**1) Create the following table and perform the necessary commands given below :**

**(a) Create Table Student with following attributes:**

**1.) Roll\_No**

**2.) Name**

**3.) Date\_of\_Birth**

**4.) Branch**

**5.) Semester**

**6.) Address**

**7.) Year\_of \_Admission**

**(b) Enter at least 10 records in the above table and answer the following queries using SQL : i.) Find the name of all the students who are enrolled in EXTC branch and having date of birth as 01/01/2008**

**ii.) List the name and roll number of all the students who are enrolled in year 2015. iii.)List the name and address of all the students who are currently in fifth semester for Computer department**

**iv.)Retrieve total number of students enrolled in IT department**   
  
  
-- (a) Create Table Student

CREATE TABLE Student (

Roll\_No INT PRIMARY KEY,

Name VARCHAR(50),

Date\_of\_Birth DATE,

Branch VARCHAR(50),

Semester INT,

Address VARCHAR(100),

Year\_of\_Admission INT

);

-- (b) Insert at least 10 records

INSERT INTO Student (Roll\_No, Name, Date\_of\_Birth, Branch, Semester, Address, Year\_of\_Admission)

VALUES

(1, 'John Doe', '2000-05-15', 'EXTC', 5, '123 Main St, Mumbai', 2017),

(2, 'Jane Smith', '2001-02-20', 'IT', 4, '456 Elm St, Pune', 2016),

(3, 'Alice Johnson', '1999-10-10', 'Computer', 5, '789 Oak St, Bangalore', 2017),

(4, 'Bob Williams', '2002-08-25', 'EXTC', 3, '101 Pine St, Delhi', 2018),

(5, 'Eva Brown', '2000-12-01', 'IT', 6, '234 Cedar St, Chennai', 2015),

(6, 'Michael Lee', '1998-06-30', 'Computer', 2, '345 Maple St, Hyderabad', 2019),

(7, 'Sarah Davis', '2003-04-05', 'EXTC', 4, '567 Birch St, Kolkata', 2016),

(8, 'David Wilson', '2001-09-12', 'IT', 7, '789 Pine St, Mumbai', 2015),

(9, 'Sophia Taylor', '1999-11-20', 'Computer', 5, '890 Elm St, Pune', 2017),

(10, 'Matthew Anderson', '2002-03-18', 'EXTC', 3, '112 Oak St, Delhi', 2018);

-- (b) (i) Find the name of all the students who are enrolled in EXTC branch and having date of birth as 01/01/2008

SELECT Name

FROM Student

WHERE Branch = 'EXTC' AND Date\_of\_Birth = '2008-01-01';

-- (b) (ii) List the name and roll number of all the students who are enrolled in year 2015

SELECT Roll\_No, Name

FROM Student

WHERE Year\_of\_Admission = 2015;

-- (b) (iii) List the name and address of all the students who are currently in fifth semester for Computer department

SELECT Name, Address

FROM Student

WHERE Semester = 5 AND Branch = 'Computer';

-- (b) (iv) Retrieve total number of students enrolled in IT department

SELECT COUNT(\*)

FROM Student

WHERE Branch = 'IT';

**2) Write a query in SQL to create a table employee and department.**

**Employee(empno,ename,deptno,job,hiredate)**

**Department(deptno,dname,loc)**

**A. Include the following constraints on column of Employee table.**

**i) make empno as primary key of the table**

**ii) ename attribute does not contain NULL values**

**iii) job attribute allow only UPPERCASE entries**

**iv) put the current date as default date in hire date column in case data is not supplied for the column.**

**B. Include the following constraints on column of dept table.**

**i) make deptno as primary key.**

**ii) dname,loc attributes does not contain NULL values**

**iii)enforce REFERENTIAL INTEGRITY where deptno attribute of dept table as primary key and deptno attribute of emp table as foreign key.**

**iv) put default value ofloc as “Mumbai”**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY, -- (A.i) Primary key constraint

ename VARCHAR(50) NOT NULL, -- (A.ii) Not NULL constraint

deptno INT,

job VARCHAR(50) CHECK (job = UPPER(job)), -- (A.iii) Check constraint for uppercase

hiredate DATE DEFAULT CURRENT\_DATE -- (A.iv) Default value constraint

);

-- Create Department table

CREATE TABLE Department (

deptno INT PRIMARY KEY, -- (B.i) Primary key constraint

dname VARCHAR(50) NOT NULL, -- (B.ii) Not NULL constraint

loc VARCHAR(50) DEFAULT 'Mumbai' -- (B.iv) Default value constraint

);

-- Add FOREIGN KEY constraint for REFERENTIAL INTEGRITY

ALTER TABLE Employee

ADD CONSTRAINT fk\_deptno

FOREIGN KEY (deptno) REFERENCES Department(deptno); -- (B.iii) Referential integrity constraint

**3) Write a query in SQL to create a table employee and department with following attributes. Employee(empno,ename,deptno,job,hiredate)**

**Department(deptno,dname,loc)**

**1. Give list of emp name & their job spec who are working in dept no 20?**

**2. Retrieve the details of emp working in dept no 30 ?**

**3. Find list of emp whose empno is greater then manager no ?**

**4. Find all manager not working in dept no 10 ?**

**5. To find the total number of employees.**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY,

ename VARCHAR(50),

deptno INT,

job VARCHAR(50),

hiredate DATE

);

-- Create Department table

CREATE TABLE Department (

deptno INT PRIMARY KEY,

dname VARCHAR(50),

loc VARCHAR(50)

);

-- Query 1: List of emp name & their job spec who are working in dept no 20

SELECT ename, job

FROM Employee

WHERE deptno = 20;

-- Query 2: Retrieve the details of emp working in dept no 30

SELECT \*

FROM Employee

WHERE deptno = 30;

-- Query 3: Find list of emp whose empno is greater than manager no

SELECT \*

FROM Employee

WHERE empno > (SELECT MAX(empno) FROM Employee);

-- Query 4: Find all manager not working in dept no 10

SELECT \*

FROM Employee

WHERE job = 'Manager' AND deptno <> 10;

-- Query 5: Total number of employees

SELECT COUNT(\*) AS TotalEmployees

FROM Employee;

**4) Write a query in sql to create a table employee and department.**

**Employee(empno,ename,deptno,job,hiredate)**

**Department(deptno,dname,loc)**

**1. To find the total number of clerk hired after 13-Jan-2001.**

**2. Determine which department having more than two people holding a same job?**

**3. Find all departments that have at least two clerks?**

**4. Retrieve emp name and job who have the same job as that of „Allen‟?**

**5. List all emp name and their job of those department that are located at Chicago?**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY,

ename VARCHAR(50),

deptno INT,

job VARCHAR(50),

hiredate DATE

);

-- Create Department table

CREATE TABLE Department (

deptno INT PRIMARY KEY,

dname VARCHAR(50),

loc VARCHAR(50)

);

-- Query 1: Total number of clerks hired after 13-Jan-2001

SELECT COUNT(\*) AS TotalClerks

FROM Employee

WHERE job = 'Clerk' AND hiredate > '2001-01-13';

-- Query 2: Departments having more than two people holding the same job

SELECT deptno, job, COUNT(\*) AS TotalEmployees

FROM Employee

GROUP BY deptno, job

HAVING COUNT(\*) > 2;

-- Query 3: Departments that have at least two clerks

SELECT deptno

FROM Employee

WHERE job = 'Clerk'

GROUP BY deptno

HAVING COUNT(\*) >= 2;

-- Query 4: Emp name and job having the same job as Allen

SELECT ename, job

FROM Employee

WHERE job = (SELECT job FROM Employee WHERE ename = 'Allen');

-- Query 5: Emp name and job of departments located at Chicago

SELECT e.ename, e.job

FROM Employee e

JOIN Department d ON e.deptno = d.deptno

WHERE d.loc = 'Chicago';

**5) Write a query in sql to create a table employee and department.**

**Employee(empno,ename,deptno,job,hiredate,salary)**

**Department(deptno,dname,loc)**

**1. To get all employees working for dept 10 and 20.**

**2. To list all employees whose name begins with „J‟.**

**3. Retrieve all details of employees whose name is either Smith, Blake, Allen, Scott, Clark and King ? 4. Create view on appropriate tables to display ename , job , salary , dept name?**

**5. Drop the above view**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY,

ename VARCHAR(50),

deptno INT,

job VARCHAR(50),

hiredate DATE,

salary DECIMAL(10, 2)

);

-- Create Department table

CREATE TABLE Department (

deptno INT PRIMARY KEY,

dname VARCHAR(50),

loc VARCHAR(50)

);

-- Query 1: Get all employees working for dept 10 and 20

SELECT \*

FROM Employee

WHERE deptno IN (10, 20);

-- Query 2: List all employees whose name begins with 'J'

SELECT \*

FROM Employee

WHERE ename LIKE 'J%';

-- Query 3: Retrieve all details of employees with specific names

SELECT \*

FROM Employee

WHERE ename IN ('Smith', 'Blake', 'Allen', 'Scott', 'Clark', 'King');

-- Query 4: Create a view to display employee name, job, salary, and department name

CREATE VIEW EmployeeDetails AS

SELECT e.ename, e.job, e.salary, d.dname AS dept\_name

FROM Employee e

JOIN Department d ON e.deptno = d.deptno;

-- Query 5: Drop the view

DROP VIEW EmployeeDetails;

**6) Write a query in sql to create a table employee and department.**

**Employee(empno,ename,deptno,job,hiredate,salary)**

**Department(deptno,dname,loc)**

**1. To select the employees whose salary is greater than the salary of all employees working in dept no. 30 2. To list all employees in the ascending order by name.**

**3. To select all employees sorted dept wise in ascending order and within dept salary wise in descending order.**

**4. To select all employees working in location whose name is starting with L**

**5. To find the minimum salary of managers in various depts.**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY,

ename VARCHAR(50),

deptno INT,

job VARCHAR(50),

hiredate DATE,

salary DECIMAL(10, 2)

);

-- Create Department table

CREATE TABLE Department (

deptno INT PRIMARY KEY,

dname VARCHAR(50),

loc VARCHAR(50)

);

-- Query 1: Select employees with salary greater than the max salary in dept no. 30

SELECT \*

FROM Employee

WHERE salary > (SELECT MAX(salary) FROM Employee WHERE deptno = 30);

-- Query 2: List all employees in ascending order by name

SELECT \*

FROM Employee

ORDER BY ename ASC;

-- Query 3: Select employees sorted by department in ascending order and within each department by salary in descending order

SELECT \*

FROM Employee

ORDER BY deptno ASC, salary DESC;

-- Query 4: Select employees working in a location starting with 'L'

SELECT \*

FROM Employee e

JOIN Department d ON e.deptno = d.deptno

WHERE d.loc LIKE 'L%';

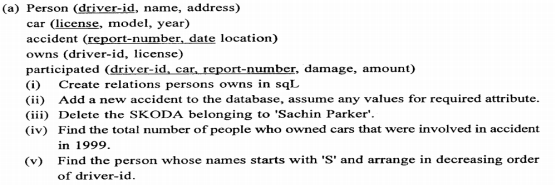
-- Query 5: Find the minimum salary of managers in various departments

SELECT MIN(salary) AS min\_manager\_salary

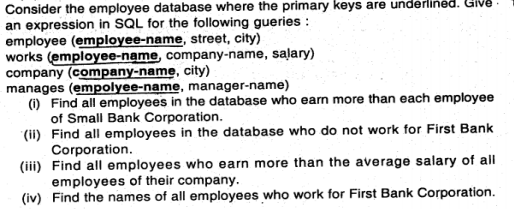
FROM Employee

WHERE job = 'Manager';

7)



8)



select \* from employee where salary > IN ( select salary from works where companyname ="Small bank corporation")

select \* from employee where employee\_name IN ( select employee\_name from works where company\_name <> "First bank corporation");

select employee\_name from employee where employee\_name IN ( select employee\_name from works where company\_name = "First bank corporation");

**9) Create table employee with following attributes and insert 10 record (Apply following query) Employee(empno,ename,deptno,job,hiredate,salary)**

**Department(deptno,dname,loc)**

**1. To list employees whose name begins with „J‟ and has „N‟ as the third character? 2. To list all employees not entitled for commission.**

**3. To get all the employees whose salary is greater than the average salary of the company. 4. To find out average minimum and maximum salary of each dept.**

**5. Create view on emp to display sum of salary grouped according to deptno**

-- Create Employee table

CREATE TABLE Employee (

empno INT PRIMARY KEY,

ename VARCHAR(50),

deptno INT,

job VARCHAR(50),

hiredate DATE,

salary DECIMAL(10, 2),

commission DECIMAL(10, 2)

);

-- Insert 10 records into the Employee table

INSERT INTO Employee (empno, ename, deptno, job, hiredate, salary, commission)

VALUES

(1, 'John', 10, 'Manager', '2022-01-01', 5000.00, 1000.00),

(2, 'Jane', 20, 'Clerk', '2022-02-01', 3000.00, NULL),

(3, 'Jim', 30, 'Manager', '2022-03-01', 5500.00, 500.00),

(4, 'Jack', 10, 'Analyst', '2022-04-01', 4000.00, NULL),

(5, 'Jill', 20, 'Clerk', '2022-05-01', 3200.00, NULL),

(6, 'Nick', 30, 'Analyst', '2022-06-01', 4500.00, NULL),

(7, 'Nancy', 10, 'Clerk', '2022-07-01', 2800.00, NULL),

(8, 'Jake', 20, 'Analyst', '2022-08-01', 4200.00, NULL),

(9, 'Noah', 30, 'Clerk', '2022-09-01', 3300.00, NULL),

(10, 'Nina', 10, 'Manager', '2022-10-01', 6000.00, 1200.00);

-- Query 1: List employees whose name begins with 'J' and has 'N' as the third character

SELECT \*

FROM Employee

WHERE ename LIKE 'J\_N%';

-- Query 2: List all employees not entitled for commission

SELECT \*

FROM Employee

WHERE commission IS NULL;

-- Query 3: Get employees whose salary is greater than the average salary of the company

SELECT \*

FROM Employee

WHERE salary > (SELECT AVG(salary) FROM Employee);

-- Query 4: Find average, minimum, and maximum salary of each department

SELECT deptno,

AVG(salary) AS avg\_salary,

MIN(salary) AS min\_salary,

MAX(salary) AS max\_salary

FROM Employee

GROUP BY deptno;

-- Query 5: Create a view to display sum of salary grouped according to deptno

CREATE VIEW SalarySumByDept AS

SELECT deptno, SUM(salary) AS total\_salary

FROM Employee

GROUP BY deptno;

10) Consider the following schema:

Suppliers(sid: integer, sname: string, address: string)

Parts(pid: integer, pname: string, color: string)

Catalog(sid: integer, pid: integer, cost: real)

The Catalog relation lists the prices charged for parts by Suppliers.

Write the following queries in SQL:

1. For each part, find the sname of the supplier who charges the most for that part. 2. Find the sids of suppliers who supply only red parts.

3. Find the sids of suppliers who supply a red part and a green part.

4. Find the snames of suppliers who supply every red part.

5. Find the pnames of parts supplied by Acme Widget Suppliers and no one else.

**11) Create a table customer(acc\_no, cust\_name, avail\_balance)**

**Create table mini\_statement(acc\_no, avail\_balance)**

**Insert into customer following records:**

**Customer(1000, “Fanny”, 7000);**

**Customer(1001,”Peter”, 12000);**

**Write a trigger to insert old values into mini\_statement table(including acc\_no, avail\_balance as parameters) before updating any record in customer table**

**12) Create a table customer(acc\_no, cust\_name, avail\_balance)**

**Create table micro\_statement(acc\_no, avail\_balance)**

**Insert following record in table customer:**

**Customer(1000, “Fanny”, 7000);**

**Customer(1001,”Peter”, 12000);**

**Customer(1002,”Janitor”, 4500)**

**Write a trigger to insert new values of acc\_no and avail\_balance in micro\_statement after an update has occurred**

CREATE TRIGGER before\_customer\_update

BEFORE UPDATE ON customer

FOR EACH ROW

BEGIN

INSERT INTO mini\_statement (acc\_no, avail\_balance) VALUES (OLD.acc\_no, OLD.avail\_balance);

END;

CREATE TRIGGER after\_customer\_update

AFTER UPDATE ON customer

FOR EACH ROW

BEGIN

INSERT INTO micro\_statement (acc\_no, avail\_balance) VALUES (NEW.acc\_no, NEW.avail\_balance);

END;

13) Create a table customer (cust\_id, cust\_name, balance) and insert 3 records to it

Write a transaction which update the balance of all 3 customers and using TCL Commands(Commit, Rollback and Savepoint) show the changes made to actual records

**14) Create a table student (student\_id, stud\_name, percentage) and insert 3 records to it Write a transaction which update the percentage of all 3 students and using TCL Commands(Commit, Rollback and Savepoint) show the changes made to actual records**

CREATE TABLE customer (

cust\_id INT PRIMARY KEY,

cust\_name VARCHAR(50),

balance DECIMAL(10, 2)

);

INSERT INTO customer (cust\_id, cust\_name, balance)

VALUES

(1, 'Alice', 5000),

(2, 'Bob', 7000),

(3, 'Charlie', 9000);

-- Start the transaction

START TRANSACTION;

-- Update the balances of all customers

UPDATE customer SET balance = balance + 1000;

-- Display the updated records

SELECT \* FROM customer;

-- Savepoint to capture changes

SAVEPOINT after\_update;

-- Rollback to the savepoint to undo the changes

ROLLBACK TO after\_update;

-- Display the original records

SELECT \* FROM customer;

-- Commit the transaction to make changes permanent

COMMIT;

**15) Consider the relational database. Write an expression in SQL for following schema employee(employee-name, street, city)**

**works(employee-name, company-name, salary)**

**company(company-name, city)**

**manages(employee-name,manager-name)**

**Write following SQL queries**

**i) Retrieve details of all employees working for “Infosys” company**

SELECT e.\*

FROM employee e

JOIN works w ON e.employee\_name = w.employee\_name

WHERE w.company\_name = 'Infosys';

**ii) Retrieve employee-name in uppercase for all employees**

select UPPER(employee\_name) from employee

**iii) Replace existing company name of employees from Infosys to TCS**

UPDATE works

SET company\_name = 'TCS'

WHERE company\_name = 'Infosys';

**iv) Retrieve manager name along with employee name working for “TCS” Company**

SELECT e.employee\_name, m.manager\_name

FROM employee e

JOIN works w ON e.employee\_name = w.employee\_name

JOIN manages m ON e.employee\_name = m.employee\_name

WHERE w.company\_name = 'TCS';

**v) Retrieve details of all employees whose ename starts with “P**”

SELECT \*

FROM employee

WHERE employee\_name LIKE 'P%';

16) Consider an online bookstore database with the following tables:

∙ books: Contains information about books such as book\_id, title, author\_id, genre\_id, and price. ∙ authors: Contains information about authors such as author\_id, author\_name, and country. ∙ genres: Contains information about book genres such as genre\_id and genre\_name. ∙ customers: Contains information about customers such as customer\_id, name, email, and city. ∙ orders: Contains information about orders such as order\_id, customer\_id, order\_date, and total\_amount. ∙ order\_details: Contains information about the details of each order such as order\_detail\_id, order\_id, book\_id, quantity, and subtotal.

1. Get the list of books with their authors and genres

2. Get the total amount spent by each customer:

3. Get the list of customers along with the titles of books they have ordered

4. Get the top-selling authors (authors with the highest total number of book sales)

**17) We have a database for an online bookstore with relation books and customer, and we want to grant specific privileges to different users.**

**∙ Create a new user named 'bookstore\_manager' with a password.**

CREATE USER 'bookstore\_manager'@'localhost' IDENTIFIED BY 'password';

**∙ Grant the SELECT privilege on the 'books' table to the 'bookstore\_manager' user, allowing them to retrieve data from the 'books' table.**

GRANT SELECT ON bookstore.books TO 'bookstore\_manager'@'localhost';

**∙ Grant the INSERT, UPDATE, and DELETE privileges on the 'customers' table to the 'bookstore\_manager' user, allowing them to insert, update, and delete records in the 'customers' table.**GRANT INSERT, UPDATE, DELETE ON bookstore.customers TO 'bookstore\_manager'@'localhost';

18) Let's consider a scenario where a customer places an order on our online bookstore. We want to ensure that the order process is treated as a single transaction. Apply TCL commands using following steps ∙ We begin a transaction using the BEGIN TRANSACTION command.

∙ We insert the order details (such as customer ID, order date, and total amount) into the 'orders' table. ∙ We insert the individual items of the order (book ID, quantity, and subtotal) into the 'order\_details' table. ∙ We perform a check to ensure that the total amount matches the sum of individual subtotals. If the validation fails, we rollback the transaction using the ROLLBACK command.

∙ If the validation succeeds, we commit the transaction using the COMMIT command.