1. **What does the following query return?**

   SELECT e.employee_id, e.employee_name, d.department_name
   FROM employees e
   INNER JOIN departments d ON e.department_id =
   d.department_id;

   A)

   SELECT e.employee_id, e.employee_name,
   d.department_name
   FROM employees e, departments d
   WHERE e.department_id = d.department_id;

   B)

   SELECT e.employee_id, e.employee_name,
   d.department_name
   FROM employees e
   LEFT JOIN departments d ON e.department_id =
   d.department_id;

   C)

   SELECT e.employee_id, e.employee_name,
   d.department_name
   FROM employees e
   RIGHT JOIN departments d ON e.department_id =
   d.department_id;

   D)

   SELECT e.employee_id, e.employee_name,
   d.department_name
   FROM employees e
   FULL OUTER JOIN departments d ON e.department_id =
   d.department_id;

2. **Which join type returns all records from the left table and matching records from the right table?**

```
SELECT *
FROM orders o
LEFT JOIN customers c ON o.customer_id = c.customer_id;
```

A)

```
SELECT *
FROM orders o, customers c
WHERE o.customer_id = c.customer_id(+);
```

B)

```
SELECT *
FROM orders o
RIGHT JOIN customers c ON o.customer_id = c.customer_id;
```

C)

```
SELECT *
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id;
```

D)

```
SELECT *
FROM orders o
FULL JOIN customers c ON o.customer_id = c.customer_id;
```

3. **What does the following query accomplish?**

```
SELECT *
FROM products p
RIGHT JOIN orders o ON p.product_id = o.product_id;
```

A)

```
SELECT *
FROM products p
LEFT JOIN orders o ON p.product_id = o.product_id;
```

B)

```
SELECT *
FROM products p
INNER JOIN orders o ON p.product_id = o.product_id;
```

C)

```
SELECT *
FROM products p
FULL OUTER JOIN orders o ON p.product_id = o.product_id;
```

D)

```
SELECT *
FROM products p
CROSS JOIN orders o ON p.product_id = o.product_id;
```

Constraints

4. **Which constraint is used to ensure that a column contains only unique values?**

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50) UNIQUE
);
```

A)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
```

```
    username VARCHAR(50) CONSTRAINT uniq_username
UNIQUE
);
```

B)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    UNIQUE (username)
);
```

C)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    CHECK UNIQUE (username)
);
```

D)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    CONSTRAINT UNIQUE (username)
);
```

Subqueries

5. **What is the purpose of the following subquery?**

```
SELECT order_id, customer_id
FROM orders
WHERE customer_id IN (
   SELECT customer_id
   FROM customers
   WHERE country = 'USA'
);
```

A)

```
SELECT order_id, customer_id
FROM orders
WHERE customer_id = (SELECT customer_id FROM customers
WHERE country = 'USA');
```

B)

```
SELECT order_id, customer_id
FROM orders
WHERE customer_id NOT IN (SELECT customer_id FROM
customers WHERE country = 'USA');
```

C)

```
SELECT order_id, customer_id
FROM orders
WHERE EXISTS (SELECT * FROM customers WHERE
customers.customer_id = orders.customer_id AND country =
'USA');
```

D)

```
SELECT order_id, customer_id
FROM orders
```

WHERE customer_id = ANY (SELECT customer_id FROM customers WHERE country = 'USA');

6. **Which type of subquery returns a single value and can be used within the SELECT statement of an outer query?**

```
SELECT order_id,
    (SELECT customer_name FROM customers WHERE customers.customer_id = orders.customer_id) AS customer_name
FROM orders;
```

A)

```
SELECT order_id, (SELECT MAX(customer_name) FROM customers) AS customer_name
FROM orders;
```

B)

```
SELECT order_id, (SELECT COUNT(*) FROM customers WHERE customers.customer_id = orders.customer_id) AS customer_count
FROM orders;
```

C)

```
SELECT order_id, (SELECT customer_name FROM customers WHERE customers.customer_id = orders.customer_id) AS customer_name
FROM orders;
```

D)

```
SELECT order_id, (SELECT customer_name FROM customers) AS customer_name
FROM orders;
```

DDL (Data Definition Language)

7. **What does the following command do?**

```
ALTER TABLE employees
ADD COLUMN hire_date DATE;
```

A)

```
ALTER TABLE employees
ADD hire_date DATE;
```

B)

```
ALTER employees
ADD COLUMN hire_date DATE;
```

C)

```
ALTER TABLE employees
MODIFY hire_date DATE;
```

D)

```
ALTER TABLE employees
CHANGE COLUMN hire_date DATE;
```

8. **Which command is used to remove a table from the database?**

```
DROP TABLE customers;
```

A)

```
DELETE TABLE customers;
```

B)

```
REMOVE TABLE customers;
```

C)

DROP customers;

D)

DROP TABLE customers;

DML (Data Manipulation Language)

9. **What is the purpose of the following command?**

```
UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 2;
```

A)

```
UPDATE employees
SET salary = salary + 0.1
WHERE department_id = 2;
```

B)

```
UPDATE employees
SET salary = salary * 0.1
WHERE department_id = 2;
```

C)

```
UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 1;
```

D)

```
UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 2;
```

10. **What does the following command accomplish?**

```
INSERT INTO employees (employee_id, employee_name,
department_id)
VALUES (101, 'John Doe', 3);
```

A)

Inserts a new employee with ID 101 and name 'John Doe' into department 3.

B)

Modifies the employee with ID 101 to have name 'John Doe' and department 3.

C)

Deletes the employee with ID 101.

D)

Updates the employee with ID 101.

Group By and Having

11. **What does the following query do?**

```
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id
HAVING AVG(salary) > 50000;
```

A)

Lists all employees with a salary above 50,000 grouped by department.

B)

Lists departments with an average salary above 50,000.

C)

Lists departments and their average salary, filtering out those with an average salary below 50,000.

D)

Lists employees with a salary above 50,000, grouped by department.

12. **What is the result of the following query?**

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 10;
```

- A)

Lists departments with more than 10 employees.

B)

Lists departments with exactly 10 employees.

C)

Lists departments with fewer than 10 employees.

D)

Lists departments with exactly one employee.

Constraints

13. **Which constraint is used to ensure that a column cannot have NULL values?**

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100) NOT NULL
);
```

A)

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    CHECK (employee_name IS NOT NULL)
);
```

B)

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    NOT NULL (employee_name)
);
```

C)

```
CREATE TABLE employees (
```

```
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    CONSTRAINT nn_employee_name NOT NULL
(employee_name)
);
```

D)

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100) NOT NULL
);
```

14. **What does the following statement do?**

```
ALTER TABLE products
ADD CONSTRAINT chk_price CHECK (price > 0);
```

A)

Adds a check constraint on the 'products' table to ensure that the 'price' column has a value greater than 0.

B)

Modifies the 'products' table structure by adding a 'price' column and applying a check constraint to it.

C)

Drops the check constraint named 'chk_price' from the 'products' table.

D)

Adds a foreign key constraint to the 'products' table referencing the 'price' column.

Normalization

15. **What is the primary goal of database normalization?**

    A) To minimize data redundancy and dependency

    B) To optimize query performance

    C) To ensure high availability of the database

    D) To enforce referential integrity

16. **Which normalization form ensures that there are no repeating groups of data within a table?**

    A) First Normal Form (1NF)

    B) Second Normal Form (2NF)

    C) Third Normal Form (3NF)

    D) Boyce-Codd Normal Form (BCNF)

Set Operations

17. **What does the following query do?**

```
SELECT customer_id, customer_name
FROM customers
UNION
SELECT supplier_id, supplier_name
FROM suppliers;
```

    A)

Combines and returns all distinct customer IDs and supplier IDs along with their names from the 'customers' and 'suppliers' tables.

B)

Returns only the customer IDs and names from the 'customers' table.

C) Returns only the supplier IDs and names from the 'suppliers' table.

D) Returns customer IDs and supplier IDs along with their names, including duplicates, from the 'customers' and 'suppliers' tables

Subqueries

18. **What does the following subquery accomplish?**

```
SELECT employee_id, employee_name
FROM employees
WHERE department_id = (
    SELECT department_id
    FROM departments
    WHERE department_name = 'Sales'
);
```

A) Lists all employees in the 'Sales' department.

B) Lists all departments where employees are from the 'Sales' department.

C) Lists all employees in departments with the same department ID as the 'Sales' department.

D) Lists all employees and their department IDs from the 'Sales' department.

19. **Which type of subquery is executed once for each row processed by the outer query?**

```
SELECT product_name, unit_price
FROM products
WHERE unit_price > (
   SELECT AVG(unit_price)
   FROM products
);
```

A) Correlated Subquery

B) Nested Subquery

C) Scalar Subquery

D) Derived Subquery

Joins

20. **What is the result of the following query?**

```
SELECT c.customer_id, c.customer_name, COUNT(o.order_id) AS
order_count
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.customer_name
```

ORDER BY order_count DESC;

A) Lists customers sorted by their number of orders in descending order.

B) Lists orders sorted by customer ID and customer name.

C) Lists customers with more than one order.

D) Lists customers with fewer than one order.