- **The `SELECT` keyword is fundamental in SQL queries, used to select and return data from a database.**

- **It is always present in SQL queries.**

2. **Basic Syntax of `SELECT` Statement:**

Syntax:

- Example: To select the "First Name" from the "Actor" table:

```sql

SELECT FirstName FROM Actor;

```

- The result is the "FirstName" column with its data entries.

3. \*\*Selecting Multiple Columns:\*\*

- Multiple columns can be selected by separating them with commas:

```sql

SELECT FirstName, LastName FROM Actor;

```

- Note: No comma is placed before the `FROM` keyword.

4. \*\*Selecting All Columns:\*\*

- To select all columns from a table without listing them:

```sql

SELECT \* FROM table\_name;

```

- The `\*` (asterisk) symbol is used to indicate all columns.

5. \*\*Code Formatting:\*\*

- SQL code formatting is flexible; it can be written in a single line or across multiple lines.

- The formatting does not affect the execution of the query.

The `**ORDER BY**` clause in SQL **is used to sort the results of a query based on one or more columns.** The sorting can be done alphabetically, numerically, or chronologically, depending on the data type of the column being ordered.

Syntax:

SELECT column1, column2, ...

FROM table\_name

ORDER BY column\_name [ASC|DESC];

Key Concepts:

1. \*\*Basic Usage\*\*:

- The `ORDER BY` clause is placed at the end of a SQL `SELECT` statement.

- You specify the column by which you want to order the results immediately after the `ORDER BY` keyword.

2. \*\*Sorting Order\*\*:

- By default, `ORDER BY` sorts the data in \*\*ascending\*\* order (A-Z, 0-9, earliest to latest dates).

- You can explicitly specify ascending order using `ASC`, although it's optional.

- To sort in \*\*descending\*\* order (Z-A, 9-0, latest to earliest dates), use the `DESC` keyword.

3. \*\*Example\*\*:

- To sort by a column named `first\_name`:

```sql

SELECT first\_name, last\_name

FROM employees

ORDER BY first\_name;

```

- To sort by `first\_name` in descending order:

```sql

SELECT first\_name, last\_name

FROM employees

ORDER BY first\_name DESC;

```

4. \*\*Multiple Columns\*\*:

- When there are duplicate values in the column used for sorting, you can specify additional columns to refine the sorting.

- Example: Sorting first by `first\_name` and then by `last\_name`:

```sql

SELECT first\_name, last\_name

FROM employees

ORDER BY first\_name ASC, last\_name ASC;

```

- You can mix sorting orders:

```sql

SELECT first\_name, last\_name

FROM employees

ORDER BY first\_name ASC, last\_name DESC;

```

5. \*\*Practical Example\*\*:

- Consider a `payment` table with `customer\_id` and `amount` columns.

- To order the payments first by `customer\_id` and then by `amount` in descending order:

```sql

SELECT \*

FROM payment

ORDER BY customer\_id ASC, amount DESC;

```

- This will sort the results by `customer\_id` in ascending order and then, for each customer, by `amount` in descending order.

6. \*\*Why Specify `ASC`?\*\*:

- Even though `ASC` is the default sorting order, explicitly specifying it can improve code readability. It makes it clear that the sort order is intentional, which can be helpful when revisiting the code later.