9. Create database propertydearalers

Branch(branchno, street, city, postcode)

Staff(Staffno, fname, lname, position, Sex, DOB, salary, branchno)

PropertyforRent(propertyNo, street, city, Postcode,Type,rooms,rent,staffno,branchno)

1. Insert the data as shown in the above tables

2. Update the lname of the staff with staffno. 106 to shinde

3. Delete the branch with postcode 411041

4. List the employee who have at least 2 A's in their name.

5. List the maximum salary from each branch of the staff members from highest to lowest

6. List all properties managed by a staff member having id S123

7. Find the average rent of properties in a Pune

8. Find the number of properties managed by each branch

Ans:

```
CREATE DATABASE propertydealers;

USE propertydealers;

CREATE TABLE Branch (
    branchno INT PRIMARY KEY,
    street VARCHAR(50),
    city VARCHAR(50),
    postcode VARCHAR(10)
);

CREATE TABLE Staff (
    staffno INT PRIMARY KEY,
    fname VARCHAR(50),
    lname VARCHAR(50),
    position VARCHAR(50),
    sex CHAR(1),
```

```
    DOB DATE,
    salary DECIMAL(10, 2),
    branchno INT,
    FOREIGN KEY (branchno) REFERENCES Branch(branchno)
);

CREATE TABLE PropertyforRent (
    propertyNo INT PRIMARY KEY,
    street VARCHAR(50),
    city VARCHAR(50),
    postcode VARCHAR(10),
    type VARCHAR(50),
    rooms INT,
    rent DECIMAL(10, 2),
    staffno INT,
    branchno INT,
    FOREIGN KEY (staffno) REFERENCES Staff(staffno),
    FOREIGN KEY (branchno) REFERENCES Branch(branchno)
);

INSERT INTO Branch (branchno, street, city, postcode) VALUES
(1, 'MG Road', 'Pune', '411001'),
(2, 'JM Road', 'Mumbai', '400001'),
(3, 'FC Road', 'Pune', '411041');

INSERT INTO Staff (staffno, fname, lname, position, sex, DOB, salary, branchno) VALUES
(101, 'Aakash', 'Patil', 'Manager', 'M', '1985-01-10', 50000, 1),
(102, 'Aarti', 'Kulkarni', 'Sales', 'F', '1990-05-21', 30000, 1),
(106, 'Rahul', 'Joshi', 'Agent', 'M', '1988-09-18', 35000, 2),
(103, 'Shreya', 'Sharma', 'Agent', 'F', '1995-12-14', 30000, 3),
(104, 'Anand', 'Rao', 'Sales', 'M', '1987-07-19', 40000, 2);
```

```
INSERT INTO PropertyforRent (propertyNo, street, city, postcode, type, rooms, rent, staffno, branchno) VALUES
(1, 'Street 1', 'Pune', '411001', 'Apartment', 3, 15000, 101, 1),
(2, 'Street 2', 'Mumbai', '400001', 'Villa', 4, 25000, 102, 2),
(3, 'Street 3', 'Pune', '411041', 'Flat', 2, 12000, 106, 2),
(4, 'Street 4', 'Mumbai', '400001', 'Apartment', 3, 18000, 104, 1),
(5, 'Street 5', 'Pune', '411041', 'Apartment', 2, 10000, 103, 3);

UPDATE Staff
SET lname = 'Shinde'
WHERE staffno = 106;

DELETE FROM Branch
WHERE postcode = '411041';

SELECT * FROM Staff
WHERE fname LIKE '%A%A%' OR lname LIKE '%A%A%';

SELECT branchno, MAX(salary) AS max_salary
FROM Staff
GROUP BY branchno
ORDER BY max_salary DESC;

SELECT * FROM PropertyforRent
WHERE staffno = 'S123';

SELECT AVG(rent) AS avg_rent
FROM PropertyforRent
WHERE city = 'Pune';

SELECT branchno, COUNT(propertyNo) AS num_properties
```

FROM PropertyforRent

GROUP BY branchno;

10. Create database propertydearalers

Branch(branchno, street, city, postcode)

Staff(Staffno, fname, lname, position, Sex, DOB, salary, branchno)

PropertyforRent(propertyNo, street, city, Postcode,Type, rooms,rent,staffno, branchno)

1. Insert the data as shown in the above tables

2. Update the street of branchNo 1001 to MG

3. List the name of staff who have salary greater than average salary of all staff.

4. Find the type and rooms of properties in pune

5. Find the name of staffs who work as salesman or saleswomen

6. Find the no. of properties which are of type flat.

7. List staff members born before 2004.

8. Find the total number of staff members in each branch.

Ans:

```
CREATE DATABASE propertydearalers;
USE propertydearalers;

-- Create Branch table
CREATE TABLE Branch (
    branchno INT PRIMARY KEY,
    street VARCHAR(50),
    city VARCHAR(50),
    postcode VARCHAR(10)
);

-- Create Staff table with a foreign key to Branch
CREATE TABLE Staff (
```

```
    staffno INT PRIMARY KEY,
    fname VARCHAR(50),
    lname VARCHAR(50),
    position VARCHAR(50),
    sex CHAR(1),
    dob DATE,
    salary DECIMAL(10, 2),
    branchno INT,
    FOREIGN KEY (branchno) REFERENCES Branch(branchno) ON DELETE CASCADE
);

-- Create PropertyforRent table with foreign keys to Staff and Branch
CREATE TABLE PropertyforRent (
    propertyNo INT PRIMARY KEY,
    street VARCHAR(50),
    city VARCHAR(50),
    postcode VARCHAR(10),
    type VARCHAR(50),
    rooms INT,
    rent DECIMAL(10, 2),
    staffno INT,
    branchno INT,
    FOREIGN KEY (staffno) REFERENCES Staff(staffno),
    FOREIGN KEY (branchno) REFERENCES Branch(branchno) ON DELETE CASCADE
);

-- Insert data into Branch
INSERT INTO Branch (branchno, street, city, postcode) VALUES
(1001, 'Main St', 'Pune', '411041'),
(1002, 'High St', 'Mumbai', '400001'),
(1003, 'Low St', 'Delhi', '110001');
```

```
-- Insert data into Staff
INSERT INTO Staff (staffno, fname, lname, position, sex, dob, salary, branchno) VALUES
(101, 'John', 'Doe', 'Salesman', 'M', '1985-03-15', 50000, 1001),
(102, 'Jane', 'Smith', 'Saleswoman', 'F', '1990-07-20', 55000, 1001),
(103, 'David', 'Lee', 'Manager', 'M', '1980-12-05', 70000, 1002),
(104, 'Emily', 'Jones', 'Saleswoman', 'F', '2002-09-12', 48000, 1003),
(105, 'Michael', 'Brown', 'Salesman', 'M', '2003-04-25', 45000, 1001);

-- Insert data into PropertyforRent
INSERT INTO PropertyforRent (propertyNo, street, city, postcode, type, rooms, rent, staffno,
branchno) VALUES
(2001, 'Park Ave', 'Pune', '411041', 'Flat', 3, 15000, 101, 1001),
(2002, 'Market Rd', 'Mumbai', '400001', 'Villa', 5, 40000, 102, 1002),
(2003, 'Green Ln', 'Pune', '411041', 'Flat', 2, 12000, 103, 1001),
(2004, 'Lake View', 'Delhi', '110001', 'House', 4, 30000, 104, 1003);

UPDATE Branch
SET street = 'MG'
WHERE branchno = 1001;

SELECT fname, lname
FROM Staff
WHERE salary > (SELECT AVG(salary) FROM Staff);

SELECT type, rooms
FROM PropertyforRent
WHERE city = 'Pune';

SELECT fname, lname
FROM Staff
```

```
WHERE position IN ('Salesman', 'Saleswoman');

SELECT COUNT(*) AS FlatCount
FROM PropertyforRent
WHERE type = 'Flat';

SELECT fname, lname, dob
FROM Staff
WHERE dob < '2004-01-01';

SELECT branchno, COUNT(*) AS TotalStaff
FROM Staff
GROUP BY branchno;
```

11. 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 %

Asn:

```
CREATE DATABASE company;
USE company;

-- Create the emp table
CREATE TABLE Emp (
    eno INT PRIMARY KEY,
    ename VARCHAR(50),
    job VARCHAR(50),
    hiredate DATE,
    salary DECIMAL(10, 2),
    commission DECIMAL(10, 2),
    deptno INT,
    FOREIGN KEY (deptno) REFERENCES dept(deptno)
);

-- Create the dept table
CREATE TABLE dept (
    deptno INT PRIMARY KEY,
    deptname VARCHAR(50),
    location VARCHAR(50)
);

-- Insert data into dept
INSERT INTO dept (deptno, deptname, location) VALUES
(10, 'Dev', 'New York'),
(20, 'HR', 'London'),
(30, 'Sales', 'Chicago');

-- Insert data into Emp
```

```
INSERT INTO Emp (eno, ename, job, hiredate, salary, commission, deptno) VALUES
(101, 'John Doe', 'Salesman', '1980-08-15', 50000, 5000, 30),
(102, 'Alice Smith', 'Manager', '1978-12-01', 75000, NULL, 10),
(103, 'Bob Brown', 'Salesman', '1985-03-20', 45000, 4000, 30),
(104, 'Ivy Green', 'Developer', '1981-05-15', 80000, NULL, 10),
(105, 'Jack Black', 'Salesman', '1979-07-25', 55000, 4500, 30),
(106, 'Ingrid White', 'HR Specialist', '1982-10-10', 60000, NULL, 20),
(107, 'Eve Blue', 'Manager', '1975-04-12', 95000, NULL, 10);

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

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

SELECT *
FROM Emp
WHERE hiredate < '1981-09-30';

SELECT *
FROM Emp
ORDER BY salary DESC;

SELECT COUNT(*) AS num_employees, AVG(salary) AS avg_salary
FROM Emp
WHERE deptno = 20;

SELECT hiredate, AVG(salary) AS avg_salary, MIN(salary) AS min_salary
FROM Emp
```

```
WHERE deptno = 10
GROUP BY hiredate;

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

SELECT d.deptname, SUM(e.salary) AS total_salary
FROM Emp e
JOIN dept d ON e.deptno = d.deptno
GROUP BY d.deptname;

SELECT *
FROM Emp e
JOIN dept d ON e.deptno = d.deptno
WHERE d.deptname = 'Dev';

UPDATE Emp
SET salary = salary * 1.05
WHERE deptno = 10;
```

12. 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.

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

b. Display account no. and amount of depositors.

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

d. Find the number of depositors in the bank

e. Calculate Min,Max amount of customers.

f. Create an index on deposit table.

g. Create View on Borrow table.

Ans:
```
CREATE TABLE Deposit (
    actno INT PRIMARY KEY,
    cname VARCHAR(50),
    bname VARCHAR(50),
    amount DECIMAL(10, 2),
    adate DATE,
    FOREIGN KEY (cname) REFERENCES Customers(cname),
    FOREIGN KEY (bname) REFERENCES Branch(bname)
);
CREATE TABLE Branch (
    bname VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50)
);
CREATE TABLE Customers (
    cname VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50)
);

CREATE TABLE Borrow (
    loanno INT PRIMARY KEY,
    cname VARCHAR(50),
    bname VARCHAR(50),
    amount DECIMAL(10, 2),
    FOREIGN KEY (cname) REFERENCES Customers(cname),
    FOREIGN KEY (bname) REFERENCES Branch(bname)
);
-- Inserting data into Branch table
```

```
INSERT INTO Branch (bname, city) VALUES ('MainBranch', 'Bombay'), ('SouthBranch', 'Pune'),
('NorthBranch', 'Bombay');

-- Inserting data into Customers table
INSERT INTO Customers (cname, city) VALUES ('Anil', 'Pune'), ('Sunita', 'Bombay'), ('Ramesh', 'Delhi');

-- Inserting data into Deposit table
INSERT INTO Deposit (actno, cname, bname, amount, adate) VALUES
(1001, 'Anil', 'MainBranch', 5000.00, '2024-11-01'),
(1002, 'Sunita', 'SouthBranch', 15000.00, '2024-11-02'),
(1003, 'Ramesh', 'NorthBranch', 8000.00, '2024-11-03');

-- Inserting data into Borrow table
INSERT INTO Borrow (loanno, cname, bname, amount) VALUES
(1, 'Anil', 'MainBranch', 7000.00),
(2, 'Sunita', 'SouthBranch', 12000.00),
(3, 'Ramesh', 'NorthBranch', 15000.00);

SELECT bname FROM Branch WHERE city = 'Bombay';
SELECT actno, amount FROM Deposit;
UPDATE Customers SET city = 'Mumbai' WHERE cname = 'Anil';
SELECT COUNT(*) AS NumberOfDepositors FROM Deposit;
SELECT MIN(amount) AS MinAmount, MAX(amount) AS MaxAmount FROM Deposit;
CREATE INDEX idx_amount ON Deposit(amount);
CREATE VIEW BorrowView AS
SELECT loanno, cname, bname, amount FROM Borrow;
```

13. Create the following tables:

Orders(Order_no, cust, product, Quantity, amount, Disc)

Customers(Cust_No, Company, Cust_Rep, Credit_Limit)

Sales_Representative (Rep_no,Name, Re_office,Quota, sales)

Note: Orders (cust –foreign key for cust_no from Customer)

Customers(Cust_Rep foreign key for Rep_no from Sales_Representative)

Write a query for following:

1. List for each customer: customer name, their rep's name, their rep's office number.

2. List orders over $25,000 including the name of the salesperson who took the order and the name of the

customer who placed it.

3. Find the products which have been sold to TCS.

4. Find company which has been offered maximum discount.

5. Find the sales representatives who work in the same office.

6. Find the credit limit of company and the discount it has received

Asn:

```
CREATE TABLE Orders (
    Order_no INT PRIMARY KEY,
    cust INT,
    product VARCHAR(50),
    Quantity INT,
    amount DECIMAL(10, 2),
    Disc DECIMAL(5, 2),
    FOREIGN KEY (cust) REFERENCES Customers(Cust_No)
);
CREATE TABLE Customers (
    Cust_No INT PRIMARY KEY,
    Company VARCHAR(50),
    Cust_Rep INT,
    Credit_Limit DECIMAL(10, 2),
    FOREIGN KEY (Cust_Rep) REFERENCES Sales_Representative(Rep_no)
);
CREATE TABLE Sales_Representative (
    Rep_no INT PRIMARY KEY,
```

```
    Name VARCHAR(50),
    Re_office VARCHAR(50),
    Quota DECIMAL(10, 2),
    Sales DECIMAL(10, 2)
);

-- Inserting data into Sales_Representative table
INSERT INTO Sales_Representative (Rep_no, Name, Re_office, Quota, Sales) VALUES
(1, 'John Doe', 'Office1', 50000, 30000),
(2, 'Jane Smith', 'Office1', 60000, 45000),
(3, 'Alice Johnson', 'Office2', 70000, 50000);

-- Inserting data into Customers table
INSERT INTO Customers (Cust_No, Company, Cust_Rep, Credit_Limit) VALUES
(101, 'TCS', 1, 100000),
(102, 'Infosys', 2, 150000),
(103, 'Wipro', 3, 200000);

-- Inserting data into Orders table
INSERT INTO Orders (Order_no, cust, product, Quantity, amount, Disc) VALUES
(1001, 101, 'Laptop', 5, 30000, 5.0),
(1002, 102, 'Printer', 10, 28000, 10.0),
(1003, 103, 'Desktop', 8, 20000, 8.0),
(1004, 101, 'Monitor', 15, 5000, 15.0);

SELECT
    c.Company AS Customer_Name,
    sr.Name AS Rep_Name,
    sr.Re_office AS Rep_Office
FROM
    Customers c
```

```
JOIN
    Sales_Representative sr ON c.Cust_Rep = sr.Rep_no;

SELECT
    o.Order_no,
    o.amount,
    c.Company AS Customer_Name,
    sr.Name AS Salesperson_Name
FROM
    Orders o
JOIN
    Customers c ON o.cust = c.Cust_No
JOIN
    Sales_Representative sr ON c.Cust_Rep = sr.Rep_no
WHERE
    o.amount > 25000;

SELECT
    o.product
FROM
    Orders o
JOIN
    Customers c ON o.cust = c.Cust_No
WHERE
    c.Company = 'TCS';

SELECT
    c.Company,
    MAX(o.Disc) AS Max_Discount
FROM
    Orders o
```

```
JOIN
    Customers c ON o.cust = c.Cust_No
GROUP BY
    c.Company
ORDER BY
    Max_Discount DESC
LIMIT 1;

SELECT
    sr1.Name AS Rep1,
    sr2.Name AS Rep2,
    sr1.Re_office AS Office
FROM
    Sales_Representative sr1
JOIN
    Sales_Representative sr2 ON sr1.Re_office = sr2.Re_office AND sr1.Rep_no < sr2.Rep_no;

SELECT
    c.Company,
    c.Credit_Limit,
    SUM(o.Disc) AS Total_Discount
FROM
    Customers c
JOIN
    Orders o ON c.Cust_No = o.cust
GROUP BY
    c.Company;
```

14. Create the following tables:

Orders(Order_no, cust, product, Quantity, amount, Disc)

Customers(Cust_No, Company, Cust_Rep, Credit_Limit)

Sales_Representative (Rep_no,Name, Re_office,Quota, sales)

Note: Orders (cust –foreign key for cust_no from Customer)

Customers(Cust_Rep foreign key for Rep_no from Sales_Representative)

Write a query for following:

1. List for each customer : customer name, credit limit, rep name serving the customer and rep sales.

2. List all orders showing number and amount, and name and credit limit of customer.

3. Find the product wise sale amount of products which exceeds $12000.

4. Find the names of amount, names of customers and names of representatives who have been involved

in the sale of software.

5. Find the credit limit of company and the discount it has received

6. Find the sales representatives who work in the same office.

Ans:

```
CREATE TABLE Orders (
    Order_no INT PRIMARY KEY,
    cust INT,
    product VARCHAR(50),
    Quantity INT,
    amount DECIMAL(10, 2),
    Disc DECIMAL(5, 2),
    FOREIGN KEY (cust) REFERENCES Customers(Cust_No)
);

CREATE TABLE Customers (
    Cust_No INT PRIMARY KEY,
    Company VARCHAR(50),
    Cust_Rep INT,
    Credit_Limit DECIMAL(10, 2),
    FOREIGN KEY (Cust_Rep) REFERENCES Sales_Representative(Rep_no)
);
```

```
CREATE TABLE Sales_Representative (
    Rep_no INT PRIMARY KEY,
    Name VARCHAR(50),
    Re_office VARCHAR(50),
    Quota DECIMAL(10, 2),
    Sales DECIMAL(10, 2)
);

-- Sample data for Sales_Representative
INSERT INTO Sales_Representative (Rep_no, Name, Re_office, Quota, Sales) VALUES
(1, 'John Doe', 'Office1', 50000, 30000),
(2, 'Jane Smith', 'Office1', 60000, 45000),
(3, 'Alice Johnson', 'Office2', 70000, 50000);

-- Sample data for Customers
INSERT INTO Customers (Cust_No, Company, Cust_Rep, Credit_Limit) VALUES
(101, 'TCS', 1, 100000),
(102, 'Infosys', 2, 150000),
(103, 'Wipro', 3, 200000);

-- Sample data for Orders
INSERT INTO Orders (Order_no, cust, product, Quantity, amount, Disc) VALUES
(1001, 101, 'Laptop', 5, 30000, 5.0),
(1002, 102, 'Software', 2, 15000, 10.0),
(1003, 103, 'Desktop', 8, 20000, 8.0),
(1004, 101, 'Software', 1, 12000, 15.0);

SELECT
    c.Company AS Customer_Name,
    c.Credit_Limit,
    sr.Name AS Rep_Name,
```

```
    sr.Sales AS Rep_Sales
FROM
    Customers c
JOIN
    Sales_Representative sr ON c.Cust_Rep = sr.Rep_no;

SELECT
    o.Order_no,
    o.amount,
    c.Company AS Customer_Name,
    c.Credit_Limit
FROM
    Orders o
JOIN
    Customers c ON o.cust = c.Cust_No;

SELECT
    o.product,
    SUM(o.amount) AS Total_Sales
FROM
    Orders o
GROUP BY
    o.product
HAVING
    SUM(o.amount) > 12000;

SELECT
    o.amount,
    c.Company AS Customer_Name,
    sr.Name AS Rep_Name
FROM
```

```
    Orders o
JOIN
    Customers c ON o.cust = c.Cust_No
JOIN
    Sales_Representative sr ON c.Cust_Rep = sr.Rep_no
WHERE
    o.product = 'Software';

SELECT
    c.Company,
    c.Credit_Limit,
    SUM(o.Disc) AS Total_Discount
FROM
    Customers c
JOIN
    Orders o ON c.Cust_No = o.cust
GROUP BY
    c.Company;

SELECT
    sr1.Name AS Rep1,
    sr2.Name AS Rep2,
    sr1.Re_office AS Office
FROM
    Sales_Representative sr1
JOIN
    Sales_Representative sr2 ON sr1.Re_office = sr2.Re_office AND sr1.Rep_no < sr2.Rep_no;
```

15. 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)

Solve following queries by SQL

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

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

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

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

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

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

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

8. Find books with a copyright date before 2010:

And:

```
CREATE TABLE PUBLISHER (
    PID INT PRIMARY KEY,
    PNAME VARCHAR(50),
    ADDRESS VARCHAR(100),
    STATE VARCHAR(20),
    PHONE VARCHAR(15),
    EMAILID VARCHAR(50)
);


CREATE TABLE BOOK (
    ISBN INT PRIMARY KEY,
    BOOK_TITLE VARCHAR(100),
    CATEGORY VARCHAR(30),
    PRICE DECIMAL(10, 2),
    COPYRIGHT_DATE DATE,
    YEAR INT,
    PAGE_COUNT INT,
    PID INT,
```

```
    FOREIGN KEY (PID) REFERENCES PUBLISHER(PID)
);

CREATE TABLE AUTHOR (
    AID INT PRIMARY KEY,
    ANAME VARCHAR(50),
    STATE VARCHAR(20),
    CITY VARCHAR(50),
    ZIP VARCHAR(10),
    PHONE VARCHAR(15),
    URL VARCHAR(100)
);
INSERT INTO PUBLISHER (PID, PNAME, ADDRESS, STATE, PHONE, EMAILID)
VALUES
(1, 'MEHTA', '123 Main St', 'Maharashtra', '9876543210', 'contact@mehta.com'),
(2, 'PEARSON', '456 Elm St', 'Delhi', '9123456780', 'info@pearson.com'),
(3, 'OXFORD', '789 Maple Ave', 'Karnataka', '8765432109', 'help@oxford.com');

INSERT INTO BOOK (ISBN, BOOK_TITLE, CATEGORY, PRICE, COPYRIGHT_DATE, YEAR, PAGE_COUNT,
PID)
VALUES
(101, 'Introduction to Algorithms', 'Computing', 50.00, '2009-08-10', 2009, 1200, 1),
(102, 'Database Systems', 'Computing', 40.00, '2012-05-15', 2012, 950, 2),
(103, 'Theory of Computation', 'Computing', 55.00, '2005-11-30', 2005, 1100, 3),
(104, 'Networking Basics', 'Networking', 35.00, '2011-03-20', 2011, 300, 1),
(105, 'Data Science Fundamentals', 'Data Science', 65.00, '2018-06-10', 2018, 850, 2);
INSERT INTO BOOK (ISBN, BOOK_TITLE, CATEGORY, PRICE, COPYRIGHT_DATE, YEAR, PAGE_COUNT,
PID)
VALUES
(101, 'Introduction to Algorithms', 'Computing', 50.00, '2009-08-10', 2009, 1200, 1),
(102, 'Database Systems', 'Computing', 40.00, '2012-05-15', 2012, 950, 2),
(103, 'Theory of Computation', 'Computing', 55.00, '2005-11-30', 2005, 1100, 3),
```

```
(104, 'Networking Basics', 'Networking', 35.00, '2011-03-20', 2011, 300, 1),
(105, 'Data Science Fundamentals', 'Data Science', 65.00, '2018-06-10', 2018, 850, 2);

INSERT INTO BOOK_AUTHOR (ISBN, AID)
VALUES
(101, 1),   -- 'Introduction to Algorithms' by 'CHETAN BHAGAT'
(102, 2),   -- 'Database Systems' by 'RAVI KUMAR'
(103, 3),   -- 'Theory of Computation' by 'KORTH'
(104, 4),   -- 'Networking Basics' by 'ANDREW TANENBAUM'
(105, 1);   -- 'Data Science Fundamentals' by 'CHETAN BHAGAT'

SELECT CITY, PHONE, URL
FROM AUTHOR
WHERE ANAME = 'CHETAN BHAGAT';

-- Create the BOOK_AUTHOR linking table
CREATE TABLE BOOK_AUTHOR (
    ISBN INT,
    AID INT,
    FOREIGN KEY (ISBN) REFERENCES BOOK(ISBN),
    FOREIGN KEY (AID) REFERENCES AUTHOR(AID)
);

-- Query for book title, price, author name, and URL for publisher ‘MEHTA’
SELECT
    b.BOOK_TITLE,
    b.PRICE,
    a.ANAME,
    a.URL
FROM
    BOOK b
```

```
JOIN
   PUBLISHER p ON b.PID = p.PID
JOIN
   BOOK_AUTHOR ba ON b.ISBN = ba.ISBN
JOIN
   AUTHOR a ON ba.AID = a.AID
WHERE
   p.PNAME = 'MEHTA';

UPDATE PUBLISHER
SET PHONE = '123456'
WHERE PNAME = 'MEHTA';

SELECT
   p.PNAME,
   AVG(b.PRICE) AS Avg_Price,
   MAX(b.PRICE) AS Max_Price,
   MIN(b.PRICE) AS Min_Price
FROM
   BOOK b
JOIN
   PUBLISHER p ON b.PID = p.PID
GROUP BY
   p.PNAME;

DELETE FROM BOOK
WHERE PAGE_COUNT < 100;

SELECT *
FROM AUTHOR
WHERE CITY = 'Pune' AND ANAME LIKE 'C%';
```

```
SELECT *
FROM AUTHOR
WHERE CITY = (SELECT CITY FROM AUTHOR WHERE ANAME = 'Korth');


SELECT *
FROM BOOK
WHERE YEAR(COPYRIGHT_DATE) < 2010;
```

16: Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in

examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are

between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class

```
DELIMITER //


CREATE PROCEDURE proc_Grade(IN student_marks INT, OUT grade VARCHAR(50))
BEGIN
   IF student_marks <= 1500 AND student_marks >= 990 THEN
      SET grade = 'Distinction';
   ELSEIF student_marks BETWEEN 900 AND 989 THEN
      SET grade = 'First Class';
   ELSEIF student_marks BETWEEN 825 AND 899 THEN
      SET grade = 'Higher Second Class';
   ELSE
      SET grade = 'No Category';
   END IF;
END //


CALL proc_Grade(900,@grade);
Select @grade;
```

17. Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.

Suggested Problem statement: Consider Tables:

1. Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)

2. Fine(Roll_no,Date,Amt)

 Accept Roll_no and NameofBook 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 and 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

And: DECLARE

```
   -- Declare variables to accept user input
   v_roll_no        Borrower.Roll_no%TYPE;
   v_name_of_book   Borrower.NameofBook%TYPE;
   v_date_of_issue  Borrower.DateofIssue%TYPE;
   v_status         Borrower.Status%TYPE;

   v_fine_amount    NUMBER(10, 2) := 0; -- Fine amount to be calculated
   v_return_date    DATE;
   v_days_diff      NUMBER;

   -- Exception Declaration
   no_fine_exception EXCEPTION;
   v_error_message  VARCHAR2(100);

BEGIN
   -- Accept user input
   -- In real applications, you would use ACCEPT command or use an interface to accept input
   v_roll_no := &roll_no;  -- User enters Roll No
```

```
v_name_of_book := '&name_of_book';  -- User enters Name of the Book

-- Fetching the Borrower details
SELECT DateofIssue, Status INTO v_date_of_issue, v_status
FROM Borrower
WHERE Roll_no = v_roll_no AND NameofBook = v_name_of_book;

-- Get current date
v_return_date := SYSDATE;

-- Calculate the difference in days
v_days_diff := v_return_date - v_date_of_issue;

-- Control Structure for Fine Calculation
IF v_days_diff BETWEEN 15 AND 30 THEN
   v_fine_amount := v_days_diff * 5; -- Fine of Rs. 5 per day
ELSIF v_days_diff > 30 THEN
   v_fine_amount := v_days_diff * 50; -- Fine of Rs. 50 per day
ELSE
   -- If book is returned in less than 15 days, no fine
   RAISE no_fine_exception;
END IF;

-- Update the status of the Borrower to 'R' (Returned)
UPDATE Borrower
SET Status = 'R'
WHERE Roll_no = v_roll_no AND NameofBook = v_name_of_book;

-- If fine is applicable, insert data into Fine table
IF v_fine_amount > 0 THEN
   INSERT INTO Fine(Roll_no, Date, Amt)
```

```
        VALUES (v_roll_no, v_return_date, v_fine_amount);
    END IF;

    -- Commit the transaction
    COMMIT;

EXCEPTION
    -- Exception Handling for no fine
    WHEN no_fine_exception THEN
        v_error_message := 'No fine applicable for this return.';
        DBMS_OUTPUT.PUT_LINE(v_error_message);
        -- If no fine, update status to returned, just as a safe side
        UPDATE Borrower
        SET Status = 'R'
        WHERE Roll_no = v_roll_no AND NameofBook = v_name_of_book;
        COMMIT;

    -- Generic Exception Handling
    WHEN OTHERS THEN
        v_error_message := 'An error occurred: ' || SQLERRM;
        DBMS_OUTPUT.PUT_LINE(v_error_message);
        ROLLBACK; -- Rollback if any error occurs
END;
/
```

18. Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.

Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scoredby students in

examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are

between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a

PL/SQL block to use procedure created with above requirement. Stud_Marks(name, total_marks)

Result(Roll,Name, Class)

```
DELIMITER //

CREATE PROCEDURE proc_Grade(IN student_marks INT, IN student_name VARCHAR(100), OUT student_class VARCHAR(50))
BEGIN
    IF student_marks >= 990 AND student_marks <= 1500 THEN
        SET student_class = 'Distinction';
    ELSEIF student_marks BETWEEN 900 AND 989 THEN
        SET student_class = 'First Class';
    ELSEIF student_marks BETWEEN 825 AND 899 THEN
        SET student_class = 'Higher Second Class';
    ELSE
        SET student_class = 'No Category';
    END IF;
END //

DELIMITER ;
```

19. Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

Write a database trigger on Library 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 Library_Audit table.

And:

```
CREATE TABLE Library (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Publisher VARCHAR(100),
    Year_Published INT
```

```
);

CREATE TABLE Library_Audit (
    Audit_ID INT AUTO_INCREMENT PRIMARY KEY,
    Action_Type VARCHAR(10),  -- Stores "UPDATE" or "DELETE"
    Book_ID INT,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Publisher VARCHAR(100),
    Year_Published INT,
    Action_Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //

CREATE TRIGGER Library_Update_Audit
AFTER UPDATE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (Action_Type, Book_ID, Title, Author, Publisher, Year_Published)
    VALUES ('UPDATE', OLD.Book_ID, OLD.Title, OLD.Author, OLD.Publisher, OLD.Year_Published);
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER Library_Delete_Audit
BEFORE DELETE ON Library
FOR EACH ROW
BEGIN
```

```
    INSERT INTO Library_Audit (Action_Type, Book_ID, Title, Author, Publisher, Year_Published)
    VALUES ('DELETE', OLD.Book_ID, OLD.Title, OLD.Author, OLD.Publisher, OLD.Year_Published);
END //

DELIMITER ;
UPDATE Library
SET Author = 'Updated Author'
WHERE Book_ID = 1;

DELETE FROM Library
WHERE Book_ID = 2;
```

20. Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

Write a database trigger on Library 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 Library_Audit table.

And:

```
DELIMITER //

CREATE TRIGGER before_Library_update
BEFORE UPDATE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (Book_ID, Title, Author, Publisher, Year_Published, Operation)
    VALUES (OLD.Book_ID, OLD.Title, OLD.Author, OLD.Publisher, OLD.Year_Published, 'UPDATE');
END //

DELIMITER ;

DELIMITER //
```

```
CREATE TRIGGER before_Library_delete
BEFORE DELETE ON Library
FOR EACH ROW
BEGIN
   INSERT INTO Library_Audit (Book_ID, Title, Author, Publisher, Year_Published, Operation)
   VALUES (OLD.Book_ID, OLD.Title, OLD.Author, OLD.Publisher, OLD.Year_Published, 'DELETE');
END //

DELIMITER ;
```

23. SQL Queries - all types of Join, Sub-Query and View: (Note: also examiner can ask it without join)

Create the tables Employee(EmpID, Ename, Salary, Contactno, City, DeptID) and

Department (DeptID, Dname, Location)

1. Display Employee Name and Department name from given entity.

2. Retrieve list of employees along with their department names and locations.(Using inner join or natural join)

3. Display all employee names, location and their department name which are matching with employee table.

4. Display all department names, salary, location and department names of employee.(Right Joins)

5. Display list of all employees and departments (Full joins)

6. Display all possible combinations of employees and departments (cross joins)

7. Retrieve parts of employees who shares the same department(Self Join)

8. Create simple and complex views

9. Update simple view

10. Delete the view

Ans;

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   Dname VARCHAR(50),
   Location VARCHAR(50)
);
```

```
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    Ename VARCHAR(50),
    Salary DECIMAL(10, 2),
    Contactno VARCHAR(15),
    City VARCHAR(50),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

SELECT E.Ename, D.Dname
FROM Employee E
JOIN Department D ON E.DeptID = D.DeptID;

SELECT E.Ename, D.Dname, D.Location
FROM Employee E
INNER JOIN Department D ON E.DeptID = D.DeptID;

SELECT E.Ename, D.Location, D.Dname
FROM Employee E
JOIN Department D ON E.DeptID = D.DeptID;

SELECT D.Dname, E.Salary, D.Location, E.Ename
FROM Employee E
RIGHT JOIN Department D ON E.DeptID = D.DeptID;

SELECT E.Ename, D.Dname
FROM Employee E
LEFT JOIN Department D ON E.DeptID = D.DeptID
UNION
```

```
SELECT E.Ename, D.Dname
FROM Employee E
RIGHT JOIN Department D ON E.DeptID = D.DeptID;

SELECT E.Ename, D.Dname
FROM Employee E
CROSS JOIN Department D;

SELECT E1.Ename AS Employee1, E2.Ename AS Employee2, D.Dname
FROM Employee E1
JOIN Employee E2 ON E1.DeptID = E2.DeptID AND E1.EmpID < E2.EmpID
JOIN Department D ON E1.DeptID = D.DeptID;

CREATE VIEW Employee_Department_View AS
SELECT E.Ename, D.Dname, D.Location
FROM Employee E
JOIN Department D ON E.DeptID = D.DeptID;

CREATE VIEW Department_Salary_View AS
SELECT D.Dname, COUNT(E.EmpID) AS Employee_Count, AVG(E.Salary) AS Avg_Salary
FROM Department D
LEFT JOIN Employee E ON D.DeptID = E.DeptID
GROUP BY D.Dname;

UPDATE Employee_Department_View
SET Location = 'New York'
WHERE Ename = 'John Doe';

DROP VIEW IF EXISTS Employee_Department_View;
DROP VIEW IF EXISTS Department_Salary_View;
```