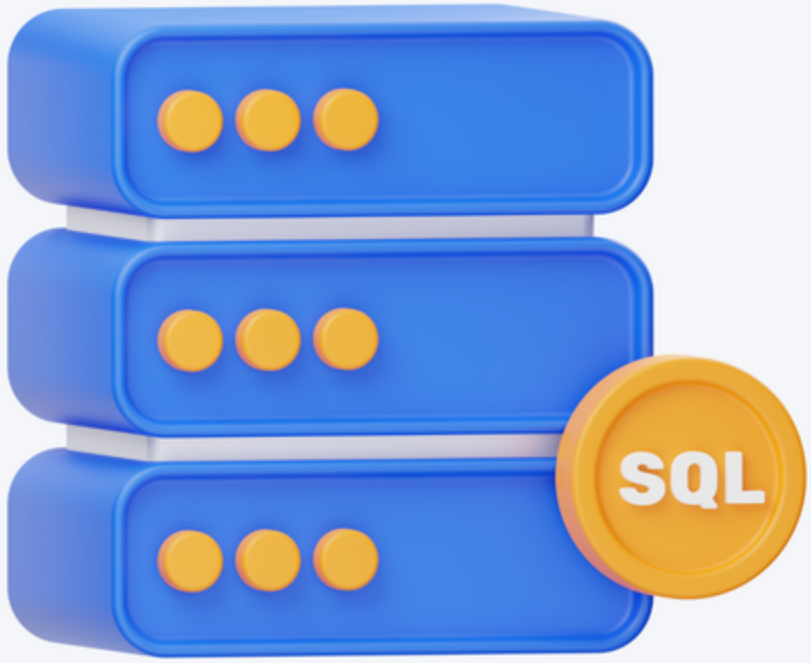
Stop Guessing. Master Your Data's Timeline.



The Core Four

- `CURRENT_DATE` returns today's date from your system.
- `CURRENT_TIME` fetches the current time of day.
- `NOW()` or `GETDATE()` gets the exact current timestamp.
- `DATEADD()` adds or subtracts a time interval to a date.
- `DATEDIFF()` calculates the difference between two dates.
- `DATEPART()` extracts a specific part (like year or month).

Formatting Dates

- Use `TO_CHAR()` (PostgreSQL) or `FORMAT()` (SQL Server) to style dates.
- Convert date formats for reports: 'YYYY-MM-DD', 'MM/DD/YY'.
- Display month names: e.g., 'January' instead of '01'.
- Show day of the week: 'Monday', 'Tuesday', etc.
- `CAST()` or `CONVERT()` changes a date's data type.
- Formatting ensures consistency across your data outputs.

Truncate and Extract

- `DATE_TRUNC()` (PostgreSQL) rounds a date to a precision like 'month'.
- `DATEFROMPARTS()` (SQL Server) builds a date from year, month, day.
- `EXTRACT()` gets a single field like year, quarter, or week.
- Isolate the year from a timestamp for yearly trends.
- Pull the quarter to aggregate financial data.
- Get the day of the week to analyze weekly patterns.

Real-World Use Cases

- Calculate a user's age from their birthdate.
- Filter records from the last 7, 30, or 90 days.
- Group sales data by month or quarter for reporting.
- Schedule future events by adding days to the current date.
- Track how many days between an order and its shipment.
- Analyze peak activity hours by extracting the hour from timestamps.

Pro Tips & Pitfalls

- Always be aware of your database's timezone settings.
- Date functions can vary by system (e.g., MySQL vs. SQL Server).
- Use intervals with `DATEADD` for clean, timezone-aware math.
- Beware of performance costs when formatting dates in WHERE clauses.
- Test your queries across date boundaries (month-end, leap years).
- Comment your date logic so others understand your intent.

FOLLOW FOR MORE

Like this post? Repost!

Share your thoughts in the comments!