SQL Clauses Lab

Objective

This lab aims to teach you the basic SQL clauses by running queries on a company database consisting of two tables: `employee` and `salary`. By the end of this lab, you will understand how to use SELECT, WHERE, ORDER BY, GROUP BY, HAVING, BETWEEN, LIKE, IN, LIMIT, and OFFSET clauses.

Prerequisites

Ensure you have access to a SQL environment where you can create and manipulate databases.

Database Schema

`employee` Table
- `id` (INT, Auto Increment, Primary Key)
- `first_name` (VARCHAR)
- `last_name` (VARCHAR)
- `email` (VARCHAR)
- `hire_date` (DATE)
- `KRA_PIN` (VARCHAR)

`salary` Table
- `first_name` (VARCHAR)
- `last_name` (VARCHAR)
- `email` (VARCHAR)
- `hire_date` (DATE)
- `salary` (DECIMAL)
- `KRA_PIN` (VARCHAR)

Setup: Create and Populate the Tables with the data below:

```
-- Create employee table
CREATE TABLE employee (
        id INT AUTO_INCREMENT PRIMARY KEY,
        first_name VARCHAR(50),
        last_name VARCHAR(50),
        email VARCHAR(100),
        hire_date DATE,
        KRA_PIN VARCHAR(15)
);
```

```
-- Create salary table
CREATE TABLE salary (
        first_name VARCHAR(50),
        last_name VARCHAR(50),
        email VARCHAR(100),
        hire_date DATE,
        salary DECIMAL(10, 2),
        KRA_PIN VARCHAR(15)
);
```

Insert sample data into employee table:

```
INSERT INTO employee (first_name, last_name, email, hire_date, KRA_PIN) VALUES
('John', 'Doe', 'john.doe@example.com', '2020-01-15', 'A1234567B'),
('Jane', 'Smith', 'jane.smith@example.com', '2019-03-22', 'B2345678C'),
('Emily', 'Jones', 'emily.jones@example.com', '2021-07-01', 'C3456789D'),
('Michael', 'Brown', 'michael.brown@example.com', '2018-11-11', 'D4567890E'),
('Sarah', 'Davis', 'sarah.davis@example.com', '2022-05-20', 'E5678901F'),
('David', 'Wilson', 'david.wilson@example.com', '2017-04-04', 'F6789012G'),
('Laura', 'Martinez', 'laura.martinez@example.com', '2016-02-18', 'G7890123H'),
('Daniel', 'Taylor', 'daniel.taylor@example.com', '2023-01-01', 'H8901234I'),
('Olivia', 'Anderson', 'olivia.anderson@example.com', '2015-09-30', 'I9012345J'),
('Matthew', 'Thomas', 'matthew.thomas@example.com', '2020-06-25', 'J0123456K'),
('Sophia', 'Jackson', 'sophia.jackson@example.com', '2019-08-08', 'K1234567L'),
('James', 'White', 'james.white@example.com', '2021-12-12', 'L2345678M'),
('Emma', 'Harris', 'emma.harris@example.com', '2018-03-03', 'M3456789N'),
('Liam', 'Clark', 'liam.clark@example.com', '2022-07-07', 'N4567890O'),
('Isabella', 'Lewis', 'isabella.lewis@example.com', '2017-01-21', 'O5678901P'),
('Ethan', 'Robinson', 'ethan.robinson@example.com', '2016-11-15', 'P6789012Q'),
('Ava', 'Walker', 'ava.walker@example.com', '2023-05-05', 'Q7890123R'),
('Noah', 'Young', 'noah.young@example.com', '2015-10-10', 'R8901234S'),
('Mia', 'King', 'mia.king@example.com', '2020-12-12', 'S9012345T'),
('Alexander', 'Scott', 'alexander.scott@example.com', '2019-02-02', 'T0123456U');
```

Insert sample data into salary table

```
INSERT INTO salary (first_name, last_name, email, hire_date, salary, KRA_PIN) VALUES
('John', 'Doe', 'john.doe@example.com', '2020-01-15', 60000, 'A1234567B'),
('Jane', 'Smith', 'jane.smith@example.com', '2019-03-22', 75000, 'B2345678C'),
('Emily', 'Jones', 'emily.jones@example.com', '2021-07-01', 50000, 'C3456789D'),
('Michael', 'Brown', 'michael.brown@example.com', '2018-11-11', 85000, 'D4567890E'),
('Sarah', 'Davis', 'sarah.davis@example.com', '2022-05-20', 62000, 'E5678901F'),
('David', 'Wilson', 'david.wilson@example.com', '2017-04-04', 72000, 'F6789012G'),
('Laura', 'Martinez', 'laura.martinez@example.com', '2016-02-18', 81000, 'G7890123H'),
('Daniel', 'Taylor', 'daniel.taylor@example.com', '2023-01-01', 67000, 'H8901234I'),
('Olivia', 'Anderson', 'olivia.anderson@example.com', '2015-09-30', 76000, 'I9012345J'),
('Matthew', 'Thomas', 'matthew.thomas@example.com', '2020-06-25', 55000, 'J0123456K'),
('Sophia', 'Jackson', 'sophia.jackson@example.com', '2019-08-08', 70000, 'K1234567L'),
('James', 'White', 'james.white@example.com', '2021-12-12', 58000, 'L2345678M'),
('Emma', 'Harris', 'emma.harris@example.com', '2018-03-03', 53000, 'M3456789N'),
('Liam', 'Clark', 'liam.clark@example.com', '2022-07-07', 64000, 'N4567890O'),
('Isabella', 'Lewis', 'isabella.lewis@example.com', '2017-01-21', 69000, 'O5678901P'),
('Ethan', 'Robinson', 'ethan.robinson@example.com', '2016-11-15', 74000, 'P6789012Q'),
('Ava', 'Walker', 'ava.walker@example.com', '2023-05-05', 61000, 'Q7890123R'),
('Noah', 'Young', 'noah.young@example.com', '2015-10-10', 87000, 'R8901234S'),
('Mia', 'King', 'mia.king@example.com', '2020-12-12', 68000, 'S9012345T'),
('Alexander', 'Scott', 'alexander.scott@example.com', '2019-02-02', 59000, 'T0123456U');
```

1. WHERE Clause

The WHERE clause is used to filter records.

-- Select employees hired after January 1, 2020

-- Select employees with a salary greater than 60000

2. BETWEEN Operator

The BETWEEN operator selects values within a given range.

-- Select employees hired between January 1, 2019, and December 31, 2020

-- Select employees with a salary between 60000 and 80000

<u>3. LIKE Operator</u>
The LIKE operator is used to search for a specified pattern in a column.

-- Select employees whose first name starts with 'J'

-- Select employees whose email contains 'example'
SELECT * FROM salary WHERE email LIKE '%example%';

<u>4. IN Operator</u>
The IN operator allows you to specify multiple values in a WHERE clause.

-- Select employees whose last name is either 'Smith' or 'Jones'

-- Select employees with salaries of either 60000, 70000, or 80000

<u>5. ORDER BY Clause</u>
The ORDER BY clause is used to sort the result-set.

-- Select all employees ordered by hire date ascending

-- Select all salaries ordered by salary in descending order

<u>6. GROUP BY Clause</u>
The GROUP BY clause groups rows that have the same values into summary rows.

-- Group employees by hire date

-- Group salaries by hire date

-- Group salaries by first name and hire date

7. HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

-- Group by hire date and filter groups with more than 1 employee

-- Group by hire date and filter groups with more than 1 salary record

8. LIMIT and OFFSET Clauses

The LIMIT clause is used to specify the number of records to return, while OFFSET specifies the starting point.

-- Select the first 5 employees

-- Select 5 employees starting from the 6th record

-- Select the first 3 salary records

-- Select 3 salary records starting from the 4th record

Conclusion

By following this lab, you should now understand how to use various SQL clauses, including SELECT, WHERE, ORDER BY, GROUP BY, HAVING, BETWEEN, LIKE, IN, LIMIT, and OFFSET to query and manipulate data in a database. Practice these queries and try modifying them to deepen your understanding of SQL.