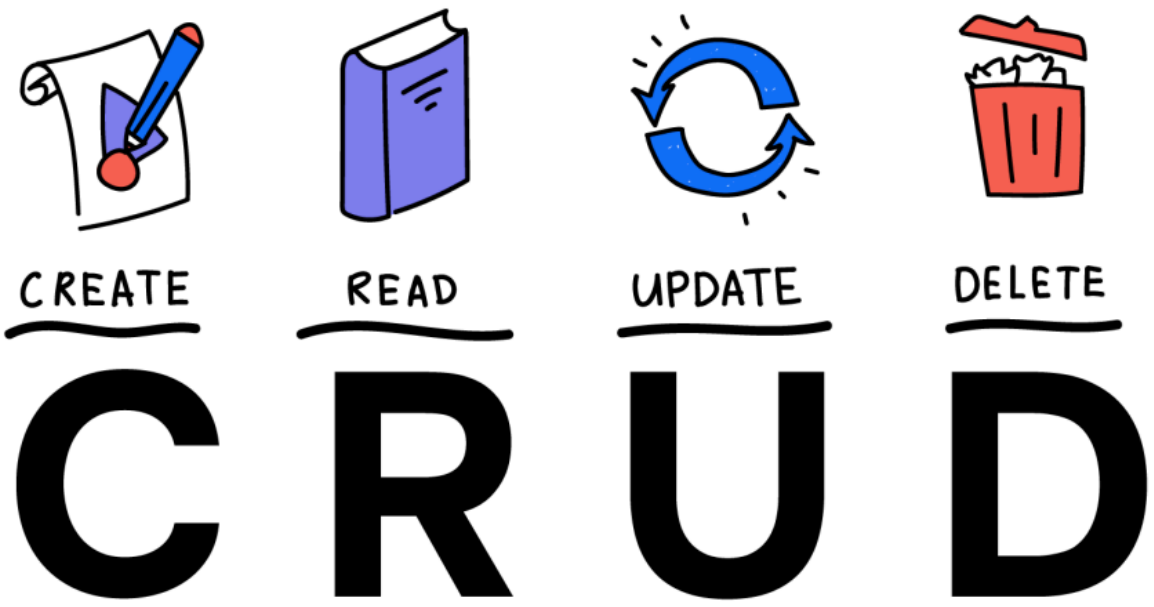
STRUCTURED QUERY LANGUAGE

What is SQL?

SQL (Structured Query Language) is a programming language used to interact with database.

SQL Application



CRUD is an acronym for CREATE, READ(SELECT), UPDATE, and DELETE statements in SQL

Relational Database	Non-Relational Database
SQL database	NoSQL database
Data stored in tables	Data stored are either key-value pairs, document-based, graph databases or wide-column stores
These databases have fixed or static or predefined schema	They have dynamic schema
Low performance with huge volumes of data	Easily work with huge volumes of data
Eg: PostgreSQL, MySQL, MS SQL Server	Eg: MongoDB, Cassandra, Hbase

SQL Commands:

There are mainly 3 types of SQL commands:

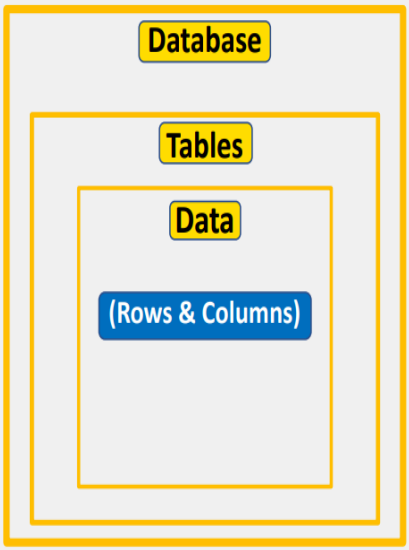
- DDL (Data Definition Language): create, alter, and drop
- DML (Data Manipulation Language): select, insert, update and delete
- DCL (Data Control Language): grant and revoke permission to users

What is Database?

Database is a system that allow users to store and organise data

SQL Databases

SQL Structure



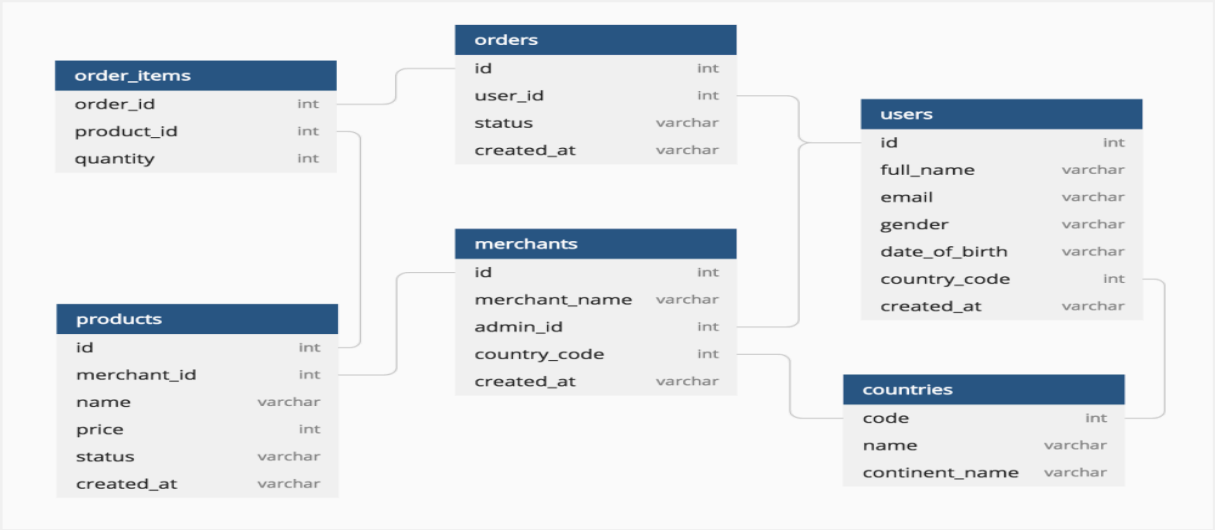
Columns

Rows

Example

```
mysql> SELECT* FROM employee;
+--------+----------+------------------+---------+---------+
| emp_id | emp_name | emp_dept         | emp_age | emp_sex |
+--------+----------+------------------+---------+---------+
| E00001 | JHONNY   | BACKEND DEVELOPER |      26 | male    |
| E00002 | DARSHI   | NULL             |      27 | male    |
| E00003 | JASMINE  | NULL             |      37 | female  |
| E00004 | LILLY    | NULL             |      47 | female  |
| E00005 | RONALD   | UI DEVELOPER     |      26 | male    |
+--------+----------+------------------+---------+---------+
5 rows in set (0.01 sec)
```

Database Diagram



- SQL Commands:
- CREATE
- INSERT
- UPDATE
- BACKUP
- DELETE
- ALTER
- DROP, TRUNCATE

Data Types in SQL

Definition: Specifies the type of values a column can store in a table.

Defined: During table creation in a database.

Main Categories:

String – CHAR (fixed length), VARCHAR (variable length).

Numeric – INT (integer), FLOAT (decimal number), BOOL (true/false).

Date & Time – DATE (YYYY-MM-DD), DATETIME (YYYY-MM-DD hh:mm:ss).

Primary Key (PK) & Foreign Key (FK)

Primary Key (PK): A unique column in a table used to identify records. A table can have only one PK, and it must be unique and NOT NULL.

Foreign Key (FK): A column that links two or more tables. A table can have multiple FKs, which can contain duplicate and NULL values.

SQL Constraints

Definition: Rules applied to table columns to ensure data accuracy and reliability.

Defined: During table creation (CREATE TABLE) or after (ALTER TABLE).

Common SQL Constraints:

NOT NULL – Prevents NULL values in a column.

UNIQUE – Ensures all values in a column are distinct.

PRIMARY KEY – Combines NOT NULL and UNIQUE, uniquely identifying records.

FOREIGN KEY – Links multiple tables and maintains referential integrity.

CHECK – Ensures values meet a specific condition.

DEFAULT – Assigns a default value if none is provided.

CREATE INDEX – Improves data retrieval speed.

CREATE TABLE in SQL

Definition: Used to create a new table in a database.

Syntax:

```sql
CREATE TABLE table_name (
  column_name1 datatype constraint,
  column_name2 datatype constraint,
  column_name3 datatype constraint
);
```

```sql
CREATE TABLE customer (
  CustID int8 PRIMARY KEY,
  CustName varchar(50) NOT NULL,
  Age int NOT NULL,
  City char(50),
  Salary numeric
);
```

INSERT INTO in SQL

Definition: Used to insert new records into a table.

Syntax:

```sql
INSERT INTO table_name (column1, column2, ..., columnN)
VALUES (value1, value2, ..., valueN);
```

```sql
INSERT INTO customer (CustID, CustName, Age, City, Salary)
VALUES
(1, 'Sam', 26, 'Delhi', 9000),
(2, 'Ram', 19, 'Bangalore', 11000),
(3, 'Pam', 31, 'Mumbai', 6000),
(4, 'Jam', 42, 'Pune', 10000);
```

Update Values In Table

The UPDATE command is used to update existing rows in a table

- Syntax

```
UPDATE TABLE_NAME
SET "Column_name1" = 'value1', "Column_name2" = 'value2'
WHERE "ID" = 'value'
```

Example

```
UPDATE customer
SET CustName = 'Xam', Age= 32
WHERE CustID = 4;
```

ALTER TABLE in SQL

Definition: Used to add, delete, or modify columns in an existing table.

Syntax & Examples

- **Add Column**:

```sql
ALTER TABLE table_name ADD COLUMN column_name datatype;
```

Example:

```sql
ALTER TABLE customer ADD COLUMN Gender varchar(10);
```

Modify Column:

```sql
ALTER TABLE table_name ALTER COLUMN column_name datatype;
```

Example:

```sql
ALTER TABLE customer ALTER COLUMN Gender char(10);
```

Drop Column:

```sql
ALTER TABLE table_name DROP COLUMN column_name;
```

Example:

```sql
ALTER TABLE customer DROP COLUMN Gender;
```

Delete Values In Table

The DELETE statement is used to delete existing records in a table

Syntax

```
DELETE FROM table_name WHERE condition;
```

Example

```
DELETE FROM customer
WHERE CustID = 3;
```

Drop & Truncate Table

The DROP TABLE command deletes a table in the database

- Syntax

```
DROP TABLE table_name;
```

The TRUNCATE TABLE command deletes the data inside a table, but not the table itself

- Syntax

```
TRUNCATE TABLE table_name;
```

SELECT Statement

The SELECT statement is used to select data from a database.

- Syntax

```
SELECT column_name FROM table_name;
```

To select all the fields available in the table

- Syntax

```
SELECT * FROM table_name;
```

To select distinct/unique fields available in the table

- Syntax

```
SELECT DISTINCT Column_name FROM table_name;
```

WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition

Syntax

```
SELECT column_name FROM table_name
WHERE conditions;
```

Example

```
SELECT name FROM classroom
WHERE grade='A';
```

SQL Operators

Definition: Reserved words and symbols used in SQL queries, mainly with the WHERE clause.

Types of Operators:

1.Arithmetic Operators – Perform calculations on numeric values.

Examples: +, -, *, /, %

2.Comparison Operators – Compare values in a table.

Examples: =, !=, >, >=

3.Logical Operators – Perform Boolean operations.

Examples: ALL, IN, BETWEEN, LIKE, AND, OR, NOT, ANY

4.Bitwise Operators – Perform bit-level operations on integers.

Examples: & (AND), | (OR)

LIMIT Clause

The LIMIT clause is used to set an upper limit on the number of tuples returned by SQL.

Example: below code will return 5 rows of data

```
SELECT column_name FROM table_name
LIMIT 5;
```

ORDER BY Clause

The ORDER BY is used to sort the result-set in ascending (ASC) or descending order (DESC).

Example: below code will sort the output data by column name in ascending order

```
SELECT column_name FROM table_name
ORDER BY column_name e ASC;
```

SQL Functions

Definition: A set of SQL statements that perform a specific task and return a result.

Types of Functions:

System-Defined Functions – Built-in SQL functions.

Examples: RAND(), ROUND(), UPPER(), LOWER(), COUNT(), SUM(), AVG(), MAX()

User-Defined Functions – Custom functions created by users, callable like built-in functions.

Most Used String Functions in SQL

Definition: Perform operations on strings and return modified output.

Common String Functions:

UPPER() – Converts text to uppercase.
LOWER() – Converts text to lowercase.
LENGTH() – Returns string length.
SUBSTRING() – Extracts part of a string.
NOW() – Returns current date & time.
FORMAT() – Formats a field's value.
CONCAT() – Joins multiple strings.
REPLACE() – Replaces part of a string.
TRIM() – Removes spaces from both ends.

Most Used Aggregate Functions in SQL

Definition: Perform calculations on multiple values and return a single result.

Often Used With: GROUP BY & SELECT statements.

Common Aggregate Functions:

COUNT() – Returns the number of values.

SUM() – Returns the total sum.

AVG() – Returns the average value.

MAX() – Returns the highest value.

MIN() – Returns the lowest value.

ROUND() – Rounds a number to specified decimal places

GROUP BY Statement

The GROUP BY statement group rows that have the same values into summary rows.

It is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

Syntax

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s);
```

Example

```
SELECT mode, SUM(amount) AS total
FROM payment
GROUP BY mode
```

HAVING Clause

The HAVING clause is used to apply a filter on the result of GROUP BY based on the specified condition.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition(s)
GROUP BY column_name(s)
HAVING condition(s)
```

- **Example**

```
SELECT mode, COUNT(amount) AS total
FROM payment
GROUP BY mode
HAVING COUNT(amount) >= 3
ORDER BY total DESC
```

SQL JOIN

- JOIN means to combine something.
- A JOIN clause is used to combine data from two or more tables, based on a related column between them
- Let's understand the joins through an example:

JOIN Example

Question: How much amount was paid by customer 'Madan', what was mode and payment date?

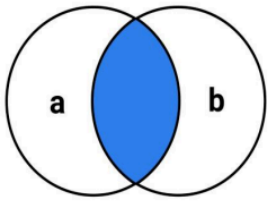
customer_id [PK] bigint	first_name character varying (50)	last_name character varying (50)	address_id bigint
1	Mary	Smith	5
2	Madan	Mohan	6
3	Linda	Williams	7
4	Barbara	Jones	8
5	Elizabeth	Brown	9

customer_id [PK] bigint	amount bigint	mode character varying (50)	payment_date date
1	60	Cash	2020-09-24
2	30	Credit Card	2020-04-27
3	90	Credit Card	2020-07-07
4	50	Debit Crad	2020-02-12
5	40	Mobile Payment	2020-11-20

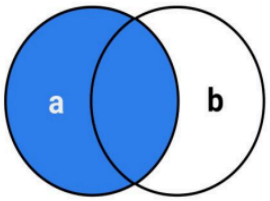
Answer: Amount = 30,
Mode = Credit Card,
Date = 2020-04-27

TYPES OF JOINS

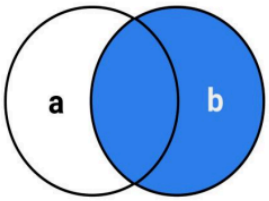
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN



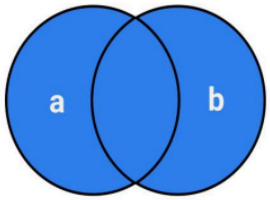
a INNER JOIN b



a LEFT JOIN b



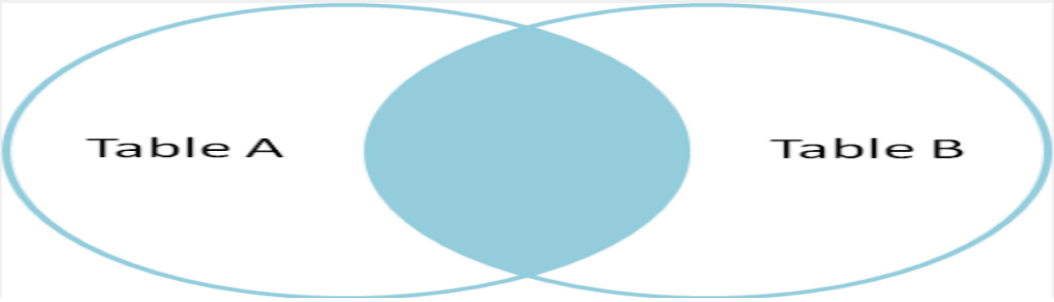
a RIGHT JOIN b



a FULL OUTER JOIN b

INNER JOIN

- Returns records that have matching values in both tables



Syntax

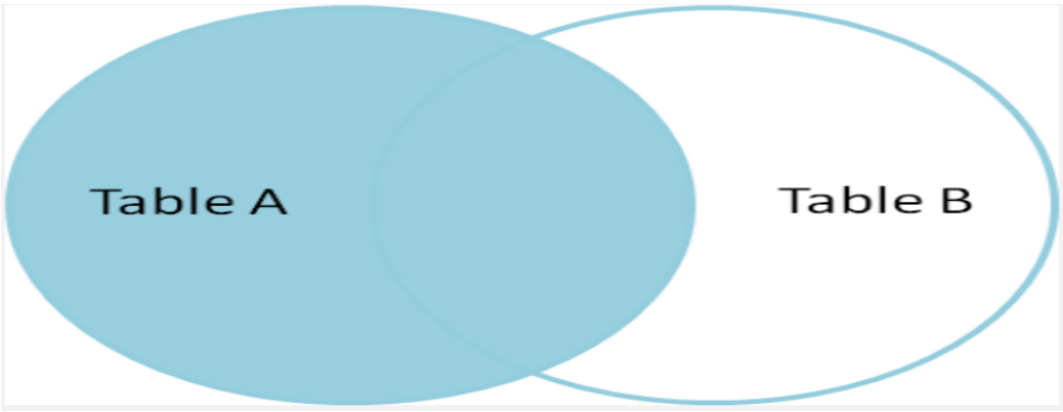
```
SELECT column_name(s)
FROM TableA
INNER JOIN TableB
ON TableA.col_name = TableB.col_name
```

Example

```
SELECT *
FROM customer AS c
INNER JOIN payment AS p
ON c.customer_id = p.customer_id
```

LEFT JOIN

- Returns all records from the left table, and the matched records from the right table



Syntax

```
SELECT column_name(s)
FROM TableA
LEFT JOIN TableB
ON TableA.col_name = TableB.col_name
```

Example

```
SELECT *
FROM customer AS c
LEFT JOIN payment AS p
ON c.customer_id = p.customer_id
```

RIGHT JOIN

- Returns all records from the right table, and the matched records from the left table

Syntax

```
SELECT column_name(s)
FROM TableA
RIGHT JOIN TableB
ON TableA.col_name = TableB.col_name
```

Example

```
SELECT *
FROM customer AS c
RIGHT JOIN payment AS p
ON c.customer_id = p.customer_id
```

FULL JOIN

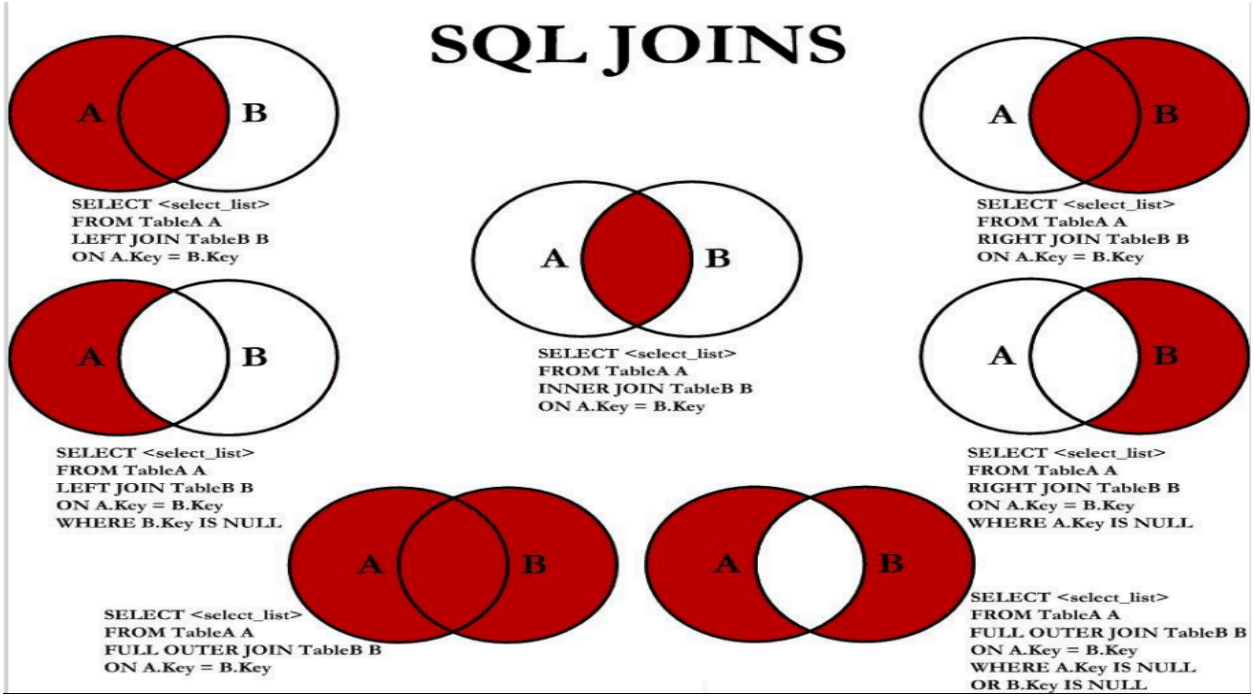
- Returns all records when there is a match in either left or right table

Syntax

```
SELECT column_name(s)
FROM TableA
FULL OUTER JOIN TableB
ON TableA.col_name = TableB.col_name
```

Example

```
SELECT *
FROM customer AS c
FULL OUTER JOIN payment AS p
ON c.customer_id = p.customer_id
```



SELF JOIN

- A self join is a regular join in which a table is joined to itself
- SELF Joins are powerful for comparing values in a column of rows with the same table
- Syntax

```
SELECT column_name(s)
FROM Table AS T1
JOIN Table AS T2
ON T1.col_name = T2.col_name
```

UNION in SQL

Definition: The UNION operator combines the result sets of two or more SELECT queries and removes duplicates.

Condition: Both queries must have the same number of columns with matching data types.

```sql
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers;
```

SUB QUERY

A Subquery or Inner query or a Nested query allows us to create complex query on the output of another query

- Sub query syntax involves two SELECT statements
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator
( SELECT column_name FROM table_name WHERE ... );
```

Question: Find the details of customers, whose payment amount is more than the average of total amount paid by all customers

Divide above question into two parts:

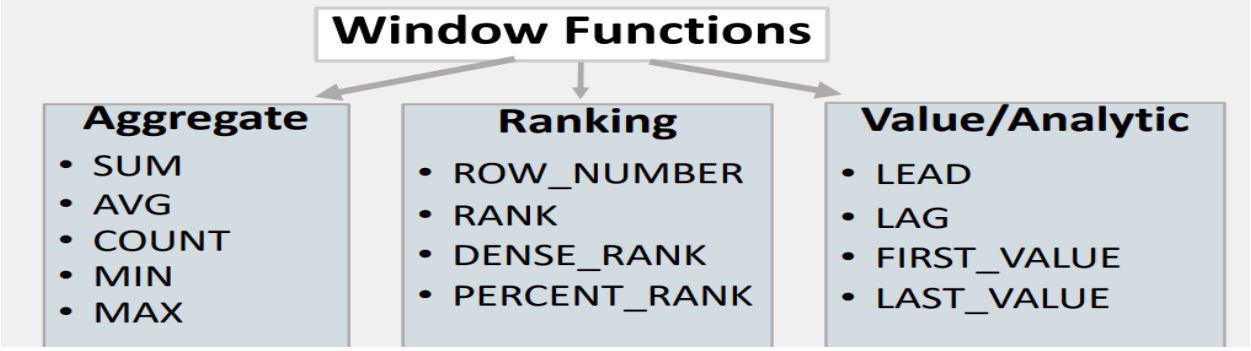
1. Find the average amount
2. Filter the customers whose amount > average amount

	customer_id [PK] bigint	amount bigint	mode character varying (50)	payment_date date
1	1	60	Cash	2020-09-24
2	2	30	Credit Card	2020-04-27
3	8	110	Cash	2021-01-26
4	10	70	mobile Payment	2021-02-28
5	11	80	Cash	2021-03-01

Window Functions in SQL

Definition: Perform calculations across a set of table rows related to the current row without collapsing results like aggregate functions.

Used With: OVER() clause to define a window (subset of rows).




```sql
SELECT Name, Salary,
RANK() OVER (ORDER BY Salary DESC) AS Rank
FROM Employees;
```

Quick Assignment: WINDOW FUNCTION

Offset the LEAD and LAG values by 2 in the output columns ?

INPUT

new_id
100
200
200
300
500
500
700

OUTPUT

new_id	LEAD	LAG
100	200	NULL
200	300	NULL
200	500	100
300	500	200
500	700	200
500	NULL	300
700	NULL	500

```
SELECT new_id,
LEAD(new_id, 2)  OVER( ORDER BY new_id) AS "LEAD_by2",
LAG(new_id, 2)  OVER( ORDER BY new_id) AS "LAG_by2"
FROM test  data
```

new_id	LEAD_by2	LAG_by2
100	200	null
200	300	null
200	500	100
300	500	200
500	700	200
500	null	300
700	null	500

CASE Expression in SQL

Definition: A conditional statement in SQL that works like IF-ELSE, returning specific values based on conditions.

Used In: SELECT, WHERE, ORDER BY clauses.

- **CASE Expression Syntax**

```
CASE Expression
    WHEN value1 THEN result1
    WHEN value2 THEN result2
    WHEN valueN THEN resultN
    ELSE other_result
END;
```

Example

```sql
SELECT Name, Age,
CASE
     WHEN Age < 18 THEN 'Minor'
     WHEN Age BETWEEN 18 AND 60 THEN 'Adult'
     ELSE 'Senior'
END AS AgeGroup
FROM Persons;
```