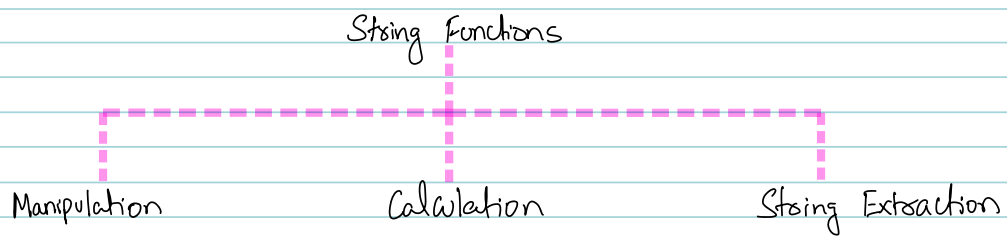
String Functions



Manipulation	Calculation	String Extraction
CONCAT	LEN	LEFT
UPPER		RIGHT
LOWER		SUBSTRING
TRIM		
REPLACE		

CONCAT	Combines multiple strings into one
UPPER	Converts all characters to uppercase
LOWER	Converts all characters to lowercase
TRIM	Removes Leading and Trailing spaces
REPLACE	Replaces specific character with a new character
LEN	Counts how many characters
LEFT	Extracts specific Number of Characters from the start
RIGHT	Extracts specific Number of Characters from the End
Substring	Extracts a part of string at a specified position

★ Syntax- CONCAT

```
SELECT first_name, country
CONCAT (first_name, country) AS name
FROM .....
```

output

firstname	country	name
Alex	China	Alexchina

```
SELECT first_name, country
CONCAT (first_name, ' ', country) AS name
FROM .....
```

output

firstname	country	name
Alex	China	Alex China

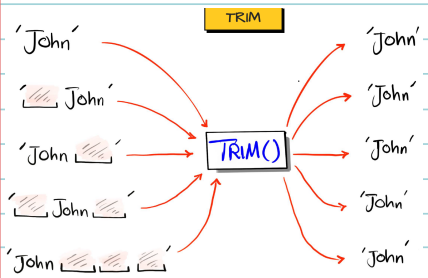
Note: You can add space by mentioning b/w single quotes

★ Syntax UPPER & LOWER

```
SELECT first_name, country
CONCAT (first_name, ' ', country) AS name
LOWER (first_name) AS low
FROM .....
```

} Similarly for upper

★ Syntax TRIM



It basically removes space before, b/w or after Strings

These are three methods

1) Discovers if there any row with blank spaces

```
SELECT
   first_name
FROM customers
WHERE first_name != TRIM(first_name)
```

Here we will retrieve data where the first_name column does not match after trim function is executed. This way we will know if there any blank spaces

```
SELECT
   first_name
   LEN(first_name) As len_name)
   LEN(TRIM(first_name)) len_trim_name
FROM customers
```

output

first_name	len_name	len_trim_name
Maria	6	5

What happens here is that we cross check if there is any changes in the len calculations.

we can resolve the difference by just Subtracting

```
LEN(first_name) As len_name) — LEN(TRIM(first name)) len trim_name As diff.
```

★ Syntax REPLACE

```
SELECT
      Phone_No.
REPLACE (Phone_No., '-', ' ') As Clean_Phone.
```

Here it will replace the "—" in the phone No. & replace it with empty space

★ Syntax RIGHT & LEFT

LEFT - Starts from start of a string
Right - " " End " " "

```
SELECT
   first_name
   LEFT (first_name, 2) first_2_chars
```

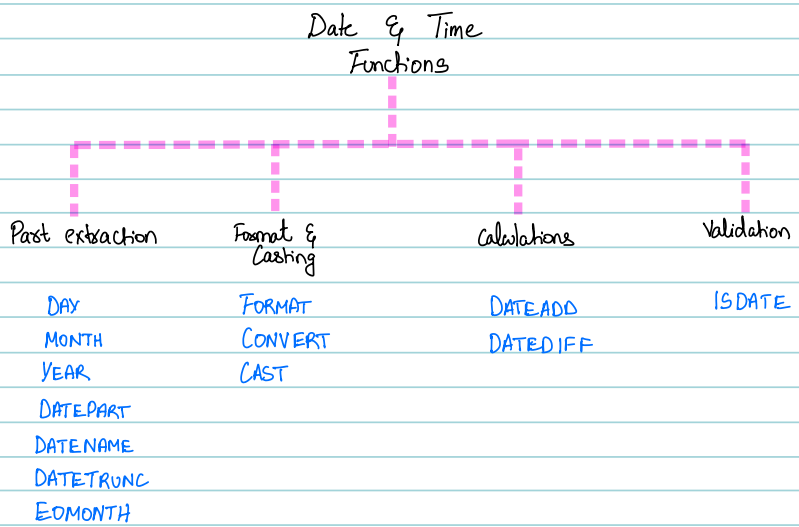
★ Syntax SUBSTRING

SUBSTRING (Value, Start, length)

Let's say you don't know precisely how much length to mention but want until the end of the respective String.

then,

SUBSTRING(TRIM(first_name), 2, LEN(first_name)) AS Whatever



- Standard format

Year - Month - DAY ⇒ 2025 - 08 - 20

Hours - Minutes - Seconds ⇒ 18 : 55 : 45

- These are built-in SQL functions that:
  1) Extract specific parts of a date/time (like just the year or day)
  2) Perform calculations (like adding 7 days to a date)
  3) Convert b/w date/time formats
  4) Compare dates or filters data based on them

🕒 1. `CURRENT_DATE`, `CURRENT_TIME`, `NOW()`, `GETDATE()`

➤ What They Do

They return the system's current **date**, **time**, or **timestamp**.

✅ Use Cases

- Automatically timestamp new records
- Fetch today's orders or system logs
- Calculate durations from "today"

💡 SQL Syntax

```sql
-- Returns current full date and time (most systems)
SELECT NOW();  -- PostgreSQL / MySQL

-- SQL Server
SELECT GETDATE();  -- Outputs: 2025-06-23 14:58:43.290

-- Returns only the date (without time)
SELECT CURRENT_DATE;  -- PostgreSQL / MySQL

-- Returns current time
SELECT CURRENT_TIME;  -- Outputs: 14:58:43
```

📆 2. `DATEPART()` / `EXTRACT()`

➤ What They Do

Extract a specific part of a date: year, month, day, weekday, etc.

✅ Use Cases

- Group sales by month
- Filter events that happened in a specific quarter or hour

💡 SQL Syntax

```sql
-- SQL Server
SELECT DATEPART(YEAR, '2025-06-23');  -- Output: 2025
SELECT DATEPART(MONTH, GETDATE());    -- Output: 6

-- PostgreSQL / Standard SQL
SELECT EXTRACT(DAY FROM DATE '2025-06-23');    -- Output: 23
SELECT EXTRACT(WEEK FROM CURRENT_DATE);       -- Output: Week number
```

📅 3. `YEAR()`, `MONTH()`, `DAY()`

➤ What They Do

Simplified versions of `DATEPART`. Easier to read and widely supported.

✅ Use Cases

- Identify month of birth
- Filter data by year directly

💡 SQL Syntax

```sql
SELECT YEAR('2025-06-23');   -- Output: 2025
SELECT MONTH('2025-06-23');  -- Output: 6
SELECT DAY('2025-06-23');    -- Output: 23
```

🧱 4. `DATEDIFF()`

➤ What It Does

Calculates the difference (in days, months, years, etc.) between two dates.

✅ Use Cases

- Calculate age
- How long since a subscription started
- Time gap between two purchases

💡 SQL Syntax

```sql
-- SQL Server syntax
SELECT DATEDIFF(DAY, '2025-06-01', '2025-06-23');   -- Output: 22

-- PostgreSQL / MySQL
SELECT '2025-06-23'::date - '2025-06-01'::date;     -- Output: 22
```

➕ 5. `DATEADD()`

➤ What It Does

Adds or subtracts a specific time interval to/from a date.

✅ Use Cases

- Forecast a future date (7 days from now)
- Determine expiration dates

💡 SQL Syntax

```sql
-- SQL Server
SELECT DATEADD(DAY, 10, '2025-06-23');   -- Output: 2025-07-03
SELECT DATEADD(MONTH, -1, GETDATE());    -- 1 month ago from today

-- PostgreSQL
SELECT CURRENT_DATE + INTERVAL '10 days';   -- Output: 2025-07-03
```

💊 6. `FORMAT()`

➤ What It Does

Formats a date/time into a **readable string**. Mostly used in reporting or exporting.

✅ Use Cases

- Display date in `MM/DD/YYYY` format
- Include weekday names in reports

💡 SQL Syntax

```sql
-- SQL Server only
SELECT FORMAT(GETDATE(), 'yyyy-MM-dd');          -- Output: 2025-06-23
SELECT FORMAT(GETDATE(), 'dddd, MMM dd yyyy');   -- Output: Monday, Jun 23 2025

-- PostgreSQL (use TO_CHAR)
SELECT TO_CHAR(NOW(), 'YYYY-MM-DD');             -- Output: 2025-06-23
SELECT TO_CHAR(NOW(), 'FMDay, Mon DD YYYY');     -- Output: Monday, Jun 23 2025
```

📘 SUMMARY TABLE

Function	Action	Example Usage
`NOW()` / `GETDATE()`	Get current timestamp	`SELECT GETDATE();`
`CURRENT_DATE`	Just date, no time	`SELECT CURRENT_DATE;`
`DATEPART`, `EXTRACT`	Get part of a date	`DATEPART(MONTH, ...)` or `EXTRACT(MONTH FROM ...)`
`YEAR`, `MONTH`, `DAY`	Shortcuts for date parts	`YEAR(BirthDate)`
`DATEDIFF()`	Difference between 2 dates	`DATEDIFF(DAY, '2025-01-01', '2025-06-23')`
`DATEADD()`	Add time to a date	`DATEADD(DAY, 7, GETDATE())`
`FORMAT()` / `TO_CHAR()`	Pretty formatting	`FORMAT(GETDATE(), 'MMMM yyyy')`

🧠 NULL vs Blank vs Zero in SQL — Beginner-Friendly Deep Dive

🌑 1. What is `NULL`?

➤ Definition

`NULL` represents **"unknown", "missing", or "not applicable"** value in SQL.
It's **not the same** as `0` or `''` (empty string).

✅ When Does `NULL` Occur?

- A column wasn't filled during `INSERT`
- Data is unavailable or optional (e.g., `MiddleName`)
- Result of an expression like `5 / NULL`

💡 SQL Examples

```sql
-- A table where "EndDate" is not yet set
SELECT * FROM Projects WHERE EndDate IS NULL;

-- Count how many records are missing data
SELECT COUNT(*) FROM Employees WHERE PhoneNumber IS NULL;

-- Replace NULLs with a default value
SELECT COALESCE(MiddleName, 'N/A') FROM People;
```

🕳️ 2. What is a **Blank String** (`''`)?

➤ Definition

`''` is a **known value** — it means **"empty"** but **not unknown**.
It's like someone filled out a form but left that field blank on purpose.

✅ When Does It Occur?

- Input form was submitted with an empty string.
- Someone used: `INSERT INTO Students (Nickname) VALUES ('');`

💡 SQL Examples

```sql
-- Check for records where name was intentionally left blank
SELECT * FROM Users WHERE Nickname = '';

-- Update blanks to NULL if you treat them as the same
UPDATE Users SET Nickname = NULL WHERE Nickname = '';
```

🧱 3. What is `0`?

➤ Definition

`0` is a **numeric value**. It means "none" or "zero quantity," not missing.

✅ When Does It Occur?

- A price was intentionally set to 0.
- Quantity of items is 0 in stock.

💡 SQL Examples

```sql
-- Find free products
SELECT * FROM Products WHERE Price = 0;

-- Treat 0 and NULL differently
SELECT * FROM Payments WHERE Amount IS NULL;  -- Missing amount
SELECT * FROM Payments WHERE Amount = 0;      -- Zero payment
```

⚠️ WARNING: `NULL` Doesn't Behave Like You Think

✘ Wrong:

```sql
SELECT * FROM Users WHERE MiddleName = NULL;  -- NO RESULTS!
```

✓ Right:

```sql
SELECT * FROM Users WHERE MiddleName IS NULL;
```

🧠 Because in SQL, `NULL = NULL` is not true!
NULL means "unknown" — so `unknown = unknown` is still unknown!

🔁 Useful Functions for NULL Handling

Function	What It Does	Example
`IS NULL`, `IS NOT NULL`	Checks if a value is or isn't NULL	`WHERE Email IS NOT NULL`
`COALESCE(a, b, ...)`	Returns the **first non-NULL** value	`COALESCE(Nickname, 'Guest')`
`NULLIF(a, b)`	Returns NULL if a = b	`NULLIF(Salary, 0)` → NULL if salary is 0
`ISNULL(a, b)`	SQL Server version of COALESCE	`ISNULL(Email, 'not_provided@example.com')`
`IFNULL(a, b)`	MySQL version of COALESCE	`IFNULL(score, 0)`

🧪 Comparisons Summary Table

Type	Meaning	Use `=`?	Use `IS`?	Can be Aggregated?
`NULL`	Missing / Unknown	❌ No	✅ Yes	❌ No (ignored by COUNT/AVG/etc.)
`''`	Empty String (known empty)	✅ Yes	❌ No	✅ Yes
`0`	Known numeric zero	✅ Yes	❌ No	✅ Yes

🎯 SQL Query Examples

🔹 Find users with missing phone numbers:

sql Copy Edit

```sql
SELECT * FROM Users WHERE PhoneNumber IS NULL;
```

🔹 Replace NULLs with default values:

sql Copy Edit

```sql
SELECT Name, COALESCE(Nickname, 'Guest') FROM Users;
```

🔹 Check for either NULL or blank:

sql Copy Edit

```sql
SELECT * FROM Users
WHERE Email IS NULL OR Email = '';
```

🔹 Count only non-NULL salaries:

sql Copy Edit

```sql
SELECT AVG(Salary) FROM Employees WHERE Salary IS NOT NULL;
```