SQL SELECT & Filtering

# 1. SELECT Statement

The SELECT statement is the foundation of SQL. It is used to retrieve data from a database and store the results in a virtual table (called a result set).

**Syntax:**

|  |
| --- |
| SELECT column1, column2, ... FROM table\_name; |

**- SELECT:** Specifies which columns you want to display.  
**- FROM:** Tells SQL from which table the data should be retrieved.

**Example 1:**

|  |
| --- |
| SELECT first\_name, last\_name, department FROM employee; |

**Explanation:** This retrieves the first\_name, last\_name, and department columns from the employee table. The result is a list of all employees with these details.

# 2. Filtering Data with WHERE

The WHERE clause is used to filter rows based on conditions. Only rows that meet the condition will be included in the result.

**Syntax:**

|  |
| --- |
| SELECT column1, column2, ... FROM table\_name WHERE condition; |

**Common Operators:**

|  |
| --- |
| 1. **= :** Equal to. 2. **< > or != :** Not Equal to. 3. **> :** Greater than. 4. **< :** Less than. 5. **>= :** Greater than or Equal to. 6. **<= :** Less than or Equal to. 7. **BETWEEN :** Between a Range of Values. 8. **LIKE :** Search for a Pattern in String. 9. **IN :** Match any Value in a List. 10. **IS NULL :** Check for NULL value. |

**Example:**

**Q1: Retrieve employees whose salary is greater than 60,000 USD?**

|  |
| --- |
| SELECT first\_name, last\_name, salary FROM employee WHERE salary > 60000; |

**Explanation:** This retrieves employees whose salary is greater than 60,000. SQL filters row by row, including only those meeting the condition.

# 3. Combining Conditions with AND / OR

When multiple conditions are required, AND and OR can be used. Parentheses () are critical to ensure correct logic.

* **AND:** Returns rows where both conditions are true.
* **OR:** Returns rows where either condition is true.
* **Parentheses ():** Used to group conditions and ensure the desired logical evaluation.

**Example 1:**

**Q2: Retrieve employees who work in either the Sales or Marketing departments and earn more than 70,000 USD?**

|  |
| --- |
| SELECT first\_name, last\_name, department, salary FROM employee WHERE (department = 'Sales' OR department = 'Marketing') AND salary > 70000; |

**Explanation:** The OR ensures employees from either Sales or Marketing are considered. The AND ensures only those earning above 70,000 are selected. Parentheses make sure department check happens first.

**Example 2:**

**Q3: Find employees from HR department who earn more than 60,000 USD or employees from Operations department who earn more than 75,000 USD?**

|  |
| --- |
| SELECT first\_name, last\_name, department, salary FROM employee WHERE (department = 'HR' AND salary > 60000)  OR (department = 'Operations' AND salary > 75000); |

**Explanation:** This applies different salary rules depending on the department. Parentheses ensure that the rules are applied correctly.

# 4. Sorting with ORDER BY

The ORDER BY clause sorts results in ascending (ASC) or descending (DESC) order.

**Syntax:**

|  |
| --- |
| SELECT column1, column2 FROM table\_name ORDER BY column1 [ASC|DESC], column2 [ASC|DESC]; |

**Example 1:**

**Q4: Retrieve employees and sort them by salary in descending order?**

|  |
| --- |
| SELECT first\_name, last\_name, salary FROM employee ORDER BY salary DESC; |

**Explanation:** Employees are displayed in order of their salary from highest to lowest.

**Example 2 (Multiple Columns):**

**Q5: Retrieve employees & sort them by their salary in descending order, & department in ascending so the most recently hired employees appear first?**

|  |
| --- |
| SELECT first\_name, last\_name, department, salary FROM employee ORDER BY department ASC, salary DESC; |

**Explanation:** Employees are grouped by department alphabetically. Within each department, employees are listed by salary from highest to lowest.

Adding multiple columns in the ORDER BY clause gives you more control over how the results is ordered and presented, especially when you need to highlight specific patterns or rankings within groups of data.

# 5. Aliasing with AS

Aliasing renames columns or tables in the result for better readability. Aliases exist only for the query’s output.

**Example:**

|  |
| --- |
| SELECT first\_name AS "First Name",  last\_name AS "Last Name",  salary AS "Annual Salary" FROM employee; |

**Explanation:** Column names in the output are displayed as First Name, Last Name, and Annual Salary. This is especially useful in reports and dashboards.

# 6. LIMIT and OFFSET (Pagination)

The LIMIT clause restricts how many rows are shown. OFFSET skips rows before starting to display results.

**Syntax:**

|  |
| --- |
| SELECT column1, column2 FROM table\_name LIMIT number\_of\_rows OFFSET skip\_rows; |

**Example:**

**Q6: Retrieve the first 5 employees, skipping the first 10 records?**

|  |
| --- |
| SELECT first\_name, last\_name, department FROM employee LIMIT 5 OFFSET 10; |

**Explanation:** Skips the first 10 rows and shows the next 5. Useful for pagination (e.g., displaying page 3 of results).

**⚠️ Edge Cases with LIMIT/OFFSET:**

1. **Performance Issues:** Large OFFSET values force SQL to scan many rows before returning results → slower queries.

**2. Inconsistent Results:** If data changes between paginated queries, rows may repeat or be skipped.

**3. Non-Deterministic Ordering:** Always use ORDER BY to ensure predictable results when paginating.

**4. Alternative (Keyset Pagination):**

|  |
| --- |
| SELECT first\_name, last\_name FROM employee WHERE id > last\_seen\_id ORDER BY id LIMIT 10; |

# 7. Extra Useful Features

**a) DISTINCT:** Removes duplicates from results.

**Example:**

|  |
| --- |
| SELECT DISTINCT department FROM employee; |

**Explanation:** Shows each department only once, even if multiple employees belong to it.

**b) NULL Filtering**

**Example:**

|  |
| --- |
| SELECT first\_name, last\_name FROM employee WHERE department IS NULL; |

**Explanation:** Retrieves employees without a department assigned.

**c) Finding Nth Highest Value**  
**Example:**

|  |
| --- |
| SELECT first\_name, last\_name, salary FROM employee ORDER BY salary DESC LIMIT 1 OFFSET 1; |

**Explanation:** Retrieves the 2nd highest salary (skips the top row and shows the next one).

# 8. Summary Table

|  |  |  |
| --- | --- | --- |
| **Clause** | **Purpose** | **Example** |
| **SELECT** | Columns to retrieve | SELECT first\_name, salary |
| **FROM** | Table to query | FROM employee |
| **WHERE** | Filtering rows | WHERE salary > 60000 |
| **AND / OR** | Combine conditions | WHERE dept='Sales' AND salary>50000 |
| **ORDER BY** | Sorting | ORDER BY salary DESC |
| **AS** | Aliasing | salary AS "Annual Salary" |
| **DISTINCT** | Remove duplicates | SELECT DISTINCT department |
| **LIMIT / OFFSET** | Pagination | LIMIT 5 OFFSET 10 |
| **IS NULL** | Missing values | WHERE department IS NULL |
| **BETWEEN** | Range | WHERE salary BETWEEN 50000 AND 90000 |