Section - 01

1. Write the query to CREATE a new table.

```
CREATE TABLE Employees (
employee_id INT PRIMARY KEY,
First_name VARCHAR(100),
Last_name (100),
Email VARCHAR(100),
Birth_date DATE,
Salary INT);
```

2. Write the query to INSERT a new record into a table

```
INSERT INTO Employees
(employee_id,First_name,Last_name, Email,Birth_date,Salary)
VALUES (1, 'Nafis', ‘Iqbal’,’nafisiqbal@gmail.com’,25000);
```

3. Write the query to DELETE records from a table

=>DELETE FROM table_name WHERE condition;

4. Write the query to update records in a table.

```
=>UPDATE table_name SET
column1 = new_value1, column2 = new_value2, ...
WHERE condition;
```

5. Write a query to add a new column to a table.

=> ALTER TABLE table_name ADD new_column_name datatype;

6. Write a query to delete a column from a table.

=> ALTER TABLE table_name DROP COLUMN column_name;

7. Write a query to rename a column of a table.

=> ALTER TABLE table_name RENAME COLUMN old_column_name TO new_column_name;

8. Write the query to select all records from a table to retrieve data.

=> SELECT * FROM table_name;

9. Write the query to select a specific column from a table.

=> SELECT specific_column FROM table_name;

10. Write the query to select distinct values from a column.
=> SELECT DISTINCT column_name FROM table_name;

11. Write the query to filter records using a WHERE clause.
=> SELECT * FROM table_name WHERE condition;

12. Write the query to sort records in descending order.
=> SELECT * FROM table_name ORDER BY column_name DESC;

13. Write the query to sort records in ascending order.
=> SELECT * FROM table_name ORDER BY column_name ASC;

14. Write the query to join two tables based on a common column.
=> SELECT * FROM table1
JOIN table2 ON table1.common_column = table2.common_column;

15. Write the query to count the number of rows in a table.
=> SELECT COUNT(*) FROM table_name;

16. Write the query to group records and calculate aggregate functions.
=> SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name;

17. Write the query to limit the number of rows returned in a result set.
=> SELECT * FROM table_name LIMIT 5;

18. Write the query to find the sum of values in a column.
=> SELECT SUM (column_name) FROM table_name;

19. Write the query to find the average value of a column.
=> SELECT AVG (column_name) FROM table_name;

20. Write the query to get the minimum value from a column.
=> SELECT MIN (column_name) FROM table_name;

21. Write the query to retrieve the maximum value from a column.
=> SELECT MAX (column_name) FROM table_name;

22. Write the query to retrieve rows with values within a specified range.

=> SELECT * FROM table_nameWHERE column_name BETWEEN value1 AND value2;

23. Write the query to retrieve rows with values matching a list of specified values.
=> SELECT columnName FROM tableName
WHERE columnName IN (Value1, Value2, Value3);

24. Write the query to search for patterns in a column using wildcard characters.
=> SELECT columnName FROM tableName
WHERE columnName LIKE 'abc%';

25. Write the query to filter data based on aggregate functions in a GROUP BY query.
=> SELECT column_name, AVG(another_column) FROM table_name
GROUP BY column_name HAVING AVG(another_column) > 10;

26. Write the query to delete all rows from a table, but keep the table structure.
=> DELETE FROM table_name;

27. Write a query to implement simple and complex views.
=>CREATE VIEW view_name AS SELECT column1, column2
FROM table_name WHERE condition;

28. Write a query to implement stored procedures.
=> CREATE PROCEDURE procedure_name AS BEGIN
SELECT * FROM table WHERE column1 = param1 AND column2 = param2;
END
29. Write a query to implement triggers.
=> CREATE TRIGGER trigger_name
AFTER INSERT ON table_name FOR EACH ROW BEGIN
SELECT * FROM table WHERE column1 = param1 AND column2 = param2; END;

Section - 02

1. Consider the following relational database :
   Orders(order_no, purchase_amount, order_date, customer_id, salesman_id)
   Customers(customers_id, customer_name, city, grade, salesman_id)
   Salesman(salesman_id, name, city, commission)

   a) Write a sql statement to make a list with order no, purchase amount, customer name and their cities for those orders which order amount between

500 and 2000.

b) Write a sql statement to display those orders which are not issued on date 2024-01-05 and issued by the salesman whose id is 505.

c) Write a query to find the sums of amount from the orders table grouped by date, eliminating all those dates where the sum was not atleast 1000.

d) Write a query to display all those orders by the customers not located in the same cities where their salesman live.

a) SELECT Orders.order_no, Orders.purchase_amount, Customers.customer_name, Customers.city FROM Orders JOIN Customers ON Orders.customer_id = Customers.customer_id WHERE Orders.purchase_amount BETWEEN 500 AND 2000;

b) SELECT * FROM Orders WHERE order_date != '2024-01-05' AND salesman_id = 505;

c) SELECT order_date, SUM(purchase_amount) AS total_amount FROM Orders GROUP BY order_date HAVING total_amount >= 1000;

d) SELECT Orders.*, Customers.city AS customer_city, Salesman.city AS salesman_city
FROM Orders JOIN Customers ON Orders.customer_id = Customers.customer_id JOIN Salesman ON Customers.salesman_id = Salesman.salesman_id WHERE Customers.city <> Salesman.city;

2. Consider the following database schema :

Employee(emp_id, name, hire_date, job_id, salary, commission, manager_id, department_id) Departments(department_id, department_name, manager_id, location_id)

Locations(location_id, street, post_code, city, state, country)

a) Write a query to display the department id, employee name, job and department name for all employees in the Finanace Department.

b) Write a query to display all the information of the employees who do not work on those departments where some employees works whose id within the range 100 and 200.

c) Write a query to display the employee id, name and job for all employees whose salary is more than any average salary of any department.

d) Write a query to display the department id and the total salary for those departments which contain at least one salaried employee.

a) SELECT e.department_id, e.name AS employee_name, e.job_id, d.department_name FROM Employee e JOIN Departments d ON e.department_id = d.department_id WHERE d.department_name = 'Finance';

b) SELECT * FROM Employee WHERE department_id NOT IN (SELECT department_id
FROM Employee WHERE emp_id BETWEEN 100 AND 200);

c) SELECT emp_id, name, job_id FROM Employee WHERE salary > (SELECT AVG(salary) FROM Employee GROUP BY department_id);

d) SELECT department_id, SUM(salary) AS total_salary FROM Employee GROUP BY department_id HAVING SUM(salary) > 0;

3. Consider the following relational schema :Employee(emp_id, street, city)Works(emp_id, company_name, salary)Company(company_id, company_name, city)

a) Find the name of all employees who live in the same city as the company for whichthey work)

b) Find the name of all employees who do not work for the AB Bank.

c) Find the name of all employees who earn more than every employee of SB Limited.

d) Find the name of all employees who work for FB limited.

a) SELECT e.name FROM Employee e JOIN Works w ON e.emp_id = w.emp_id JOIN company c ON w.company_name = c.company_name AND e.city = c.city;

b) SELECT e.name FROM Employee e LEFT JOIN Works w ON e.emp_id = w.emp_id AND w.company_name = 'AB Bank' WHERE w.emp_id IS NULL;

c) SELECT e.name FROM Employee e JOIN Works w ON e.emp_id = w.emp_id WHERE salary > ALL (SELECT salary FROM Works WHERE company_name = 'SB Limited');

d) SELECT e.name FROM Employee e JOIN Works w ON e.emp_id = w.emp_id WHERE w.company_name = 'FB Limited';

4. Consider the following relational schema :

Student(student_id, name, fathers_name, admission_date, reg_no)
Courses(course_id, course_title, course_type, credit, dept_id)
Major(student_id, department_id, admission_session)
CourseRegistration(student_id, course_id, year, semester, session)
CourseComplete(student_id, course_id, year, semester, session, grade_point, grade_letter)
a) Display the students of "CSE" department of 1st Year and 2nd Semester.
b) List the students of EEE who have completed 5 course.

c) List the students who get grade point below 2.50 in any courses in session 2017-2018
d) List the student id, name and fathers name of students who go A+ in course "CSE 2201".

a) SELECT s.* FROM Student s JOIN Major m ON s.student_id = m.student_id WHERE m.department_id = 'CSE' AND m.admission_session = '1st Year' AND m.semester = '2nd Semester';
b) SELECT s.* FROM Student s JOIN Major m ON s.student_id = m.student_id JOIN CourseComplete cc ON s.student_id = cc.student_id JOIN Courses c ON cc.course_id =c.course_id WHERE m.department_id = 'EEE' GROUP BY s.student_id HAVING COUNT(cc.course_id) = 5;

c) SELECT cc.student_id, s.name, s.fathers_name FROM CourseComplete cc JOIN Student s ON cc.student_id = s.student_id WHERE cc.grade_point < 2.50 AND cc.session = 2017-2018';

d) SELECT cc.student_id, s.name, s.fathers_name FROM CourseComplete cc JOIN Student s ON cc.student_id = s.student_id WHERE cc.course_id = 'CSE 2201' AND cc.grade_letter = 'A+';

5. Consider an insurance database that maintain the following information :
Person(DriverNo, Name, Address)
Car(LicenseNo, Model, Year)
Accident(ReportNo, Date, Location)
Owns(DriverNo, LicenseNo)
Participated(DriverNo, LicenseNo, ReportNo, DamageAmount)
Write down the SQL command to find the following queries :
i. Find the total number of people whose cars were involved in accident.

ii. Find the number of accident in which the belongs to “Mr. X”.
iii. Delete the “CAR1” car belongs to “Mr. X”.
iv. Add new accident report for the “CAR2” car belongs to “Mr. Y”.

i. SELECT COUNT(DISTINCT DriverNo) FROM Participated;

ii. SELECT COUNT(*) FROM Accident WHERE DriverNo IN (SELECT DriverNo FROM Person WHERE Name = 'Mr. X');
iii. DELETE FROM Car WHERE LicenseNo = 'CAR1' AND DriverNo IN (SELECT DriverNo FROM Person WHERE Name = 'Mr. X');
iv. INSERT INTO Accident (DriverNo, LicenseNo, ReportNo, DamageAmount)
VALUES ('DriverNo', 'CAR2', 'ReportNo', 'DamageAmount')
WHERE DriverNo IN (SELECT DriverNo FROM Person WHERE Name = 'Mr. Y');

6. Consider the following relational schema :
Branch(branch_id, branch_name, branch_city, assets)
Customer(customer_id, customer_name, customer_city, customer_street)
Account(account_number, branch_id, balance)
Loan(loan_number, branch_id, amount)
Depositor(customer_id, account_number)
Borrower(customer_id, loan_number)
a) Find the loan number for each loan of an amount greater than $1200.
b) Find the names of all customers who have a loan, an account, or both from the bank.
c) Find the name of all customers who have a loan at the perryridge branch.
d) Find the name of all customers who an account at every branch located in Brroklyn.

a) SELECT loan_number FROM Loan WHERE amount > 1200;

b) SELECT DISTINCT c.name FROM Customer c LEFT JOIN Borrower b ON c.customer_id = b.customer_id LEFT JOIN Account a ON c.customer_id = a.customer_id WHERE b.customer_id IS NOT NULL OR a.customer_id IS NOT NULL;

c) SELECT DISTINCT c.name FROM Customer c JOIN Borrower b ON c.customer_id = b.customer_id JOIN Loan l ON b.loan_number = l.loan_number WHERE l.branch_id = (SELECT branch_id FROM Locations WHERE branch_city = 'Perryridge');

d) SELECT DISTINCT c.name FROM Customer c JOIN Account a ON c.customer_id = a.customer_id JOIN Locations l ON a.branch_id = l.location_id WHERE l.city = 'Brroklyn';