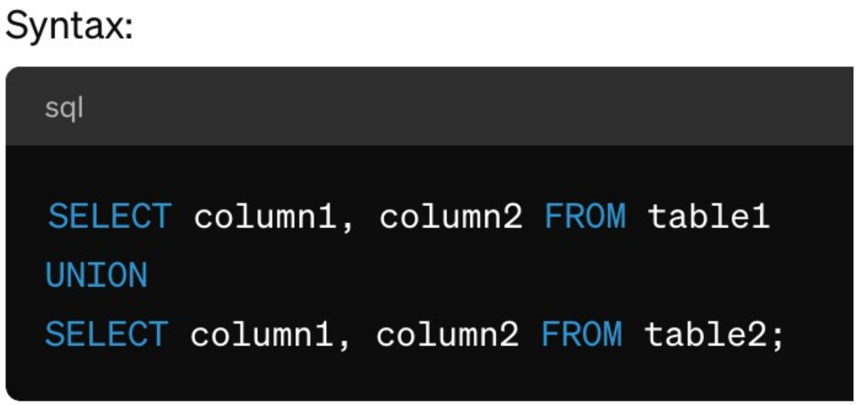
NATIONAL UNIVERSITY

ADVANCE SQL CONCEPTS.

SET OPERATIONS

UNION Operator:

- Combines the result sets of two or more SELECT statements into a single result set, eliminating duplicates.



Syntax:

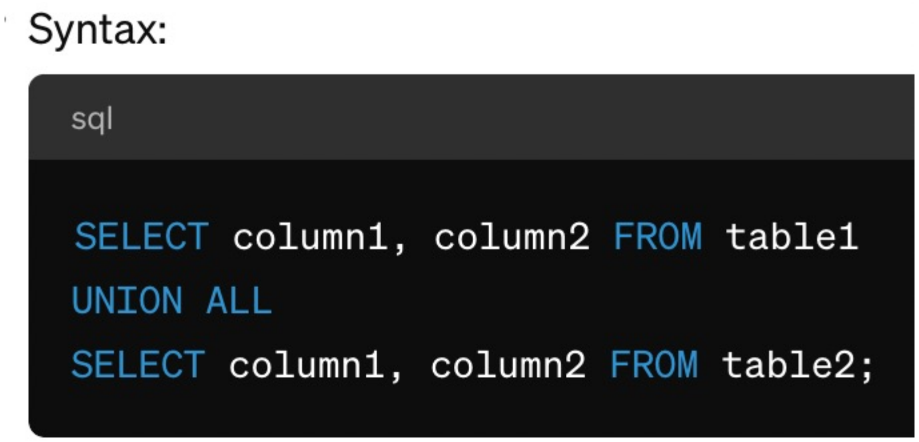
sql

```sql
SELECT column1, column2 FROM table1
UNION
SELECT column1, column2 FROM table2;
```

SET OPERATIONS

UNION ALL Operator:

- Similar to UNION but includes all rows, including duplicates, from the result sets of the SELECT statements.



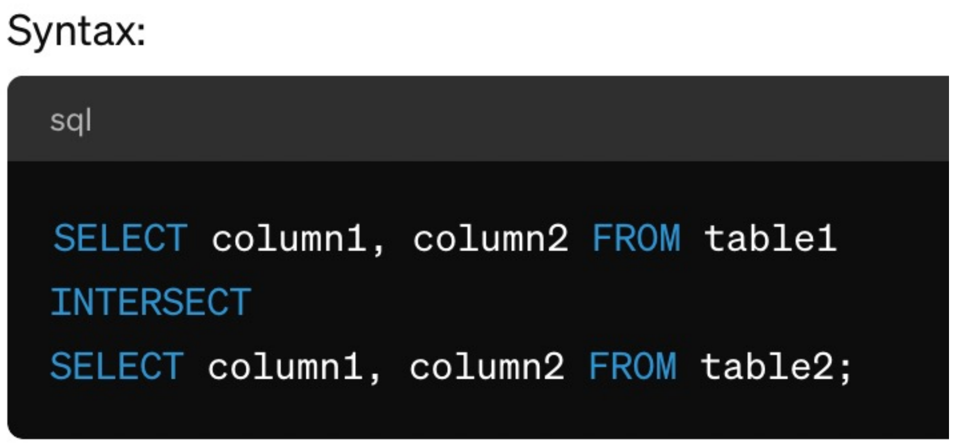
Syntax:

```sql
SELECT column1, column2 FROM table1
UNION ALL
SELECT column1, column2 FROM table2;
```

SET OPERATIONS

INTERSECT Operator:

- Retrieves rows that appear in both result sets of the SELECT statements.



Syntax:

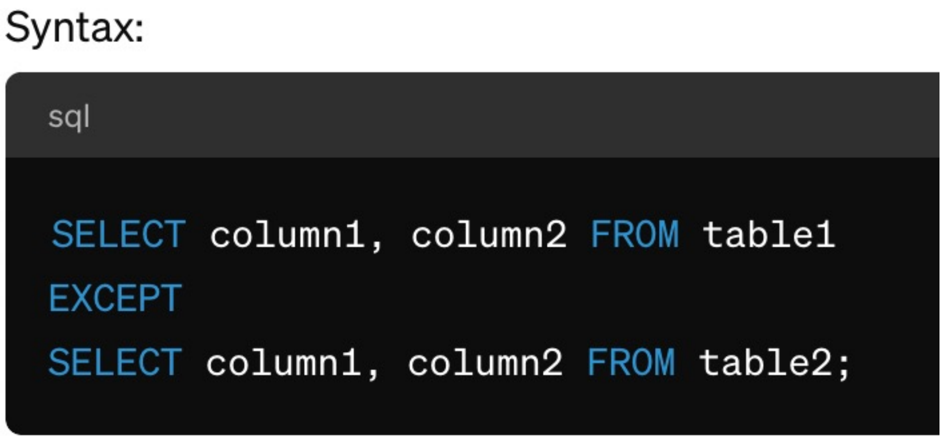
sql

```sql
SELECT column1, column2 FROM table1
INTERSECT
SELECT column1, column2 FROM table2;
```

SET OPERATIONS

EXCEPT Operator:

- Retrieves rows from the first SELECT statement that are not present in the result set of the second SELECT statement.



Syntax:

sql

```sql
SELECT column1, column2 FROM table1
EXCEPT
SELECT column1, column2 FROM table2;
```

EXAMPLES:

- Suppose we have two tables, Employees and Managers, with similar structures (EmployeeID, FirstName, LastName, DepartmentID, etc.).
- UNION

```
SELECT FirstName, LastName FROM Employees
UNION
SELECT FirstName, LastName FROM Managers;
```

- UNION ALL

```
SELECT FirstName, LastName FROM Employees
UNION ALL
SELECT FirstName, LastName FROM Managers;
```

EXAMPLES:

- INTERSECT

```
SELECT FirstName, LastName FROM Employees
INTERSECT
SELECT FirstName, LastName FROM Managers;
```

- EXCEPT

```
SELECT FirstName, LastName FROM Employees
EXCEPT
SELECT FirstName, LastName FROM Managers;
```

SET OPERATIONS CONSIDERATIONS:

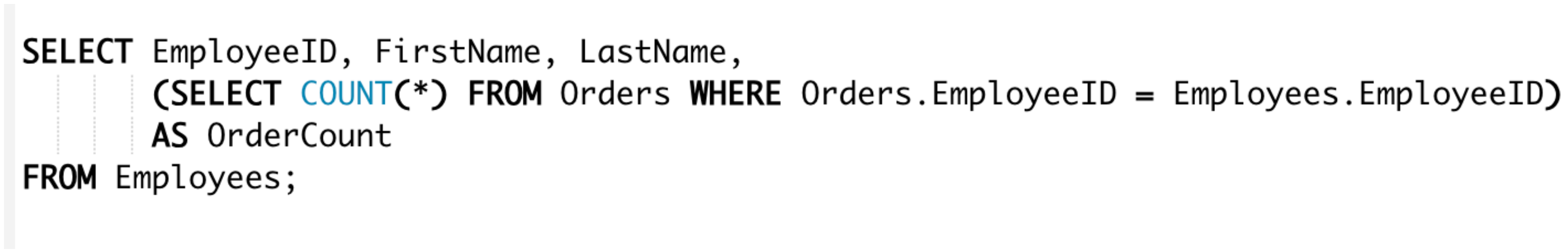
- Column Types: The columns selected in each SELECT statement within a set operation (UNION, INTERSECT, EXCEPT) must have compatible data types.
- Ordering: The order of rows in the result set may vary depending on the operation used. Use ORDER BY if a specific order is required.
- Performance: Set operations can impact performance, especially when dealing with large result sets. Ensure proper indexing and query optimization.

SUBQUERIES:

- Subqueries are queries embedded within another query.
- They can be used in SELECT, INSERT, UPDATE, DELETE statements, or even within other subqueries.
- Common types include correlated and non-correlated subqueries.

CORRELATED SUBQUERIES:

- Correlated subqueries depend on the outer query for their execution and can be thought of as looping through each row of the outer query.



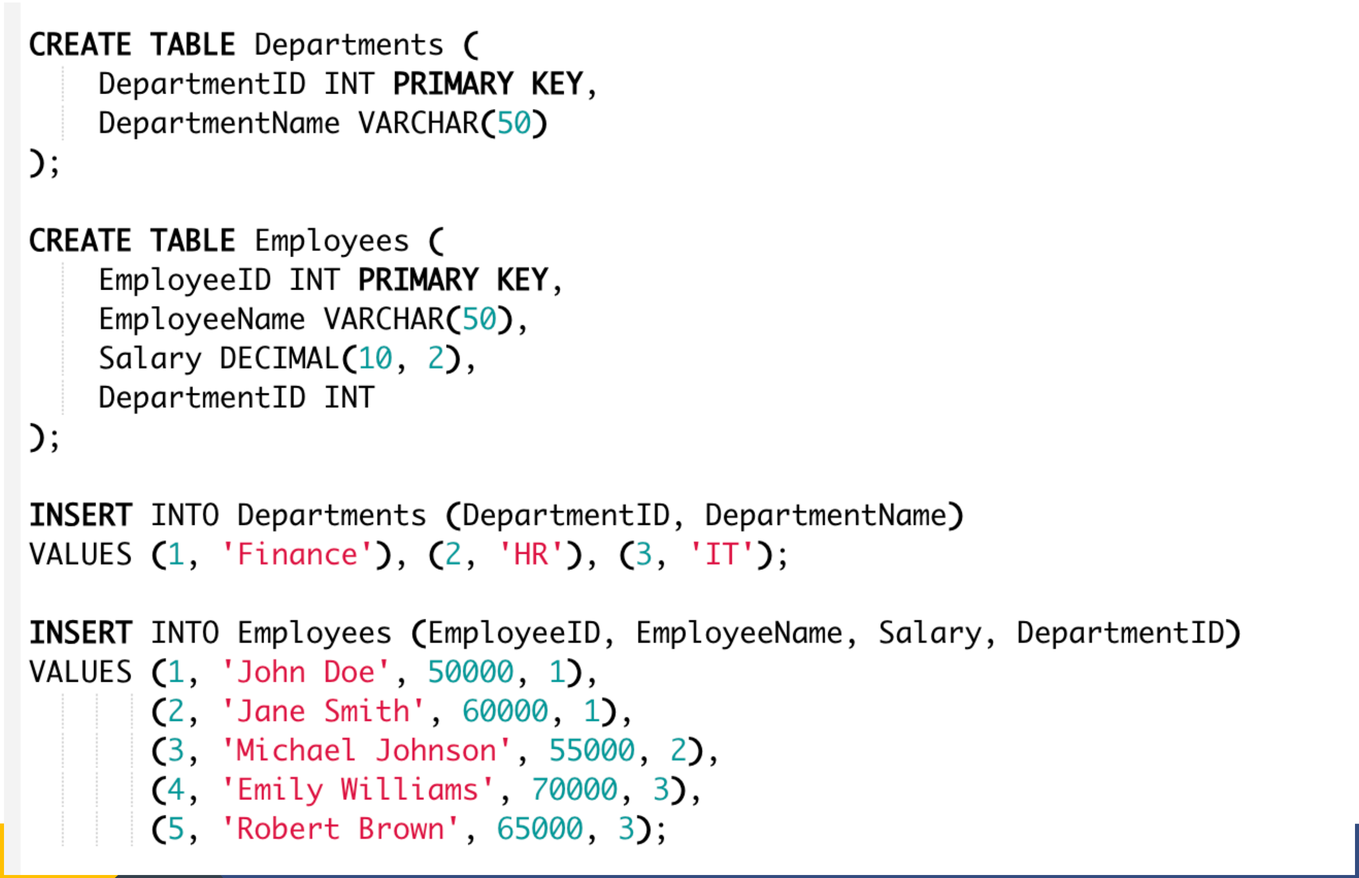
```sql
SELECT EmployeeID, FirstName, LastName,
       (SELECT COUNT(*) FROM Orders WHERE Orders.EmployeeID = Employees.EmployeeID)
       AS OrderCount
FROM Employees;
```

NON-CORRELATED SUBQUERIES:

- Non-correlated subqueries are independent of the outer query and can be executed on their own.

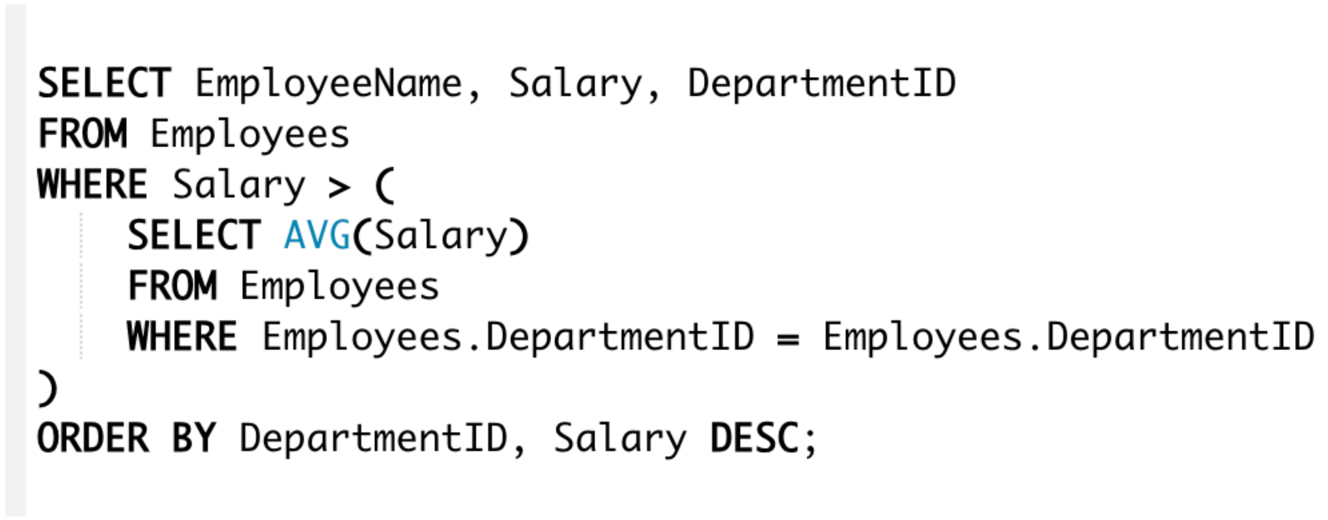
```
SELECT * FROM Employees WHERE DepartmentID IN
(SELECT DepartmentID FROM Departments WHERE DepartmentName = 'Sales');
```

LESSON - SUBQUERIES



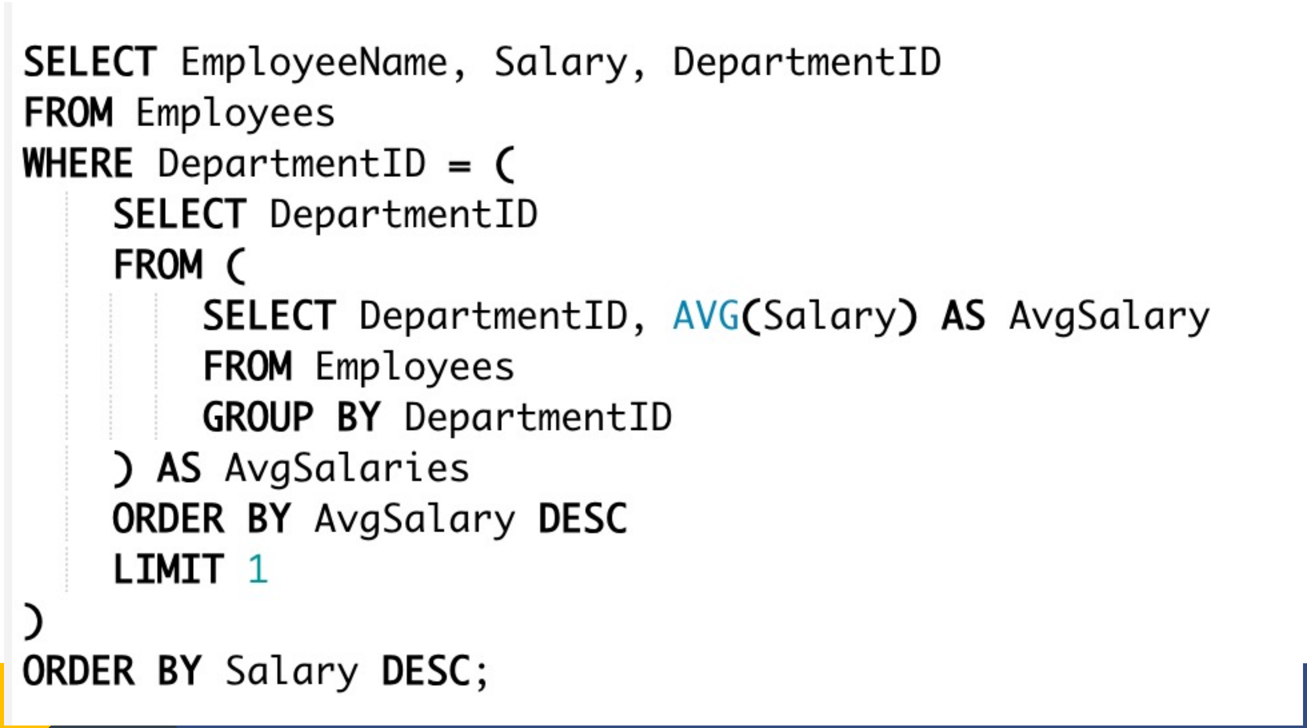
```sql
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50)
);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    EmployeeName VARCHAR(50),
    Salary DECIMAL(10, 2),
    DepartmentID INT
);

INSERT INTO Departments (DepartmentID, DepartmentName)
VALUES (1, 'Finance'), (2, 'HR'), (3, 'IT');

INSERT INTO Employees (EmployeeID, EmployeeName, Salary, DepartmentID)
VALUES (1, 'John Doe', 50000, 1),
       (2, 'Jane Smith', 60000, 1),
       (3, 'Michael Johnson', 55000, 2),
       (4, 'Emily Williams', 70000, 3),
       (5, 'Robert Brown', 65000, 3);
```

WRITE A QUERY USING SUBQUERIES TO FETCH INFORMATION ABOUT EMPLOYEES WHOSE SALARIES ARE ABOVE THE AVERAGE SALARY IN THEIR RESPECTIVE DEPARTMENTS:



```sql
SELECT EmployeeName, Salary, DepartmentID
FROM Employees
WHERE Salary > (
    SELECT AVG(Salary)
    FROM Employees
    WHERE Employees.DepartmentID = Employees.DepartmentID
)
ORDER BY DepartmentID, Salary DESC;
```

WRITE A QUERY TO FIND THE DEPARTMENT WITH THE HIGHEST AVERAGE SALARY AND THEN LIST ALL EMPLOYEES FROM THAT DEPARTMENT.



```sql
SELECT EmployeeName, Salary, DepartmentID
FROM Employees
WHERE DepartmentID = (
    SELECT DepartmentID
    FROM (
        SELECT DepartmentID, AVG(Salary) AS AvgSalary
        FROM Employees
        GROUP BY DepartmentID
    ) AS AvgSalaries
    ORDER BY AvgSalary DESC
    LIMIT 1
)
ORDER BY Salary DESC;
```

EXERCISE:

Question 1: What is a subquery in SQL?

A query that performs mathematical operations
B. A query that retrieves data from multiple tables
C. A query nested inside another query
D. A query that modifies database schema

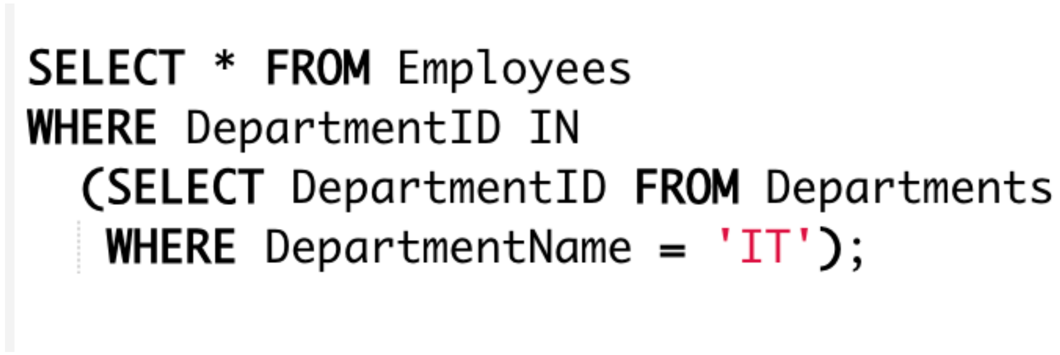
Question 2: Which keyword is used to create a subquery in SQL?

A. SELECT
B. FROM
C. WHERE
D. SUBQUERY

Question 3: What is the purpose of a subquery in SQL?

A. To update database records
B. To join multiple tables
C. To retrieve data based on conditions
D. To create new tables

Question 4: What does the following SQL query do?



```
SELECT * FROM Employees
WHERE DepartmentID IN
  (SELECT DepartmentID FROM Departments
   WHERE DepartmentName = 'IT');
```

A. Retrieves all employees from the IT department
B. Retrieves all employees from the Finance department
C. Retrieves all departments where employees work
D. Retrieves all employees from the HR department

Question 5: Which SQL clause is commonly used with subqueries to filter results?

A. ORDER BY
B. GROUP BY
C. WHERE
D. HAVING