

DBMS Lab

Assignment 1

```
CREATE DATABASE assignment1; USE
assignment1;

-- Q1: Create Tables with Proper Primary and Foreign Keys
CREATE TABLE employee (
    employee_name VARCHAR(50) PRIMARY KEY, street
    VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE company (
    company_name VARCHAR(50) PRIMARY KEY, city
    VARCHAR(50)
);

CREATE TABLE works (
    employee_name VARCHAR(50),
    company_name VARCHAR(50), salary
    DECIMAL(10, 2),
    PRIMARY KEY (employee_name, company_name),
    FOREIGN KEY (employee_name) REFERENCES
employee(employee_name),
    FOREIGN KEY (company_name) REFERENCES company(company_name)
);

CREATE TABLE manages ( employee_name
    VARCHAR(50),
```

```
manager_name VARCHAR(50),  
PRIMARY KEY (employee_name, manager_name),  
FOREIGN KEY (employee_name) REFERENCES  
employee(employee_name),  
FOREIGN KEY (manager_name) REFERENCES  
employee(employee_name)  
);
```

-- Q2: Insert Records into Each Table

```
INSERT INTO employee (employee_name, street, city)  
VALUES
```

```
('Raj Shukla', 'Street A', 'Pune'),  
( 'Sandeep Patil', 'Street B', 'Mumbai'),  
( 'Shital Sonje', 'Street C', 'Nashik'),  
( 'Anita Sharma', 'Street D', 'Delhi'),  
( 'Ravi Verma', 'Street E', 'Kolkata'),  
( 'Pooja Singh', 'Street F', 'Pune'),  
( 'Vikas Gupta', 'Street G', 'Mumbai'),  
( 'Amit Kumar', 'Street H', 'Nashik'),  
( 'Sunita Desai', 'Street I', 'Delhi'),  
( 'Manoj Mehta', 'Street J', 'Kolkata');
```

```
INSERT INTO company (company_name, city) VALUES
```

```
('First Bank Corporation', 'Pune'),  
( 'Small Bank Corporation', 'Mumbai'),  
( 'Special Bank Corporation', 'Nashik');
```

```
INSERT INTO works (employee_name, company_name,  
salary) VALUES
```

```
('Raj Shukla', 'First Bank Corporation', 60000),  
( 'Sandeep Patil', 'Small Bank Corporation', 45000),  
( 'Shital Sonje', 'Special Bank Corporation', 50000),  
( 'Anita Sharma', 'First Bank Corporation', 55000),
```

```
('Ravi Verma', 'First Bank Corporation', 65000),  
( 'Pooja Singh', 'Small Bank Corporation', 48000),  
( 'Vikas Gupta', 'Special Bank Corporation', 52000),  
( 'Amit Kumar', 'First Bank Corporation', 58000),  
( 'Sunita Desai', 'Small Bank Corporation', 47000),  
( 'Manoj Mehta', 'Special Bank Corporation', 53000);
```

```
INSERT INTO manages (employee_name, manager_name)  
VALUES
```

```
('Raj Shukla', 'Anita Sharma'),  
( 'Sandeep Patil', 'Ravi Verma'),  
( 'Shital Sonje', 'Pooja Singh'),  
( 'Anita Sharma', 'Ravi Verma'),  
( 'Ravi Verma', 'Anita Sharma');
```

```
-- Q3: Add Birthdate Column to Employee Table
```

```
ALTER TABLE employee ADD COLUMN birthdate DATE;
```

```
-- Q4: Update Birthdate of All Employees
```

```
UPDATE employee SET birthdate = '1980-01-01' WHERE  
employee_name = 'Raj Shukla';
```

```
UPDATE employee SET birthdate = '1982-02-15' WHERE  
employee_name = 'Sandeep Patil';
```

```
UPDATE employee SET birthdate = '1985-03-20' WHERE  
employee_name = 'Shital Sonje';
```

```
UPDATE employee SET birthdate = '1990-04-10' WHERE  
employee_name = 'Anita Sharma';
```

```
UPDATE employee SET birthdate = '1983-05-25' WHERE  
employee_name = 'Ravi Verma';
```

```
UPDATE employee SET birthdate = '1988-06-30' WHERE  
employee_name = 'Pooja Singh';
```

```
UPDATE employee SET birthdate = '1979-07-18' WHERE  
employee_name = 'Vikas Gupta';
```

```
UPDATE employee SET birthdate = '1984-08-12' WHERE
employee_name = 'Amit Kumar';
UPDATE employee SET birthdate = '1987-09-22' WHERE
employee_name = 'Sunita Desai';
UPDATE employee SET birthdate = '1992-10-05' WHERE
employee_name = 'Manoj Mehta';

-- Q5: Delete Specified Employees
DELETE FROM works WHERE employee_name IN ('Raj
Shukla', 'Sandeep Patil', 'Shital Sonje');
DELETE FROM manages WHERE employee_name IN ('Raj
Shukla', 'Sandeep Patil', 'Shital Sonje');
DELETE FROM employee WHERE employee_name IN ('Raj
Shukla', 'Sandeep Patil', 'Shital Sonje');
-- Ensure the employees to be deleted do not have
foreign key dependencies in other tables

-- Q6: Rename Salary to Monthly Salary
ALTER TABLE works CHANGE COLUMN salary monthly_salary
DECIMAL(10, 2);

-- Q7: Create Dependants Table
CREATE TABLE dependant (
    employee_name VARCHAR(50),
    dependant_name VARCHAR(50),
    relation VARCHAR(50),
    PRIMARY KEY (employee_name, dependant_name),
    FOREIGN KEY (employee_name) REFERENCES
employee(employee_name)
);

-- Q8: Add Rows to Dependants Table
INSERT INTO dependant (employee_name, dependant_name,
relation) VALUES
```

```
('Anita Sharma', 'Amit Sharma', 'Spouse'),  
( 'Ravi Verma', 'Rekha Verma', 'Spouse'),  
( 'Pooja Singh', 'Anil Singh', 'Spouse'),  
( 'Vikas Gupta', 'Vandana Gupta', 'Spouse'),  
( 'Amit Kumar', 'Neha Kumar', 'Spouse');
```

```
-- Q9: Difference Between TRUNCATE and DELETE  
-- No SQL code, just explanation:  
-- TRUNCATE: Removes all rows from a table without  
logging individual row deletions. It's faster and  
uses fewer system resources.  
-- DELETE: Removes rows one at a time and logs each  
row deletion. It can be used with a WHERE clause to  
delete specific rows.
```

```
-- Q10: Single-Select Queries
```

```
-- a. Find the names of employees starting with "S"  
SELECT employee_name FROM employee WHERE  
employee_name LIKE 'S%';
```

```
-- b. Find the names of employees who work for First  
Bank Corporation
```

```
SELECT employee_name FROM works WHERE company_name =  
'First Bank Corporation';
```

```
-- c. Find the names of the companies located in  
"Pune"
```

```
SELECT company_name FROM company WHERE city = 'Pune';
```

```
-- d. Find the age of employees
```

```
SELECT employee_name, YEAR(CURDATE()) -  
YEAR(birthdate) AS age FROM employee;
```

-- e. Find the names and cities of residence of all employees who work for First Bank Corporation

```
SELECT e.employee_name, e.city FROM employee e JOIN  
works w ON e.employee_name = w.employee_name WHERE  
w.company_name = 'First Bank Corporation';
```

-- f. Find the salaries of the employees whose salary is greater than 50000

```
SELECT employee_name, monthly_salary FROM works WHERE  
monthly_salary > 50000;
```

-- g. Find the employee names, street addresses, cities of residence, and company names of all employees

```
SELECT e.employee_name, e.street, e.city,  
w.company_name FROM employee e JOIN works w ON  
e.employee_name = w.employee_name;
```

-- h. Find the manager of "Sandeep Patil"

```
SELECT manager_name FROM manages WHERE employee_name  
= 'Sandeep Patil';
```

-- i. Find the age of all employees

```
SELECT employee_name, YEAR(CURDATE()) -  
YEAR(birthdate) AS age FROM employee;
```

-- j. Find the names of the employees having their birthday in January

```
SELECT employee_name FROM employee WHERE  
MONTH(birthdate) = 1;
```

-- Q11: Five More Queries with Date, String, and Other Functions

-- a. Find employees whose names contain 'a'

```

SELECT employee_name FROM employee WHERE
employee_name LIKE '%a%';

-- b. Calculate the average salary of employees
SELECT AVG(monthly_salary) AS avg_salary FROM works;

-- c. Count the number of employees in each company
SELECT company_name, COUNT(employee_name) AS
num_employees FROM works GROUP BY company_name;

-- d. Find the length of each employee's name
SELECT employee_name, LENGTH(employee_name) AS
name_length FROM employee;

-- e. Find employees hired before a specific date
-- Assuming there is a hire_date column in the works
table
ALTER TABLE works ADD COLUMN hire_date DATE;

UPDATE works SET hire_date = '2020-01-01' WHERE
employee_name = 'Anita Sharma';
-- Repeat the update for other employees as needed

SELECT employee_name FROM works WHERE hire_date <
'2021-01-01';

```

Assignment 2

```

tee assignment2.txt
CREATE DATABASE assignment2;
USE assignment2;

-- Create tables

```

```
CREATE TABLE instructor (  
    instructor_id VARCHAR(5),  
    instructor_name VARCHAR(90),  
    instructor_city VARCHAR(25),  
    specialization VARCHAR(30),  
    PRIMARY KEY (instructor_id)  
);
```

```
CREATE TABLE student (  
    student_id INT,  
    student_name VARCHAR(60),  
    instructor_id VARCHAR(5),  
    student_city VARCHAR(35),  
    FOREIGN KEY (instructor_id) REFERENCES  
instructor(instructor_id)  
);
```

-- Insert data into instructor table

```
INSERT INTO instructor VALUES  
( '1001', 'Ashwin', 'Maharashtra', 'Science'),  
( '1002', 'Nina', 'Maharashtra', 'Arts'),  
( '1003', 'Ravi', 'Tamil Nadu', 'Commerce'),  
( '1004', 'Anil', 'Uttar Pradesh', 'Science'),  
( '1005', 'Sita', 'Telangana', 'Arts'),  
( '1006', 'Sanjay', 'Gujarat', 'Commerce'),  
( '1007', 'Arun', 'Punjab', 'Science'),  
( '1008', 'Meera', 'Kerala', 'Arts'),  
( '1009', 'Rajesh', 'Rajasthan', 'Commerce'),  
( '1010', 'Kumar', 'Madhya Pradesh', 'Science'),  
( '1011', 'Neelam', 'Odisha', 'Arts'),  
( '1012', 'Deepak', 'West Bengal', 'Commerce'),  
( '1013', 'Sunil', 'Haryana', 'Science'),  
( '1014', 'Pooja', 'Bihar', 'Arts'),  
( '1015', 'Suresh', 'Assam', 'Commerce'),
```



```
('1016', 'Gaurav', 'Chhattisgarh', 'Science'),  
( '1017', 'Ritika', 'Jharkhand', 'Arts'),  
( '1018', 'Kiran', 'Goa', 'Commerce'),  
( '1019', 'Vikram', 'Himachal Pradesh', 'Science'),  
( '1020', 'Asha', 'Uttarakhand', 'Arts');
```

```
-- Insert data into student table
```

```
INSERT INTO student VALUES
```

```
(2, 'Asma', '1001', 'Karnataka'),  
(3, 'Rahul', '1002', 'Maharashtra'),  
(4, 'Priya', '1003', 'Tamil Nadu'),  
(5, 'Amit', '1004', 'Uttar Pradesh'),  
(6, 'Sneha', '1005', 'Telangana'),  
(7, 'Vikram', '1006', 'Gujarat'),  
(8, 'Neha', '1007', 'Punjab'),  
(9, 'Sanjay', '1008', 'Kerala'),  
(10, 'Meera', '1009', 'Rajasthan'),  
(11, 'Ravi', '1010', 'Madhya Pradesh'),  
(12, 'Pooja', '1011', 'Odisha'),  
(13, 'Kiran', '1012', 'West Bengal'),  
(14, 'Deepak', '1013', 'Haryana'),  
(15, 'Anjali', '1014', 'Bihar'),  
(16, 'Rahul', '1015', 'Assam'),  
(17, 'Jaya', '1016', 'Chhattisgarh'),  
(18, 'Vinay', '1017', 'Jharkhand'),  
(19, 'Swati', '1018', 'Goa'),  
(20, 'Raj', '1019', 'Himachal Pradesh'),  
(21, 'Asha', '1020', 'Uttarakhand');
```

```
-- 1. Find the instructor of each student.
```

```
SELECT student.student_name,  
instructor.instructor_name  
FROM student
```

```
JOIN instructor ON student.instructor_id =  
instructor.instructor_id;
```

```
-- 2. Find the student who is not having any  
instructor.
```

```
SELECT student_name  
FROM student  
WHERE instructor_id IS NULL;
```

```
-- 3. Find the student who is not having any  
instructor as well as the instructor who is not  
having a student.
```

```
-- Students without an instructor
```

```
SELECT student_name  
FROM student  
WHERE instructor_id IS NULL;
```

```
-- Instructors without students
```

```
SELECT instructor_name  
FROM instructor  
WHERE instructor_id NOT IN (SELECT instructor_id FROM  
student);
```

```
-- 4. List students and their instructors for a  
specific specialization (e.g., 'Science').
```

```
SELECT student.student_name,  
instructor.instructor_name  
FROM student  
JOIN instructor ON student.instructor_id =  
instructor.instructor_id  
WHERE instructor.specialization = 'Science';
```

```
-- 5. Develop SQL query to find students whose city  
does not match with their instructor.
```

```
SELECT student.student_name, student.student_city,  
instructor.instructor_name,  
instructor.instructor_city  
FROM student  
JOIN instructor ON student.instructor_id =  
instructor.instructor_id  
WHERE student.student_city <>  
instructor.instructor_city;
```

-- 6. List All Instructors and Their Students
(Including Without Students).

```
SELECT instructor.instructor_name,  
student.student_name  
FROM instructor  
LEFT JOIN student ON instructor.instructor_id =  
student.instructor_id  
ORDER BY instructor.instructor_name;
```

-- 7. List of students of each instructor who stays
in Pune (Instructor).

```
SELECT student.student_name,  
instructor.instructor_name  
FROM student  
JOIN instructor ON student.instructor_id =  
instructor.instructor_id  
WHERE instructor.instructor_city = 'Pune';
```

-- 8. Calculate the Average Number of Students per
Instructor in Nashik.

```
SELECT AVG(student_count)  
FROM (  
    SELECT COUNT(student.student_id) AS student_count  
    FROM instructor
```

```
    LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
    WHERE instructor.instructor_city = 'Nashik'
    GROUP BY instructor.instructor_id
) AS avg_students;
```

-- 9. Create a view containing the total number of students whose instructor belongs to 'Pune'.

```
CREATE VIEW StudentsWithPuneInstructors AS
SELECT COUNT(student.student_id) AS total_students
FROM student
JOIN instructor ON student.instructor_id =
instructor.instructor_id
WHERE instructor.instructor_city = 'Pune';
```

-- 10. Write the following queries on the view:

-- a. List All Specializations of Instructors in Pune

```
SELECT DISTINCT specialization
FROM instructor
WHERE instructor_city = 'Pune';
```

-- b. Count Students for Each Instructor in Pune

```
SELECT instructor.instructor_name,
COUNT(student.student_id) AS student_count
FROM instructor
LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
WHERE instructor.instructor_city = 'Pune'
GROUP BY instructor.instructor_name;
```

-- c. List the Number of Students Grouped by Specialization of Instructors in Pune

```
SELECT instructor.specialization,
COUNT(student.student_id) AS student_count
```

```
FROM instructor
LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
WHERE instructor.instructor_city = 'Pune'
GROUP BY instructor.specialization;
```

-- d. List Instructors from Pune with No Students

```
SELECT instructor.instructor_name
FROM instructor
LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
WHERE instructor.instructor_city = 'Pune' AND
student.student_id IS NULL;
```

-- Design 5 more queries having join of 3 tables and
aggregate functions as per your choice:

-- Note: Since there are only two tables given
(student and instructor), the "join of 3 tables" part
cannot be fulfilled.

-- However, I'll provide aggregate functions
involving these two tables.

-- 1. Find the number of students each instructor has
in each specialization.

```
SELECT instructor.instructor_name,
instructor.specialization, COUNT(student.student_id)
AS student_count
FROM instructor
LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
GROUP BY instructor.instructor_name,
instructor.specialization;
```

```
-- 2. List the instructors who have more than 5
students and their specialization.
SELECT instructor.instructor_name,
instructor.specialization, COUNT(student.student_id)
AS student_count
FROM instructor
LEFT JOIN student ON instructor.instructor_id =
student.instructor_id
GROUP BY instructor.instructor_name,
instructor.specialization
HAVING student_count > 5;

-- 3. Find the total number of students in each city
for each specialization.
SELECT instructor.specialization,
student.student_city, COUNT(student.student_id) AS
total_students
FROM instructor
JOIN student ON instructor.instructor_id =
student.instructor_id
GROUP BY instructor.specialization,
student.student_city;

-- 4. List instructors along with the total number of
students in each city.
SELECT instructor.instructor_name,
student.student_city, COUNT(student.student_id) AS
total_students
FROM instructor
JOIN student ON instructor.instructor_id =
student.instructor_id
GROUP BY instructor.instructor_name,
student.student_city;
```

```
-- 5. Find the city with the highest number of
students for each specialization.
SELECT instructor.specialization,
student.student_city, COUNT(student.student_id) AS
total_students
FROM instructor
JOIN student ON instructor.instructor_id =
student.instructor_id
GROUP BY instructor.specialization,
student.student_city
ORDER BY instructor.specialization, total_students
DESC;
```

Assignment 3

```
CREATE DATABASE assignment3;
USE assignment3;

CREATE TABLE employee (
    employee_name VARCHAR(50) PRIMARY KEY,
    street VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE company (
    company_name VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50)
```

```
);

CREATE TABLE works (
    employee_name VARCHAR(50),
    company_name VARCHAR(50),
    salary DECIMAL(10, 2),
    PRIMARY KEY (employee_name, company_name),
    FOREIGN KEY (employee_name) REFERENCES
employee(employee_name),
    FOREIGN KEY (company_name) REFERENCES
company(company_name)
);

CREATE TABLE manages (
    employee_name VARCHAR(50),
    manager_name VARCHAR(50),
    PRIMARY KEY (employee_name, manager_name),
    FOREIGN KEY (employee_name) REFERENCES
employee(employee_name),
    FOREIGN KEY (manager_name) REFERENCES
employee(employee_name)
);

INSERT INTO employee (employee_name, street, city)
VALUES
('Raj Shukla', 'Street A', 'Pune'),
('Sandeep Patil', 'Street B', 'Mumbai'),
('Shital Sonje', 'Street C', 'Nashik'),
('Anita Sharma', 'Street D', 'Delhi'),
('Ravi Verma', 'Street E', 'Kolkata'),
('Pooja Singh', 'Street F', 'Pune'),
('Vikas Gupta', 'Street G', 'Mumbai'),
('Amit Kumar', 'Street H', 'Nashik'),
('Sunita Desai', 'Street I', 'Delhi'),
```



```
('Manoj Mehta', 'Street J', 'Kolkata');

INSERT INTO company (company_name, city) VALUES
('First Bank Corporation', 'Pune'),
('Small Bank Corporation', 'Mumbai'),
('Special Bank Corporation', 'Nashik');

INSERT INTO works (employee_name, company_name,
salary) VALUES
('Raj Shukla', 'First Bank Corporation', 60000),
('Sandeep Patil', 'Small Bank Corporation', 45000),
('Shital Sonje', 'Special Bank Corporation', 50000),
('Anita Sharma', 'First Bank Corporation', 55000),
('Ravi Verma', 'First Bank Corporation', 65000),
('Pooja Singh', 'Small Bank Corporation', 48000),
('Vikas Gupta', 'Special Bank Corporation', 52000),
('Amit Kumar', 'First Bank Corporation', 58000),
('Sunita Desai', 'Small Bank Corporation', 47000),
('Manoj Mehta', 'Special Bank Corporation', 53000);

INSERT INTO manages (employee_name, manager_name)
VALUES
('Raj Shukla', 'Anita Sharma'),
('Sandeep Patil', 'Ravi Verma'),
('Shital Sonje', 'Pooja Singh'),
('Anita Sharma', 'Ravi Verma'),
('Ravi Verma', 'Anita Sharma');

-- Write SQL queries for the following

-- 1. Find the names of all employees who work for
First Bank Corporation.
SELECT employee_name
FROM works
```

```
WHERE company_name = 'First Bank Corporation';
```

-- 2. Find the names and cities of residence of all employees who work for First Bank Corporation.

```
SELECT e.employee_name, e.city  
FROM employee e  
JOIN works w ON e.employee_name = w.employee_name  
WHERE w.company_name = 'First Bank Corporation';
```

-- 3. Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than Rs.10,000.

```
SELECT e.employee_name, e.street, e.city  
FROM employee e  
JOIN works w ON e.employee_name = w.employee_name  
WHERE w.company_name = 'First Bank Corporation' AND  
w.salary > 10000;
```

-- 4. Find all employees in the database who live in the same cities as the companies for which they work.

```
SELECT e.employee_name  
FROM employee e  
JOIN works w ON e.employee_name = w.employee_name  
JOIN company c ON w.company_name = c.company_name  
WHERE e.city = c.city;
```

-- 5. Find all employees in the database who live in the same cities and on the same streets as do their managers.

```
SELECT e1.employee_name  
FROM manages m  
JOIN employee e1 ON m.employee_name =  
e1.employee_name  
JOIN employee e2 ON m.manager_name = e2.employee_name
```

```
WHERE e1.city = e2.city AND e1.street = e2.street;
```

```
-- 6. Find all employees in the database who do not  
work for First Bank Corporation.
```

```
SELECT e.employee_name  
FROM employee e  
LEFT JOIN works w ON e.employee_name =  
w.employee_name AND w.company_name = 'First Bank  
Corporation'  
WHERE w.company_name IS NULL;
```

```
-- 7. Find all employees in the database who earn  
more than each employee of Small Bank Corporation.
```

```
SELECT e.employee_name  
FROM employee e  
WHERE EXISTS (  
    SELECT *  
    FROM works w  
    WHERE w.company_name = 'Small Bank Corporation'  
    AND e.employee_name != w.employee_name  
    AND e.employee_name IN (  
        SELECT w2.employee_name  
        FROM works w2  
        WHERE w2.salary > (SELECT MAX(w3.salary) FROM  
works w3 WHERE w3.company_name = 'Small Bank  
Corporation')  
    )  
);
```

```
-- 8. Find all companies located in every city in  
which Small Bank Corporation is located.
```

```
SELECT c.company_name  
FROM company c  
WHERE NOT EXISTS (  

```

```
SELECT *
FROM company s
WHERE s.company_name = 'Small Bank Corporation'
AND s.city NOT IN (SELECT city FROM company WHERE
company_name = c.company_name)
);
```

-- 9. Find all employees who earn more than the average salary of all employees of their company.

```
SELECT e.employee_name
FROM employee e
JOIN works w ON e.employee_name = w.employee_name
WHERE w.salary > (
    SELECT AVG(w2.salary)
    FROM works w2
    WHERE w2.company_name = w.company_name
);
```

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

```
SELECT company_name
FROM works
GROUP BY company_name
ORDER BY COUNT(employee_name) DESC
LIMIT 1;
```

-- 11. Find the company that has the smallest payroll.

```
SELECT company_name
FROM works
GROUP BY company_name
ORDER BY SUM(salary) ASC
LIMIT 1;
```

```
-- 12. Find those companies whose employees earn a
higher salary, on average, than the average salary at
First Bank Corporation.
SELECT c.company_name
FROM company c
JOIN works w ON c.company_name = w.company_name
GROUP BY c.company_name
HAVING AVG(w.salary) > (SELECT AVG(salary) FROM works
WHERE company_name = 'First Bank Corporation');
```

Assignmnet 4

```
create table borrow(rollin int, name varchar(20),dt_issue date,
nameofbook varchar(50),status char(1));
insert into borrow values(1,'aa','20-Aug-2024','dbms','I');
insert into borrow values(2,'cc','20-Jul-2024','db','I');
insert into borrow values(3,'bb','30-Jul-2024','db1','I');
create table fine(roll int, rdate date, fineamt int);
```

Declare

```
roll int;
famount int:=0;
idate date;
days int;
sta char(1);
```

begin

```
roll:=:r;
select dt_issue,status into idate,sta from borrow where rollin=roll;
if sta='I' then
days := SYSDATE()- idate;
dbms_output.put_line('No of days' || days);
```

```

if(days>= 15 and days <=30)then
    famount:=(days-15)*5;
elsif (days >30)then
    famount:=((days-30)*50);
end if;
dbms_output.put_line('Fine amount' || famount);
insert into fine values(roll,SYSDATE(),famount);
update borrow set STATUS='R' where rollin=roll;

else
    dbms_output.put_line('Status is R');
end if;

EXCEPTION WHEN NO_DATA_FOUND THEN
    dbms_output.put_line(' Roll no does not exist');

end;

```

Assignment 5

```

create table employee (id int, name varchar(20), salary number,
hire_date date);

Insert into employee1 values(1,'aaa',20000,'2-Jan-2000');
Insert into employee1 values(2,'bbb',30000,'2-Jan-2020');
Insert into employee1 values(3,'ccc',20000,'2-Jul-2015');

Declare

    cursor ecur is select * from employee1;

    erec ecur%rowtype;

```

Begin

open ecur;

loop

fetch ecur into erec;

exit when ecur%NOTFOUND;

dbms_output.put_line(erec.id || ' ' || erec.name || ' ' || erec.salary);

end loop;

close ecur;

end;

Declare

cursor ecur is select * from employee1;

erec ecur%rowtype;

yr number;

incentive number:=0;

Begin

open ecur;

loop

fetch ecur into erec;

exit when ecur%NOTFOUND;

yr:=(sysdate()-erec.hire_date)/365;

If yr>5 then

incentive:=erec.salary*0.05;

```

Else
    If yr>10 then
        incentive:=erec.salary*0.1;
    End if;
End if;

dbms_output.put_line(erec.id||' '|| erec.name||' '|| erec.salary);
dbms_output.put_line('incentive '|| incentive);

end loop;

close ecur;

end;

```

Assignment 6

```

CREATE TABLE Books (
    AccNo NUMBER PRIMARY KEY,
    Title VARCHAR(50),
    Author VARCHAR(50),
    Publisher VARCHAR(50),
    Count NUMBER
);

```

```

CREATE TABLE Library_Audit (
    AccNo NUMBER,
    Title VARCHAR(50),
    Author VARCHAR(50),
    Publisher VARCHAR(50),
    Count NUMBER,

```



```

        Action_Date DATE,
        Status VARCHAR2(50)
    );

CREATE OR REPLACE TRIGGER before_delete_books
BEFORE DELETE ON Books
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action_Date, Status)
    VALUES (:OLD.AccNo, :OLD.Title, :OLD.Author, :OLD.Publisher, :OLD.Count, SYSDATE, 'deleted');
END;

CREATE OR REPLACE TRIGGER after_update_books
AFTER UPDATE ON Books
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count, Action_Date, Status)
    VALUES (:NEW.AccNo, :NEW.Title, :NEW.Author, :NEW.Publisher, :NEW.Count, SYSDATE, 'updated');
END;

```

Assignment 8

```

sudo systemctl start mongod
mongo

```

```

use Ass8

db.createCollection("books");

db.books.insertMany([
  {
    TITLE: "mongodb",
    DESCRIPTION: "A NoSQL database",
    BY: "john",
    URL: "http://example.com/mongodb",
    TAGS: ["database", "nosql", "mongodb"],
    LIKES: 150
  },

```

```

{
  TITLE: "nodejs",
  DESCRIPTION: "JavaScript runtime",
  BY: "jane",
  URL: "http://example.com/nodejs",
  TAGS: ["javascript", "node", "backend"],
  LIKES: 200
},
{
  TITLE: "react",
  DESCRIPTION: "A JavaScript library for building user interfaces",
  BY: "john",
  URL: "http://example.com/react",
  TAGS: ["javascript", "frontend", "react"],
  LIKES: 250
},
{
  TITLE: "nosql overview",
  DESCRIPTION: "Overview of NoSQL databases",
  BY: "alice",
  URL: "http://example.com/nosql",
  TAGS: ["database", "nosql"],
  LIKES: 50
},
{
  TITLE: "expressjs",
  DESCRIPTION: "Web framework for Node.js",
  BY: "bob",
  URL: "http://example.com/expressjs",
  TAGS: ["javascript", "node", "backend"],
  LIKES: 120
}
]);

db.books.insertOne({
  TITLE: "angular",
  DESCRIPTION: "A framework for building web apps",
  BY: "john",
  URL: "http://example.com/angular",
  TAGS: ["javascript", "frontend", "angular"],
  LIKES: 180,
  USERNAME: "johnDoe",
  COMMENTS: [
    { user: "jane", comment: "Great book!" },
    { user: "bob", comment: "Very informative." }
  ]
});

```

```
db.books.find({ TITLE: "mongodb" });
```

```
db.books.find({  
  $or: [{ BY: "john" }, { TITLE: "mongodb" }]  
});
```

```
db.books.find({  
  TITLE: "mongodb",  
  BY: "john"  
});
```

```
db.books.find({  
  LIKES: { $gt: 10 }  
});
```

```
db.books.find({  
  LIKES: { $gt: 100 },  
  $or: [{ TITLE: "mongodb" }, { BY: "john" }]  
});
```

```
db.books.updateOne({ TITLE: "mongodb" }, { $set: { TITLE: "mongodb overview" } });
```

```
db.books.deleteOne({ TITLE: "nosql overview" });
```

```
db.books.find({ BY: "john" }).limit(2);
```

```
db.books.find({ BY: "john" }).skip(1).limit(1);
```

```
db.books.find().sort({ TITLE: 1 });
```

```
db.books.save({  
  TITLE: "vuejs",  
  DESCRIPTION: "A progressive framework for building user interfaces",  
  BY: "sam",  
  URL: "http://example.com/vuejs",  
  TAGS: ["javascript", "frontend", "vue"],  
  LIKES: 220  
});
```

Assignment 9

```
sudo systemctl start mongod
```

mongo

```
use Ass9
```

```
db.createCollection("books");
```

```
db.books.insertMany([
  {
    TITLE: "mongodb",
    DESCRIPTION: "A NoSQL database",
    BY: "john",
    URL: "http://example.com/mongodb",
    TAGS: ["database", "nosql", "mongodb"],
    LIKES: 150
  },
  {
    TITLE: "nodejs",
    DESCRIPTION: "JavaScript runtime",
    BY: "jane",
    URL: "http://example.com/nodejs",
    TAGS: ["javascript", "node", "backend"],
    LIKES: 200
  },
  {
    TITLE: "react",
    DESCRIPTION: "A JavaScript library for building user interfaces",
    BY: "john",
    URL: "http://example.com/react",
    TAGS: ["javascript", "frontend", "react"],
    LIKES: 250
  },
  {
    TITLE: "angular",
    DESCRIPTION: "A framework for building web apps",
    BY: "john",
    URL: "http://example.com/angular",
    TAGS: ["javascript", "frontend", "angular"],
    LIKES: 180
  }
])
```

```
        TITLE: "vuejs",
        DESCRIPTION: "A progressive framework for building user interfaces",
        BY: "john",
        URL: "http://example.com/vuejs",
        TAGS: ["javascript", "frontend", "vue"],
        LIKES: 100
    }
});
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $count: "totalBooksByJohn" }
]);
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $sort: { LIKES: 1 } },
    { $limit: 1 }
]);
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $sort: { LIKES: -1 } },
    { $limit: 1 }
]);
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $group: { _id: "$BY", avgLikes: { $avg: "$LIKES" } } }
    the average likes
]);
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $sort: { _id: 1 } },
    insertion time) { $limit: 1 }]);
```

```
db.books.aggregate([
    { $match: { BY: "john" } },
    { $sort: { _id: -1 } },
    { $limit: 1 }
]);
```

```
db.books.createIndex({ BY: 1 });  
db.books.find({ BY: "john" }).explain("executionStats");
```

Assignment 10

