**SQL commands notes**

 **SELECT DISTINCT**

* **Usage**: Retrieves unique values from a column.
* **Example**:

SELECT DISTINCT amount

FROM payment;

 **WHERE**

* **Usage**: Filters rows based on specified conditions.
* **Example**:

SELECT \*

FROM payment

WHERE amount = 0;

 **COUNT(\*)**

* **Usage**: Counts the number of rows that match a specific condition.
* **Example**:

SELECT COUNT(\*)

FROM payment

WHERE customer\_id = 100;

 **SELECT with AND and OR**

* **Usage**: Combines multiple conditions to filter data.
* **Example**:

SELECT \*

FROM payment

WHERE amount = 10.99 OR amount = 9.99

AND customer\_id = 439

ORDER BY amount;

 **ORDER BY**

* **Usage**: Sorts the result set by one or more columns.
* **Example**:

SELECT \*

FROM payment

WHERE (amount = 10.99 OR amount = 9.99)

AND customer\_id = 426

ORDER BY amount;

 **LIMIT**

* **Usage**: Restricts the number of rows returned by the query.
* **Example**:

SELECT payment\_id, amount

FROM payment

WHERE amount <= 2

LIMIT 10;

 **IS NULL**

* **Usage**: Checks for NULL values in a column.
* **Example**:

SELECT COUNT(\*)

FROM rental

WHERE return\_date IS NULL;

 **ROUND()**

* **Usage**: Rounds a numeric value to a specified number of decimal places.
* **Example**:

SELECT ROUND(AVG(amount), 2) AS average, SUM(amount)

FROM payment;

 **AVG()**

* **Usage**: Calculates the average value of a numeric column.
* **Example**:

SELECT ROUND(AVG(replacement\_cost), 2) AS AVG

FROM film;

 **SUM()**

* **Usage**: Adds up the values in a numeric column.
* **Example**:

SELECT SUM(amount)

FROM payment;

 **MIN() and MAX()**

* **Usage**: Finds the minimum and maximum values in a numeric column.
* **Example**:

SELECT MIN(replacement\_cost), MAX(replacement\_cost)

FROM film;

 **COUNT(\*)**

* **Usage**: Counts the number of rows or non-NULL values in a column.
* **Example**:

SELECT staff\_id, COUNT(\*), SUM(amount) AS payment\_amount

FROM payment

WHERE amount != 0

GROUP BY staff\_id;

 **!=**

* **Usage**: Checks if a value is not equal to another value.
* **Example**:

SELECT staff\_id, COUNT(\*), SUM(amount) AS payment\_amount

FROM payment

WHERE amount != 0

GROUP BY staff\_id;

 **UPPER() and LOWER()**

* **Usage**: Converts all characters in a string to uppercase or lowercase.
* **Example**:

SELECT UPPER(email) AS email\_upper, LOWER(email) AS email\_lower

FROM customer;

 **LENGTH()**

* **Usage**: Returns the length of a string (number of characters).
* **Example**:

SELECT LENGTH(email)

FROM customer

WHERE LENGTH(email) < 30;

 **LEFT() and RIGHT()**

* **Usage**: Extracts a specified number of characters from the left or right side of a string.
* **Example**:

SELECT LEFT(first\_name, 1), RIGHT(email, 5)

FROM customer;

 **|| (Concatenation Operator)**

* **Usage**: Concatenates two or more strings together.
* **Example**:

SELECT LEFT(first\_name, 1) || '.' || LEFT(last\_name, 1) || '.' AS initials

FROM customer;

 **OR**

* **Usage**: Combines multiple conditions in a WHERE clause; at least one condition must be true.
* **Example**:

SELECT LOWER(first\_name), LOWER(last\_name), LOWER(email)

FROM customer

WHERE LENGTH(first\_name) > 10 OR LENGTH(last\_name) > 10;

 **AS**

* **Usage**: Assigns an alias to a column or expression in the result set.
* **Example**:

SELECT LEFT(first\_name, 1) || '.' || LEFT(last\_name, 1) || '.' AS initials

FROM customer;

 **ROUND()**

* **Usage**: Rounds a numeric value to a specified number of decimal places.
* **Example**:

SELECT ROUND(9.0 / 4, 2);

 **MOD / % (Modulo Operator)**

* **Usage**: Returns the remainder of a division.
* **Example**:

SELECT 10 % 4;

 **ABS(), CEILING(), FLOOR()**

* **Usage**:
  + ABS(x) returns the absolute value of x.
  + CEILING(x) returns the smallest integer greater than or equal to x.
  + FLOOR(x) returns the largest integer less than or equal to x.
* **Example**:

SELECT ABS(-5), CEILING(4.3), FLOOR(4.7);

 **CASE WHEN**

* **Usage**: Creates conditional logic within a query to return different values based on certain conditions.
* **Example**:

SELECT amount,

CASE

WHEN amount < 2 THEN 'low amount'

WHEN amount < 5 THEN 'medium amount'

ELSE 'high amount'

END

FROM payment;

 **LIKE**

* **Usage**: Performs pattern matching in string comparisons.
* **Example**:

SELECT title,

CASE

WHEN description LIKE '%Drama%' AND film.length > 90 THEN 'Long drama (tier 2)'

WHEN description LIKE '%Drama%' AND film.length < 90 THEN 'Short drama (tier 3)'

END

FROM film;

 **INNER JOIN**

* **Usage**: Combines rows from two or more tables based on a related column, returning only the rows that have matching values in both tables.
* **Example**:

SELECT s.fare\_conditions AS "Fare Conditions", COUNT(\*) AS "Count"

FROM boarding\_passes bp

INNER JOIN flights f ON bp.flight\_id = f.flight\_id

INNER JOIN seats s ON f.aircraft\_code = s.aircraft\_code AND bp.seat\_no = s.seat\_no

GROUP BY s.fare\_conditions

ORDER BY 2 DESC;

 **FULL OUTER JOIN**

* **Usage**: Returns all records when there is a match in either left or right table records; records without a match in either table are included with NULL values.
* **Example**:

SELECT \*

FROM boarding\_passes b

FULL OUTER JOIN tickets t ON b.ticket\_no = t.ticket\_no

WHERE b.ticket\_no IS NULL;

 **LEFT JOIN (Left Outer Join)**

* **Usage**: Returns all records from the left table and the matched records from the right table; if there is no match, NULL values are returned for columns from the right table.
* **Example**:

SELECT \*

FROM aircrafts\_data AS a

LEFT JOIN flights f ON a.aircraft\_code = f.aircraft\_code;

 **UNION**

* **Usage**: Combines the result sets of two or more SELECT statements into a single result set, eliminating duplicate rows.
* **Example**:

SELECT first\_name, 'actor' AS origin FROM actor

UNION

SELECT first\_name, 'customer' FROM customer

UNION

SELECT first\_name, 'staff' FROM staff

ORDER BY 2 DESC;

 **Subqueries (in WHERE clause)**

* **Usage**: A subquery is a query within another query, typically used to filter results based on a condition that involves a separate SELECT statement.
* **Example**:

SELECT \*

FROM payment

WHERE amount > (SELECT AVG(amount) FROM payment);

 **IN (with Subquery)**

* **Usage**: Checks if a value matches any value in a list or subquery result.
* **Example**:

SELECT \*

FROM payment

WHERE customer\_id IN (SELECT customer\_id FROM customer WHERE first\_name LIKE 'A%');

 **HAVING**

* **Usage**: Filters results based on an aggregate function, typically used with GROUP BY.
* **Example**:

SELECT film\_id

FROM inventory

WHERE store\_id = 2

GROUP BY film\_id

HAVING COUNT(\*) > 3;

 **DATE()**

* **Usage**: Extracts the date part of a datetime or timestamp value.
* **Example**:

SELECT first\_name, last\_name

FROM customer

WHERE customer\_id IN (

SELECT customer\_id

FROM payment

WHERE DATE(payment\_date) = '2020-01-25'

);

 **CREATE TABLE**

* **Usage**: Creates a new table in the database with specified columns and constraints.
* **Example**:

CREATE TABLE director(

director\_id SERIAL PRIMARY KEY,

director\_account\_name VARCHAR(20) UNIQUE,

first\_name VARCHAR(50),

last\_name VARCHAR(50) DEFAULT 'Not specified',

date\_of\_birth DATE,

address\_id INT REFERENCES address(address\_id)

);

 **SERIAL**

* **Usage**: Automatically generates a unique identifier for a column (usually used for primary keys).
* **Example**:

CREATE TABLE online\_sales(

transaction\_id SERIAL PRIMARY KEY,

customer\_id INT REFERENCES customer(customer\_id),

film\_id INT REFERENCES film(film\_id),

amount NUMERIC(5,2) NOT NULL

);

 **INSERT INTO**

* **Usage**: Inserts new rows into a table.
* **Example**:

INSERT INTO online\_sales (customer\_id, film\_id, amount, promotion\_code)

VALUES (124, 65, 14.99, 'PROMO2022'), (225, 231, 12.99, 'JULYPROMO');

 **ALTER TABLE**

* **Usage**: Modifies an existing table structure, such as adding, modifying, or dropping columns.
* **Example**:

ALTER TABLE director

ALTER COLUMN director\_account\_name TYPE VARCHAR(30),

ALTER COLUMN last\_name DROP DEFAULT,

ALTER COLUMN last\_name SET NOT NULL,

ADD COLUMN IF NOT EXISTS email VARCHAR(40);

 **RENAME COLUMN**

* **Usage**: Renames an existing column in a table.
* **Example**:

ALTER TABLE director

RENAME COLUMN director\_account\_name TO account\_name;

 **RENAME TO**

* **Usage**: Renames a table.
* **Example**:

ALTER TABLE director

RENAME TO directors;

 **REFERENCES**

* **Usage**: Creates a foreign key constraint that references another table.
* **Example**:

CREATE TABLE online\_sales(

customer\_id INT REFERENCES customer(customer\_id),

film\_id INT REFERENCES film(film\_id)

);

 UPDATE

**Usage**: The UPDATE statement is used to modify the existing records in a table. You can update one or more columns at a time and optionally use a WHERE clause to specify which rows should be updated.

**Example**:

UPDATE customer

SET email = 'new\_email@example.com'

WHERE customer\_id = 123;

 DELETE

**Usage**: The DELETE statement is used to remove one or more records from a table. The WHERE clause is used to specify the conditions that must be met for a row to be deleted. If the WHERE clause is omitted, all rows in the table will be deleted.

**Example**:

DELETE FROM customer

WHERE customer\_id = 123;

 CREATE TABLE AS

**Usage**: The CREATE TABLE AS statement is used to create a new table based on the result of a SELECT query. The structure and data of the new table are derived from the query result.

**Example**:

CREATE TABLE high\_spenders AS

SELECT customer\_id, SUM(amount) AS total\_spent

FROM payment

GROUP BY customer\_id

HAVING SUM(amount) > 1000;

 CREATE VIEW

**Usage**: A VIEW is a virtual table based on the result set of a SELECT query. The CREATE VIEW statement creates a new view in the database.

**Example**:

CREATE VIEW customer\_payments AS

SELECT c.first\_name, c.last\_name, SUM(p.amount) AS total\_payments

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

GROUP BY c.first\_name, c.last\_name;

 CREATE MATERIALIZED VIEW

**Usage**: A MATERIALIZED VIEW stores the result of a query physically and can be refreshed as needed. It is different from a regular VIEW in that it stores the data rather than just the query.

**Example**:

CREATE MATERIALIZED VIEW total\_revenue\_by\_category AS

SELECT c.name AS category\_name, SUM(p.amount) AS total\_revenue

FROM category c

INNER JOIN film\_category fc ON c.category\_id = fc.category\_id

INNER JOIN film f ON fc.film\_id = f.film\_id

INNER JOIN inventory i ON f.film\_id = i.film\_id

INNER JOIN rental r ON i.inventory\_id = r.inventory\_id

INNER JOIN payment p ON r.rental\_id = p.rental\_id

GROUP BY c.name;

**Managing Views**

 DROP VIEW

**Usage**: The DROP VIEW statement is used to delete an existing view from the database.

**Example**:

DROP VIEW customer\_payments;

 ALTER VIEW - RENAME COLUMN

**Usage**: The ALTER VIEW statement can be used to modify the structure of a view. However, most databases do not support directly altering the columns of a view. Instead, you would typically need to drop and recreate the view with the desired changes.

**Example**:

CREATE OR REPLACE VIEW customer\_payments AS

SELECT c.first\_name AS fname, c.last\_name AS lname, SUM(p.amount) AS total\_payments

FROM customer c

INNER JOIN payment p ON c.customer\_id = p.customer\_id

GROUP BY c.first\_name, c.last\_name;

 ALTER VIEW - RENAME TO

**Usage**: The ALTER VIEW ... RENAME TO statement renames an existing view.

**Example**:

ALTER VIEW customer\_payments RENAME TO customer\_total\_payments;