-Basic SELECT Statements

>SELECT * FROM sakila.actor;

SELECT * means fetch all columns
This returns every row from the actor table
Good for exploring a table, not ideal for production use

>SELECT DISTINCT first_name FROM sakila.actor;

DISTINCT removes duplicate values
Returns unique first names only
Useful when checking repeated values in a column

-Selecting Specific Columns

SELECT first_name, last_name FROM sakila.actor;

Selects only the required columns
Faster and cleaner than SELECT *
Best practice in real projects

-COUNT and DISTINCT COUNT

>SELECT COUNT(*) FROM sakila.film;

Counts total number of rows
Includes rows even if columns contain NULL

>SELECT COUNT(first_name) FROM sakila.actor;

Counts non-NULL values in first_name
If column has no NULLs, result = total rows

>SELECT COUNT(DISTINCT first_name) FROM sakila.actor;

Counts unique first names
Ignores duplicates

Differences:

COUNT(*) → total rows
COUNT(DISTINCT column) → unique values only

-Working with NULL

>SELECT * FROM sakila.film
WHERE original_language_id IS NULL;

Filters rows where value is missing
Always use IS NULL or IS NOT NULL
= NULL does NOT work in SQL

>SELECT rental_id, customer_id, return_date
FROM sakila.rental
WHERE return_date IS NULL;

Finds rentals that were never returned
Very common real-world query

-LIMIT

>SELECT first_name, last_name
FROM sakila.actor
LIMIT 50;

Limits number of rows returned
Mostly used for testing or previewing data

-WHERE Clause (Row Filtering)

>SELECT * FROM sakila.film
WHERE rating = 'R' AND length >= 92;

Filters rows based on conditions
Works on individual rows
Executes before grouping

-ORDER BY (Sorting Results)

>SELECT rental_rate
FROM sakila.film
ORDER BY rental_rate DESC;

Sorts the output ASC ,DESC
ORDER BY runs after SELECT

-AND vs OR

>WHERE rating = 'PG' AND rental_duration = 5;
Both conditions must be true

>WHERE rating = 'PG' OR rental_duration = 5;
Either condition can be true

brackets for multiplie operations:

>WHERE rental_duration = 6
AND (rating = 'G' OR rating = 'PG');

-LIKE Operator (Pattern Matching)

% → zero or more characters

_ → exactly one character

>SELECT city FROM sakila.city
WHERE city LIKE 'A%';

Cities starting with A

>WHERE city LIKE '_s__d%';

Matches characters by position

-NOT and NOT IN

>WHERE rental_duration NOT IN (6, 7, 3);

Excludes listed values

>WHERE NOT rental_duration = 6;

Reverses condition

-BETWEEN

>WHERE return_date
BETWEEN '2005-05-26' AND '2005-05-30';

Inclusive of both values
Can be used with dates and numbers

-GROUP BY

>SELECT customer_id, COUNT(*) AS count_rentals
FROM sakila.rental
GROUP BY customer_id;

Groups rows into summary rows
Required when using aggregate functions
One row per customer_id

-HAVING Clause

>HAVING COUNT(*) <= 30;

Filters grouped results
Always used with GROUP BY
Cannot replace WHERE

-WHERE vs HAVING
WHERE HAVING
Filters rows Filters groups
Before GROUP BY After GROUP BY
No aggregates Aggregates allowed

Example:

WHERE customer_id = 33;

HAVING SUM(amount) > 100;

-Aliasing (AS)

>COUNT(*) AS count

SUM(amount) AS total_payment

Renames columns in output

Makes results readable

Can be used in ORDER BY

-SQL Execution Order

FROM

→ JOIN

→ WHERE

→ GROUP BY

→ HAVING

→ SELECT

→ ORDER BY

→ LIMIT