','captain', '7000');

insert into Duty\_allocation(posting\_no , emp\_no, day, shift)values ('01','123461','Monday','Morning');

insert into Duty\_allocation(posting\_no , emp\_no, day, shift)values ('02','123460','Monday','Morning');

insert into Duty\_allocation(posting\_no , emp\_no, day, shift)values ('03','123463','Monday','Morning');

insert into Duty\_allocation(posting\_no , emp\_no, day, shift)values ('04','123464','Monday','Morning');

insert into Duty\_allocation(posting\_no , emp\_no, day, shift)values ('01','123462','Monday','Morning');

insert into Position (posting\_no, skill)values('01','chef');

insert into Position (posting\_no, skill)values('02','manager');

insert into Position (posting\_no, skill)values('03','waiter');

insert into Position (posting\_no, skill)values('04','cptain');

**Find the SQL queries for the following:**

1. **Get the duty allocation details for emp\_no 123461 for the month of April 1986.**

select \* from Duty\_allocation where emp\_no = '123461' and shift = 'morning';

1. **Find the shift details for Employee ‘xyz’**

select name,day,shift from Employee inner join Duty\_allocation on Employee.emp\_no = Duty\_allocation.emp\_no where name = 'XYZ';

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

select \* from Employee where payrate >= '5000';

**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, payrate from Employee where emp\_no < '123460' and payrate > some (select payrate 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 \* from Employee where skill ='chef';

**6 .Find the employees with the lowest pay rate**

select emp\_no, Name, Payrate from Employee where payrate <= all (select payrate from Employee)

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

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

select \* from Employee where skill = 'chef';

**9 .Get a list of employees not assigned a duty**

select \* from Duty\_allocation where shift = 'Null';

**10.Get a count of different employees on each shift**

select shift, COUNT(emp\_no) from Duty\_allocation group by shift;

**Ans key 2:=**

**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. 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
3. .Get a list of employees not assigned a duty

10.Get a count of different employees on each shift

select shift, count(distinct emp\_no) from duty\_allocation group by shift;

ans :-

Employee(emp\_no,name,skill,pay-rate) eno primary key

create table Eemployee (emp\_no number(10), name varchar(20), payrate number(10),primary key(emp\_no));

insert into Eemployee values (123461,'Isha',100000);

insert into Eemployee values (123460,'Avadhut',300000);

insert into Eemployee values (123463,'Manisha',200000);

insert into Eemployee values(123464,'Mahesh',400000);

select \* from Eemployee;

Position(posting\_no,skill) posting\_no primary key

create table position(posting\_no number(10), skills varchar(20), primary key(posting\_no));

insert into position values(1,'Chef');

insert into position values(2,'Chef');

insert into position values(3,'Manager');

insert into position values(4,'Accounttant');

select \* from position;

Duty\_allocation(posting\_no,emp\_no,day,shift)

insert into duty\_allocation values(1,123461,date'1986-04-16',1);

insert into duty\_allocation values(2,123460,date'1986-04-15',1);

insert into duty\_allocation values(3,123463,date'1986-04-16',2);

insert into duty\_allocation values(4,123464,date'1986-05-16',2);

create table duty\_allocation (posting\_no number(10), emp\_no number(10), day date, shift number(2), foreign key (posting\_no) references position(posting\_no), foreign key (emp\_no) references Eemployee(emp\_no));

select shift, count(distinct emp\_no) from duty\_allocation group by shift;

**Problem Statement 9:**

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

Add primary key and foreign key wherever applicable.

Insert data into the above created tables.

create table Deposite (actno varchar(10),cname varchar(10), bname varchar(10), amount varchar(10), adate varchar(10));

create table Branch (bname varchar(10) primary key, city varchar(10));

create table Borrow(loanno varchar(10), cname varchar(10), bname varchar(10) , amount varchar(10));

create table Customer(cname varchar(10), city varchar(10));

insert into Deposite(actno, cname, bname, amount, adate) values ('11','Anil','Axis','1000','1996/08/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('12','Sunil','PNB','3000','2022/09/03');

insert into Deposite(actno, cname, bname, amount, adate) values ('13','Pravin','SBI','5000','1997/07/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('14','Vijay','ICICI','7000','2022/10/05');

insert into Deposite(actno, cname, bname, amount, adate) values ('15','Arjun','HDFC','9000','1996/04/08');

insert into Branch (bname, city)values ('Axis','Pune');

insert into Branch (bname, city)values ('PNB','Mumbai');

insert into Branch (bname, city)values ('SBI','Delhi');

insert into Branch (bname, city)values ('HDFC','Perryridge');

insert into Branch (bname, city)values ('TJSB','Chennai');

insert into Customer (cname, city)values('Anil','Pune');

insert into Customer (cname, city)values('Sunil','Perryridge');

insert into Customer (cname, city)values('Pravin','Pune');

insert into Customer (cname, city)values('Vijay','Culcutta');

insert into Customer (cname, city)values('Arjun','Pune');

insert into Customer (cname, city)values('Vinod','Pune');

insert into Borrow(loanno, cname, bname, amount)values('21','Anil','Axis','10000');

insert into Borrow(loanno, cname, bname, amount)values('22','Sunil','PNB','10500');

insert into Borrow(loanno, cname, bname, amount)values('23','Arjun','HDFC','18000');

insert into Borrow(loanno, cname, bname, amount)values('24','Vijay','ICICI','10700');

insert into Borrow(loanno, cname, bname, amount)values('21','Pravin','TJSB','10000');

**1. Display names of depositors having amount greater than 4000.**

SELECT cname FROM Deposite WHERE amount >4000;

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

Select adate from Deposite 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 Deposite WHERE adate BETWEEN '1996/12/01' AND '1997/05/01';

**4. Find the average account balance at the Perryridge branch.**

select avg (amount),cname from Borrow inner join Branch on Borrow.bname = Branch.bname AND bname = 'Perryridge';

**5. Find the names of all branches where the average account balance is more than $1,200.**

select bname, avgbalance from (select bname, avg (amount) from Borrow group by bname) as result (bname, avgbalance) where avgbalance > '1200';

**6. Delete depositors having deposit less than 5000**

Delete from Deposite where amount <5000;

select \* from Deposite;

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

CREATE VIEW [Bank] AS SELECT cname, bname FROM Borrow WHERE loanno = '21';

select \* from [Bank];

**Problem Statement 10**

1. **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.

create table Deposite (actno varchar(10),cname varchar(10), bname varchar(10), amount varchar(10), adate varchar(10));

create table Branch (bname varchar(10) primary key, city varchar(10));

create table Borrow(loanno varchar(10), cname varchar(10), bname varchar(10) , amount varchar(10));

create table Customer(cname varchar(10), city varchar(10));

insert into Deposite(actno, cname, bname, amount, adate) values ('11','Anil','Axis','1000','1996/08/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('12','Sunil','PNB','3000','2022/09/03');

insert into Deposite(actno, cname, bname, amount, adate) values ('13','Pravin','SBI','5000','1997/07/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('14','Vijay','ICICI','7000','2022/10/05');

insert into Deposite(actno, cname, bname, amount, adate) values ('15','Arjun','HDFC','9000','1996/04/08');

insert into Branch (bname, city)values ('Axis','Pune');

insert into Branch (bname, city)values ('PNB','Mumbai');

insert into Branch (bname, city)values ('SBI','Delhi');

insert into Branch (bname, city)values ('HDFC','Perryridge');

insert into Branch (bname, city)values ('TJSB','Chennai');

insert into Customer (cname, city)values('Anil','Pune');

insert into Customer (cname, city)values('Sunil','Perryridge');

insert into Customer (cname, city)values('Pravin','Pune');

insert into Customer (cname, city)values('Vijay','Culcutta');

insert into Customer (cname, city)values('Arjun','Pune');

insert into Customer (cname, city)values('Vinod','Pune');

insert into Borrow(loanno, cname, bname, amount)values('21','Anil','Axis','10000');

insert into Borrow(loanno, cname, bname, amount)values('22','Sunil','PNB','10500');

insert into Borrow(loanno, cname, bname, amount)values('23','Arjun','HDFC','18000');

insert into Borrow(loanno, cname, bname, amount)values('24','Vijay','ICICI','10700');

insert into Borrow(loanno, cname, bname, amount)values('21','Pravin','TJSB','10000');

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

Select \* from Branch where city='Bombay';

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

select actno , amount from Deposite;

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

Update Customer set city='Mumbai' where city='Pune';

select \* from Customer;

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

select COUNT(actno) from Deposite;

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

select MIN(amount) from Deposite;

select MAX(amount) from Deposite;

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

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

**Problem Statement 11:**

1. 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.

create table Deposite (actno varchar(10),cname varchar(10), bname varchar(10), amount varchar(10), adate varchar(10));

create table Branch (bname varchar(10) primary key, city varchar(10));

create table Borrow(loanno varchar(10), cname varchar(10), bname varchar(10) , amount varchar(10));

create table Customer(cname varchar(10), city varchar(10));

insert into Deposite(actno, cname, bname, amount, adate) values ('11','Anil','Axis','1000','1996/08/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('12','Sunil','PNB','3000','2022/09/03');

insert into Deposite(actno, cname, bname, amount, adate) values ('13','Pravin','SBI','5000','1997/07/02');

insert into Deposite(actno, cname, bname, amount, adate) values ('14','Vijay','ICICI','7000','2022/10/05');

insert into Deposite(actno, cname, bname, amount, adate) values ('15','Arjun','HDFC','9000','1996/04/08');

insert into Branch (bname, city) values ('Axis','Pune');

insert into Branch (bname, city) values ('PNB','Bombay');

insert into Branch (bname, city) values ('SBI','Delhi');

insert into Branch (bname, city) values ('HDFC','Perryridge');

insert into Branch (bname, city) values ('Karolabagh','Chennai');

insert into Customer (cname, city)values('Anil','Pune');

insert into Customer (cname, city)values('Sunil','Perryridge');

insert into Customer (cname, city)values('Pravin','Pune');

insert into Customer (cname, city)values('Vijay','Culcutta');

insert into Customer (cname, city)values('Arjun','Pune');

insert into Customer (cname, city)values('Vinod','Pune');

insert into Borrow(loanno, cname, bname, amount)values('21','Anil','Axis','10000');

insert into Borrow(loanno, cname, bname, amount)values('22','Sunil','PNB','10500');

insert into Borrow(loanno, cname, bname, amount)values('23','Arjun','HDFC','18000');

insert into Borrow(loanno, cname, bname, amount)values('24','Vijay','ICICI','10700');

insert into Borrow(loanno, cname, bname, amount)values('21','Pravin','Karolabagh','10000');

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

select adate from Deposite where cname='Anil';

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

ALTER TABLE Deposite MODIFY actno VARCHAR(20) ;

**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 = 'Karolabagh';

**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';

select \* from Customer;

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

create View deposit\_view as select actno,cname,bname,amount,adate from Deposite;

select \* from Deposit\_view;

**Problem Statement 12:**

1. 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

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 customer name having living city Bombay and branch city Nagpur**

select c.city from Customer c, Branch b where c.city='Bombay' and b.city ='Nagpur';

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

select c.city from Customer c, Branch b where c.city=b.city ;

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

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

select br1.cname, br1.amount, d1.cname, d1.amount from Borrow br1,Deposite d1 where d1.cname = br1.cname and d1.amount > 1000 and br1.amount > 2000;

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

select c.cname from Customer c where c.city in (select b.city from Branch b where b.bname in (select d.bname from Deposite d where d.cname='Sunil'));

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

create index deposit\_index on deposit(actno);

select \* from deposite\_index;

**Problem Statement 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**

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);

insert into publisher(pid, pname, address, state, phone, emailid ) values (1, 'sunrise', 'mumbai', 'maharashtra', '9098765432', 'sunrise12@gmail.com');

insert into publisher(pid, pname, address, state, phone, emailid ) values (2, 'mehta','pune', 'maharashtra', '9128765432', 'addison 12@gmail.com');

insert into publisher(pid, pname, address, state, phone, emailid ) values (3,'morgan kaufmann', 'korth', 'maharashtra', '9548765432', 'morgan12@gmail.com');

insert into book(isbn, book\_title, category, price, copyright\_date, year, page\_count, pid) values ('0321228383', 'database systems', 'a', 255, 12, 2007, 86, 1);

insert into book(isbn, book\_title, category, price, copyright\_date, year, page\_count, pid) values ('0321228384', 'computer science', 'b', 205, 12, 2007, 80, 2);

insert into book(isbn, book\_title, category, price, copyright\_date, year, page\_count, pid) values ('0321228385', 'out of their minds', 'c', 145, 12, 2007, 70, 3);

insert into author\_book(aid, isbn) values (10,'0321228383');

insert into author\_book(aid, isbn) values (20,'0321228384');

insert into author\_book(aid, isbn) values (30,'0321228385');

insert into review (rid, isbn, rating) values(201, '0321228383', 4);

insert into review (rid, isbn, rating) values(202, '0321228384', 3);

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

insert into author (aid, aname, state, city, zip, phone, url) values (10, 'Chetan bhagat', 'maharashtra', 'mumbai', 401205, '9098765432', 'www.k10.com');

insert into author (aid, aname, state, city, zip, phone, url) values (20, 'lewis', 'maharashtra', 'pune',410501, '9128765432', 'www.lewis20.com');

insert into author (aid, aname, state, city, zip, phone, url) values (30, 'bernstein', 'maharashtra', 'korth', 402501, '9548765432', 'www.bern30.com');

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

select city,phone,url from author where aname='Chetan Bhagat';

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

select book\_title,rid,rating from review r,book b where b.isbn=r.isbn;

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

select book\_title,price,aname,url from book b,author a,publisher p where b.pid=p.pid and p.pname = 'MEHTA';

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

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

select \* from publisher;

**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;

select \* from book;

**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.**

**Part 2: PL/SQL**

**Q) 14.**

1. **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”

**Code 1:**

create table stud1( roll number(2) not null primary key, att number(2) not null,

status char(2));

insert into stud1 (roll,att) values (1,88);

insert into stud1 (roll,att) values (2,80);

insert into stud1 (roll,att) values (3,50);

insert into stud1 (roll,att) values (4,40);

insert into stud1 (roll,att) values (5,70);

**Declaration of Procedure**

declare

s\_roll stud1.roll%type:=&r\_roll;

s\_att stud1.att%type;

begin

select att into s\_att from stud1 where roll=s\_roll;

if s\_att<75 then

update stud1 set status='D' where roll=s\_roll;

dbms\_output.put\_line('Term Not Granted');

else

update stud1 set status='ND' where roll=s\_roll;

dbms\_output.put\_line('Term Granted');

end if;

end;

**Expected Output:-**

Enter value for r\_roll: 1

old 2: s\_roll stud1.roll%type:=&r\_roll;

new 2: s\_roll stud1.roll%type:=1;

Term Granted

Enter value for r\_roll: 3

old 2: s\_roll stud1.roll%type:=&r\_roll;

new 2: s\_roll stud1.roll%type:=3;

Term Not Granted

select \* from stud1;

**code 2:**

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

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

select \* from stud;

delimiter //

create procedure check\_att(in roll int)

begin

declare att int;

declare total int;

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

set total=200;

select attendance into att from stud where RollNo=roll;

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

update stud set status='ND' where RollNo=roll;

select 'Term Granted' Message;

else

update stud set status='D' where RollNo=roll;

select 'Term Not Granted' Message;

end if;

end;

//

call check\_att(1);

call check\_att(2);

call check\_att(3);

select \* from stud;

**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.

**Code 1:**

create table account\_master(

acc\_no number(3) not null primary key,

amount number(10,2)

);

insert into account\_master(acc\_no,amount)values(1,1500);

insert into account\_master(acc\_no,amount)values(2,15000);

insert into account\_master(acc\_no,amount)values(3,1000);

insert into account\_master(acc\_no,amount)values(4,11000);

insert into account\_master(acc\_no,amount)values(5,1100);

**Procedure :**

declare

c\_acc account\_master.acc\_no%type := &c\_acc;

c\_amount account\_master.amount%type;

e\_amount account\_master.amount%type;

invalid\_amount exception;

begin

e\_amount := &e\_amount;

select amount into c\_amount from account\_master where acc\_no = c\_acc;

if e\_amount>c\_amount then

raise invalid\_amount;

else

update account\_master set amount = amount-e\_amount where acc\_no = c\_acc;

dbms\_output.put\_line('Amount Withdrawal Succefull ');

end if;

exception

when invalid\_amount then

dbms\_output.put\_line('Your Account balance is low');

end;

**Expected Output:**

Enter value for c\_acc: 1

old 2: c\_acc account\_master.acc\_no%type:=&c\_acc;

new 2: c\_acc account\_master.acc\_no%type:=1;

Enter value for e\_amount: 1600

old 8: e\_amount:=&e\_amount;

new 8: e\_amount:=1600;

Enter amount:

Your account balance is low

PL/SQL procedure successfully completed.

Enter value for c\_acc: 1

old 2: c\_acc account\_master.acc\_no%type:=&c\_acc;

new 2: c\_acc account\_master.acc\_no%type:=1;

Enter value for e\_amount: 100

old 8: e\_amount:=&e\_amount;

new 8: e\_amount:=100;

Amount withdrawl Successful

PL/SQL procedure successfully completed.

Code 2:-

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

insert into account\_master values(1,10000)

insert into account\_master values (2,5000)

insert into account\_master values (3,60000);

select\*from account\_master;

delimiter //

create procedure withdraw(in acc\_id int,in amt int)

begin

declare bal int;

declare sp condition for sqlstate '45000';

select Current\_balance into bal from account\_master where ID=acc\_id;

if bal<amt then

signal sqlstate '45000'

set message\_text='NotEnoughBalance';

else

set bal = bal-amt;

update account\_master set Current\_balance=bal where ID=acc\_id;

end if;

end;

//

create procedure deposit(in acc\_id int,in amt int)

begin

declare bal int;

select current\_balance into bal from account\_master where ID=acc\_id;

update account\_master set current\_balance=bal+amt where ID=acc\_id;

end;

//

call withdraw(3,40000);

select\*from account\_master;

call deposit(2,2000);

select\*from account\_master;

call withdraw(1,75000);

**Example :15**

**A)Write an SQL code block** these raise a user defined exception where business rule is violated. BR for client\_ master table specifies when the value of bal\_due field is less than 0 handle the exception.

**Code 1:**

**Ans:=**

create table account\_master2(

acc\_no number(3) not null primary key,

amount number(10,2) ,bal\_due number(3)

);

insert into account\_master2(acc\_no,amount,bal\_due)values(1,1500,0);

insert into account\_master2(acc\_no,amount, bal\_due)values(2,15000,-1);

insert into account\_master2(acc\_no,amount, bal\_due)values(3,1000,-3);

insert into account\_master2(acc\_no,amount, bal\_due)values(4,11000,9);

insert into account\_master2(acc\_no,amount, bal\_due)values(5,1100,-2);

**Procedure :=**

declare

cust\_acc account\_master2.acc\_no%type := &c\_acc;

cust\_amount account\_master2.amount%type;

balance\_due account\_master2.bal\_due%type;

business\_rule\_violation exception;

begin

select bal\_due into balance\_due from account\_master2 where acc\_no = cust\_acc;

if balance\_due<0 then

raise business\_rule\_violation;

else

dbms\_output.put\_line('Business Rule NOT violated ');

end if;

exception

when business\_rule\_violation then

dbms\_output.put\_line(' Business Rule violated ');

end;

**Expected Output:**

set serveroutput on;

Enter value for c\_acc: 2

old 2: cust\_acc account\_master2.acc\_no%type := &c\_acc;

new 2: cust\_acc account\_master2.acc\_no%type := 2;

Business Rule violated

**Code 2:**

delimiter //

create procedure check\_br(in uid int)

begin

declare temp\_bal int;

declare sp condition for sqlstate'45000';

select bal\_due into temp\_bal from client\_master where id=uid;

if temp\_bal<0 then

signal sqlstate '45000'

set message\_text='BR violated';

else

select 'BR not violated' Message;

end if;

end

//

**C)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.

**Answer 1:=**

create table borrower(rollno number(5) not null primary key,name varchar2(20),dateofissue date not null ,nameofbook varchar2(20),status varchar2(20)) ;

insert into borrower values (1, 'Isha' , date '2017-03-06' , 'DBMS', 'I');

insert into borrower values(2,'Avadhut', date '2022-05-04','PLSQL','I');

insert into borrower values(3,'Ritul', date '2022-05-25','SQL','I');

insert into borrower values(4,' Sanjana', date '2022-06-03','Java','I');

insert into borrower values(5,'Aditya', date '2022-08-01','Cpp','I');

create table fine(rno number(10),sdate date,fineamount number(20));

**Procedure:=**

declare

rno number(5):=&rollno;

noofdays number(7);

issuedate date;

fineamount number(5);

begin

select dateofissue into issuedate from borrower where rollno=rno;

select sysdate -to\_date(issuedate)days into noofdays from dual;

if(noofdays>=15 and noofdays<=30)then

fineamount:=noofdays\*5;

elsif noofdays>30 then

fineamount:=noofdays\*50;

end if;

update borrower set status='R' where rollno=rno;

if fineamount>=0 then

insert into fine values(rno,sysdate,fineamount);

end if;

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('record not found');

end;

**Expected Output:=**

set serveroutput on;

Enter value for rollno: 2

old 2: rno number(5):=&rollno;

new 2: rno number(5):=2;

PL/SQL procedure successfully completed.

SQL> select \* from fine;

RNO SDATE FINEAMOUNT

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

2 13-NOV-22 9650

SQL> select \* from borrower;

ROLLNO NAME DATEOFISS NAMEOFBOOK STATUS

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

1 Isha 17-APR-01 DBMS I

2 Avadhut 04-MAY-22 PLSQL R

3 Ritul 25-MAY-22 SQL R

4 Sanjana 03-JUN-22 Java I

5 Aditya 01-AUG-22 Cpp I

**Ans 2:**

First Create Both the tables Borrower and Fine and Insert values in Borrower

Create procedure calcfine6(rno int,bname varchar(20))

begin

declare fine int default 0;

declare days int default 0;

declare issuedate date;

declare namee varchar(20);

declare exit handler for SQLEXCEPTION select 'create table definition';

Select name into namee from Borrower where Roll\_no = rno;

Select DOI into issuedate from Borrower where Roll\_no = rno AND NameofBook = bname;

Set days = datediff(now(),issuedate);

If(days > 15 && days < 30) then

Set fine = days \* 5;

End if;

If(days > 30) then

Set fine = days \* 50;

End if;

If(fine > 0) then

Insert into finet values(rno,namee,fine);

End if;

Update Borrower set Status = 'R' where Roll\_no = rno AND NameofBook = bname;

End

**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).

create table bankcursor(acc\_no number(10), status varchar(10));

insert into bankcursor values(101,'active');

insert into bankcursor values(102,'inactive');

insert into bankcursor values(103,'inactive');

insert into bankcursor values(104,'active');

insert into bankcursor values(105,'inactive');

select \* from bankcursor;

ACC\_NO STATUS

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

101 active

102 inactive

103 inactive

104 active

105 inactive

declare

Rows\_aff number(10);

Rows\_found number(10);

Rows\_notfound number(10);

begin

update bankcursor set status='active' where status='inactive';

IF sql%FOUND THEN

dbms\_output.put\_line(' Inactive Record Found');

END IF;

IF sql%NOTFOUND THEN

dbms\_output.put\_line(' Active Record Found');

END IF;

Rows\_aff:=(SQL%rowcount);

dbms\_output.put\_line(Rows\_aff ||'Rows are affected..' );

end;

/

**Expected output:**

Found

1 Rows are affected..

select \* from bankcursor;

ACC\_NO STATUS

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

101 active

102 active

103 active

104 active

105 active

106 active

6 rows selected.

**Code 2:**

**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.**

**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.

**Code 1:**

**Create table employee(e\_id int,salary int);**

**insert into employee values(1,10000);**

**insert into employee values(2,11000);**

**insert into employee values(3,12500);**

**insert into employee values(4,9000);**

**insert into employee values(5,10000);**

**insert into employee values(6,7000);**

**insert into employee values(7,12000);**

**create table increament\_t(**

**e\_id int,**

**salary int**

**);**

**Declare**

**Cursor c1 is select e\_id,salary from employee where salary < (select avg(salary) from employee);**

**me\_no employee.e\_id%type;**

**msalary employee.salary%type;**

**Begin**

**open c1;**

**if(c1%isopen) then**

**loop**

**fetch c1 into me\_no,msalary;**

**exit when c1%notfound;**

**if(c1%found) then**

**update employee set salary = salary + (salary \* 0.10) where e\_id = me\_no;**

**select salary into msalary from employee where e\_id = me\_no;**

**insert into increament\_t values(me\_no,msalary);**

**end if;**

**end loop;**

**end if;**

**end;**

**code 2:=**

create table emp2(e\_no number(10) , salary number(10));

insert into emp2 values(101,1000);

insert into emp2 values(102,2000);

insert into emp2 values(103,113);

insert into emp2 values(104,4000);

logic :=

(1000,2000,113,4000)/4= 1778.25

Below average salary values =1000,113(get updated )

create table increment\_salary(e\_no number (10),salary number(10));

**Code:=**

declare

Cursor crsr\_sal is select e\_no,salary from emp2 where salary<(select avg(salary) from emp2);

me\_no emp2.e\_no%type;

msalary emp2.salary%type;

begin

open crsr\_sal;

if crsr\_sal%isopen then

loop

fetch crsr\_sal into me\_no,msalary;

exit when crsr\_sal%notfound;

if crsr\_sal%found then

update emp2 set salary=salary+(salary\*0.1) where

e\_no=me\_no;

select salary into msalary from emp2 where e\_no=me\_no;

insert into increment\_salary values(me\_no,msalary);

end if;

end loop;

end if;

end;

**Expected Output :=**

SQL> select \* from increment\_salary;

E\_NO SALARY

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

101 1100

103 124

**Code 3:**

declare

av\_salary number(10,2);

begin

av\_salary := &av\_salary;

update employee2 set salary = salary\*0.10 where salary <av\_salary;

if sql%found then

dbms\_output.put\_line('Rows Updated: '||sql%rowcount);

elsifsql%notfound then

dbms\_output.put\_line('No Record Found');

end if;

end;

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));

**Code 1:**

**create table stud1(roll\_no number(5),attendance number(5),status varchar(7));**

**insert into stud1 values(8,76,'');**

**insert into stud1 values(2,74,'');**

**insert into stud1 values(3,77,'');**

**insert into stud1 values(4,73,'');**

**insert into stud1 values(5,78,'');**

**insert into stud1 values(6,72,'');**

**declare**

**cursor c1 is select roll\_no,attendance, from employee;**

**roll stud1.roll\_no%type;**

**attend stud1.attendance%type;**

**begin**

**open c1;**

**if(c1%isopen) then**

**loop**

**fetch c1 into roll,attend;**

**exit when c1%notfound;**

**if(attend > 75) then**

**update stud1 set status = 'ND' where roll\_no = roll;**

**else**

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

**insert into D\_stud values(roll);**

**end if;**

**end loop;**

**end if;**

**end;**

**code 2 ISHA:**

create table stud21(roll number(4) not null primary key, att number(4) not null,

status varchar(1) );

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

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

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

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

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

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

create table d\_stud(roll number(4) not null,att number(4) not null, satus varchar(1) );

set linesize 160;

select \* from stud21;

ROLL ATT S

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

1 78

2 58

3 76

4 66

5 56

6 88

6 rows selected.

declare

cursor stu\_cursor is

select roll,att from stud21 where att<75;

stud\_record stu\_cursor%rowtype;

begin

open stu\_cursor;

loop

fetch stu\_cursor into stud\_record;

exit when stu\_cursor%notfound;

insert into d\_stud (roll,att) values (stud\_record.roll,stud\_record.att);

update stud21 set status = 'D' where roll = stud\_record.roll;

end loop;

end;

PL/SQL procedure successfully completed.

select \* from stud21;

ROLL ATT S

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

1 78

2 58 D

3 76

4 66 D

5 56 D

6 88

6 rows selected.

select \* from d\_stud;

ROLL ATT S

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

2 58

4 66

5 56

**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)

create table bankcursor(acc\_no number(10), status varchar(10));

insert into bankcursor values(101,'active');

insert into bankcursor values(102,'inactive');

insert into bankcursor values(103,'inactive');

insert into bankcursor values(104,'active');

insert into bankcursor values(105,'inactive');

select \* from bankcursor;

ACC\_NO STATUS

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

101 active

102 inactive

103 inactive

104 active

105 inactive

declare

Rows\_aff number(10);

Rows\_found number(10);

Rows\_notfound number(10);

begin

update bankcursor set status='active' where status='inactive';

IF sql%FOUND THEN

dbms\_output.put\_line(' Inactive Record Found');

END IF;

IF sql%NOTFOUND THEN

dbms\_output.put\_line(' Active Record Found');

END IF;

Rows\_aff:=(SQL%rowcount);

dbms\_output.put\_line(Rows\_aff ||'Rows are affected..' );

end;

/

**Expected output:**

Found

1 Rows are affected..

select \* from bankcursor;

ACC\_NO STATUS

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

101 active

102 active

103 active

104 active

105 active

106 active

6 rows selected.

**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:

**Code 1:**

**create table o\_rollcall(**

**roll\_no int,**

**name varchar(20)**

**);**

**create table n\_rollcall(**

**roll\_no int,**

**name varchar(20)**

**);**

**create procedure n4(IN rno1 int)**

**begin**

**declare rno2 int;**

**declare exit\_cond boolean;**

**declare c1 cursor for select roll\_no from o\_rollcall where roll\_no>rno1;**

**open c1;**

**l1:loop**

**fetch c1 into rno2;**

**if not exists(select \* from n\_rollcall where roll\_no = rno2) then**

**insert into n\_rollcall select \* from o\_rollcall where roll\_no = rno2;**

**end if;**

**if exit\_cond then**

**close c1;**

**leave l1;**

**end if;**

**end loop l1;**

**end;**

**code Isha :-**

create table O\_rollcall (rollno number(10) not null primary key, name

varchar(20));

insert into O\_rollcall values(1,'Isha');

insert into O\_rollcall values(2,'Avadhut');

insert into O\_rollcall values(3,'Mahesh');

insert into O\_rollcall values(4,'Ritul');

insert into O\_rollcall values(5,'Aniket');

insert into O\_rollcall values(6,'Soham');

create table N\_rollcall(rollno number(10) not null primary key ,name varchar2(20));

insert into N\_rollcall values(3,'Mahesh');

insert into N\_rollcall values(5,'Aniket');

insert into N\_rollcall values(6,'Soham');

declare

cursor c1 is

select rollno,name from O\_rollcall;

cursor c2 is

select rollno from N\_rollcall;

r number(4);

rno number(4);

nm varchar2(20);

begin

open c1;

open c2;

loop

fetch c1 into rno,nm;

fetch c2 into r;

exit when c1%found=false;

if r<>rno then

insert into N\_rollcall values(rno,nm);

end if;

end loop;

close c1;

end;

PL/SQL procedure successfully completed.

select \* from N\_rollcall;

ROLLNO NAME

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

3 Mahesh

1 Isha

2 Avadhut

4 Ritul

5 Aniket

6 Soham

6 rows selected.

**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).**

**(not done this)**

**Code 1:**

**create table EMP(e\_no int, d\_no int, Salary int);**

**create table dept\_salary(d\_no int, Avg\_salary int);**

**CREATE or REPLACE PROCEDURE avgsal**

**IS**

**CURSOR department is**

**select d\_no,AVG(salary) as avgsal from emp**

**group by d\_no;**

**BEGIN**

**FOR data in department**

**LOOP**

**insert into dept\_salary values(data.d\_no,data.avgsal);**

**end LOOP;**

**end;**

**code isha:**

create table emp(e\_no number(10) not null primary key, d\_no number(10), salary number(30));

insert into emp values(1,101,1000);

insert into emp values(2,101,1000);

insert into emp values(3,102,2000);

insert into emp values(4,102,1500);

insert into emp values(5,103,800);

insert into emp values(6,103,400);

create table dept\_salary(d\_no number(10) primary key, avg\_salary number(10));

declare

cursor c1 is

select e\_no,d\_no,salary from emp;

e\_no number(4);

d\_no number(4);

salary number(20);

begin

open c1;

loop

fetch c1 into e\_no,d\_no,salary;

exit when c1%found=false;

SELECT AVG(SALARY) AS

AVERAGE\_SALARY FROM emp GROUP BY d\_no;

insert into dept\_salary values(d\_no,average\_salary);

end loop;

close c1;

end;

# 18.Trigger

**A)Write a update, delete trigger on clientmstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit\_trade table. (separate implementation using both row and statement triggers).**

**Code 1:**

**create or replace trigger storedata**

**before delete**

**on**

**clientmstr**

**for each row**

**begin**

**insert into audit\_trade values(old.c\_id,c\_name);**

**end;**

**/**

**create or replace trigger storedata**

**before update**

**on**

**clientmstr**

**for each row**

**begin**

**insert into audit\_trade values(old.c\_id,c\_name);**

**end;**

**/**

CREATE TABLE clientmstr

(

id number PRIMARY KEY,

name VARCHAR(45),

salary INT,

gender VARCHAR(12),

departmentId number

) ;

CREATE TABLE audit\_trade (

salary number

) ;

INSERT INTO clientmstr VALUES (1,'Isha', 82000, 'Female', 3);

INSERT INTO clientmstr VALUES (2,'Avadhut', 79000, 'Male', 3);

INSERT INTO clientmstr VALUES (3,'Ritul', 70000, 'Female', 4);

INSERT INTO clientmstr VALUES (4,'Rushikesh', 79000, 'Female', 4);

CREATE OR REPLACE TRIGGER display\_salary\_changes

BEFORE DELETE OR INSERT OR UPDATE ON clientmstr 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 audit\_trade values(sal\_diff);

END;

Trigger created.

update clientmstr set salary=85080 where id=2;

1 row updated.

select \* from audit\_trade;

**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).**

**Code 1:**

**create or replace trigger emp\_sal\_trigger**

**before insert or update on emp**

**for each row**

**begin**

**if:NEW.salary < 50000 then**

**RAISE\_APPLICATION\_ERROR(-20001,'MINIMUM SALARY SHOULD BE 50000');**

**end if;**

**end;**

**Part 3: =**

## **Mongodb**

**Example 19**.

Create Database DYPIT using MongoDB

Create following Collections

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

**Students(Sname,roll\_no,class)**

db.createCollection('Teachers')

db.createCollection('Students')

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

db.Teachers.insertOne({

'Tname': 'Sunita',

'dno': 1,

'dname': 'Computer',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

})

db.Teachers.insertMany([{

'Tname': 'Rajesh',

'dno': 1,

'dname': 'IT',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

},

{

'Tname': 'Praveen',

'dno': 2,

'dname': 'Computer',

'experience':5,

'salary':100011,

'date\_of\_joining':'2/2/2012'

},

{

'Tname': 'Rucha',

'dno': 3,

'dname': 'E&TC',

'experience':17,

'salary':200001,

'date\_of\_joining':'9/6/1996'

},

{

'Tname': 'Minal',

'dno': 2,

'dname': 'IT',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

},

{

'Tname': 'Rupesh',

'dno': 2,

'dname': 'E&TC',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

}

])

**Student :**

db.createCollection(‘Students’)

db.Students.insertMany([{

'Sname': 'Isha',

'roll\_no': 1,

'class': 'Computer'

},

{

'Sname': 'Avadhut',

'roll\_no': 2,

'class': 'E&TC'

},

{

'Sname': 'Ritul',

'roll\_no': 3,

'class': 'IT'

},

{

'Sname': 'Rushikesh',

'roll\_no': 4,

'class': 'IT'

}

])

**1)Find the information about all teachers**

db.Teachers.find().pretty()

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

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

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

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

**4)department having salary greate than or equl to 10000/-**

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

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

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

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

db.Teachers.insert({

'Tname': 'Praveen',

'dno': 3,

'dname': 'E&TC',

'experience':10,

'salary':5001,

'date\_of\_joining':'1/1/2021'

})

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

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

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

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

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

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

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

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

**10)Using Save() method change the dept of teacher Rajesh to IT 11.**

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

WriteResult({ "nInserted" : 1 })

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

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

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

Mongo last 2

Map reduce

**Problem 20 :-**

1.Create Database DYPIT

use DYPIT;

1. 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()

1. Find the information about all teachers of computer department

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

1. Find the information about all teachers of computer,IT,and e&TC department
2. 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()

1. Find the student information having roll\_no = 25 or Sname=xyz
2. db.Students.find({$or:[{'roll\_no':2},{'Sname':'avadhut'}]}).pretty()
3. 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':10,

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

db.Teachers.save({

'Tname': 'Rajesh',

'dno': 1,

'dname': 'Computer',

'experience':8,

'salary':50001,

'date\_of\_joining':'1/1/2019'

})

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

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

**Example 21 : Create Database DYPIT using MongoDB**

**Create following Collections**

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

**Students(Sname,roll\_no,class)**

Use DYPIT

Create following Collections

db.createCollection('Teachers')

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

db.Teachers.insertOne({

'Tname': 'Sunita',

'dno': 1,

'dname': 'Computer',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

})

db.Teachers.insertMany([{

'Tname': 'Rajesh',

'dno': 1,

'dname': 'IT',

'experience':11,

'salary':10001,

'date\_of\_joining':'1/1/2001'

},

{

'Tname': 'Praveen',

'dno': 2,

'dname': 'Computer',

'experience':5,

'salary':100011,

'date\_of\_joining':'2/2/2012'

},

{

'Tname': 'Rucha',

'dno': 3,

'dname': 'E&TC',

'experience':17,

'salary':200001,

'date\_of\_joining':'9/6/1996'

},

{

'Tname': 'Minal',

'dno': 2,

'dname': 'IT',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

},

{

'Tname': 'Rupesh',

'dno': 2,

'dname': 'E&TC',

'experience':7,

'salary':10002,

'date\_of\_joining':'1/1/2011'

}

])

Students(Sname,roll\_no,class)

db.createCollection(‘Students’)

db.Students.insertMany([{

'Sname': 'Isha',

'roll\_no': 1,

'class': 'Computer'

},

{

'Sname': 'Avadhut',

'roll\_no': 2,

'class': 'E&TC'

},

{

'Sname': 'Ritul',

'roll\_no': 3,

'class': 'IT'

},

{

'Sname': 'Rushikesh',

'roll\_no': 4,

'class': 'IT'

}

])

**1)Find the information about all teachers.**

db.Teachers.find().pretty()

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

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

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

db.Teachers.aggregate([{$match:{"dname":"E&TC"}},{$group : {\_id : "$dname", salary\_maximum : {$max : "$salary"}, salary\_minimum:{$min : "$salary"}}}])

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

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

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

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

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

db.Teachers.insert({

'Tname': 'Praveen',

'dno': 3,

'dname': 'E&TC',

'experience':10,

'salary':5001,

'date\_of\_joining':'1/1/2021'

})

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

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

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

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

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

**9)Using Save() method insert one entry in department collection 10. Find the total salary all teachers.**

db.Teachers.aggregate([{$group : {\_id : "", total\_salary : {$sum : "$salary"}}}])

**Problem Statement 22:**

**1)Create Database DYPIT using MongoDB**

**2)Create following Collections**

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

**3)Display the department wise average salary**

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

**4)display the no. Of employees working in each department**

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

**5)Display the department wise total salary of departments having total salary greater than or equals to 50000/-**

db.Teachers.find({'salary':{$gte:50000}}).pretty()

**6)Write the queries using the different operators like max, min. Etc.**

db.Teachers.aggregate([{$match:{"dname":"E&TC"}},{$group : {\_id : "$dname", salary\_maximum : {$max : "$salary"}, salary\_minimum:{$min : "$salary"}}}])

**7)Create unique index on any field for above given collections**

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

**8)Create compound index on any fields for above given collections 7. Show all the indexes created in the database DYPIT**

db.Teachers.getIndexes()

**9)Show all the indexes created in above collections.**

db.Teachers.getIndexes()

**Problem Statement 23:**

Create index and fire queries with MongoDB

**1)Import zip.json.**

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

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

db.zip.createIndex( {'ind' : 1 });

db.zip.createIndex( {'index' : 2, 'name' : 4 })

db.zip.createIndex( {'index' : 2, 'name' : 4 },{'indw' : 3 })

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

3)Display all cities having population above 1600.

db.zip.find( { population : {$gt : 1600} })

4**)Display all cities in state “KS”.**

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

**5)Display location of city "TIMKEN"**

db.zip.find( {city : 'TIMKEN'}, {location :1})

**Problem Statement 24 :-**

**Design and Implement following query using MongoDB**

**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': '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’ and the ‘Speed Demon’ achievements.**

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

**7)Write a query that returns only games that have achievements.**

**Problem Statement 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” }

db.prac25.insertMany(

... [

... { id : 0, name : 'Larry Page', email : 'krishna@gmail.com', work :'SDE-1' , age :26, gender :'Male', Salary :16660, hobbies :'singing' },

... { id : 1, name : 'Elon musk', email : 'elon@gmail.com', work :'SDE-2' , age :36, gender :'Male', Salary :196660, hobbies :'wrapper' },

... { id : 2, name : 'sarvesh', email : 'sarvesh@gmail.com', work :'Developer' , age : 29, gender :'Female', Salary : 898896, hobbies :'ceater' },

... { id : 3, name : 'chetan', email : 'chetan@gmail.com', work :'engineer' , age :20, gender :'Male', Salary :16660, hobbies :'singing' },

... { id : 4, name : 'sanket', email : 'sanket@gmail.com', work :'manager' , age :26, gender :'Female', Salary :89656, hobbies :'sleeping' }

... ]

... )

1. Get the count of Males and Females

db.prac25.aggregate([

{

"$group": {

"\_id": "$gender",

"count": { "$sum": 1 }

}

}

])

1. Count the number of users in each hobby

db.prac25.aggregate([

{$match : { } },

{$group : {\_id :'$hobbies', sumOfHobby : {$sum :1 } } }

])

* + 1. Get the count of Males and Females
    2. Count the number of users in each hobby

**Problem Statement 26**

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

* 1. Import zip.json.
  2. Find total population in each state.

Import zip.json.

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

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();

**Problem Statement : 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

db.books.insertOne({name : "Understanding JAVA", pages : 100})

db.books.insertOne({name : "Understanding JSON", pages : 200})

db.books.insertMany([

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

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

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

... ])

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()