SQL Functions

1. Mathematical Functions

- Used to perform arithmetic operations on numerical data.
    - ABS() - Returns absolute value.
    - ROUND() - Rounds a number to a specified decimal places.
    - CEILING() - Rounds up to the nearest integer.
    - FLOOR() - Rounds down to the nearest integer.
    - POWER() - Raises a number to a specified power.
    - SQRT() - Returns the square root of a number.
    - MOD() - Returns the remainder of division.

2. String Functions

- Manipulate and format text data.
    - CONCAT() - Concatenates multiple strings.
    - SUBSTRING() - Extracts a portion of a string.
    - LENGTH() - Returns the length of a string.
    - UPPER() / LOWER() - Converts to uppercase/lowercase.
    - TRIM() - Removes whitespace from the start and end.
    - REPLACE() - Replaces part of a string with another string.

3. Ranking Functions

- Provide ranking within a result set.
    - RANK() - Assigns a unique rank, allowing gaps for ties.
    - DENSE_RANK() - Ranks without gaps in ranking values.
    - ROW_NUMBER() - Assigns a unique number for each row.
    - **Usage**: Often used with OVER() clause for partitioning and ordering.

4. Date and Time Functions

- Work with dates and times to perform calculations or format them.
    - CURRENT_DATE / CURRENT_TIMESTAMP - Returns the current date or timestamp.
    - DATEADD() - Adds a specific interval to a date.
    - DATEDIFF() - Calculates the difference between two dates.
    - YEAR(), MONTH(), DAY() - Extracts parts of a date.
    - FORMAT() - Formats date/time into a specific string pattern.

5. Aggregate Functions

- Perform calculations on multiple rows and return a single result.
    - SUM() - Returns the total sum of a numeric column.
    - AVG() - Returns the average value of a numeric column.
    - COUNT() - Counts the number of rows.
    - MIN() - Finds the minimum value.
    - MAX() - Finds the maximum value.
- **Usage**: Often used with GROUP BY to aggregate data for specific groups.

SQL Order of execution

FROM
- Specifies the tables to retrieve data from.
- Joins tables if multiple tables are specified.

WHERE
- Filters rows based on specified conditions.
- Only rows that meet the condition are passed to the next step.

GROUP BY
- Groups rows sharing a common field value.
- Used in conjunction with aggregate functions (like SUM, COUNT).

HAVING
- Filters groups created by GROUP BY based on conditions.
- Similar to WHERE, but operates on aggregated data.

SELECT
- Specifies the columns to be returned in the result set.
- Includes calculated fields, aliases, and aggregated values.

DISTINCT
- Removes duplicate rows from the result set.

ORDER BY
- Sorts the result set based on specified columns.
- Defaults to ascending order unless DESC is specified.

LIMIT / OFFSET
- Limits the number of rows returned or skips a specified number of rows.
- Useful for pagination or sampling data.

ROLLUP operation in MS SQL:

1. **Purpose of ROLLUP**
   - ROLLUP is used in the GROUP BY clause to add subtotals and a grand total to query results.
   - Helps create hierarchical subtotal rows, showing subtotals for each level and an overall total.
2. **How ROLLUP Works**
   - Calculates results at each grouping level, adding subtotals for each combination.
   - The result set includes:
     - **Regular rows**: Rows grouped by all specified columns.
     - **Subtotal rows**: Rows with NULL values for one or more grouping columns to represent subtotals.

- **Grand total row**: A row with NULL in all grouping columns to show the overall total.

3. **Example of ROLLUP for Subtotals**

   SELECT Region, Product, SUM(Sales) AS TotalSales FROM SalesData GROUP BY ROLLUP (Region, Product);

   - Subtotals for each Region are calculated, showing NULL in the Product column.
   - A grand total row is added, showing NULL in both Region and Product.
4. **Interpreting Results**
   - Rows with NULL values in one or more columns represent subtotal or grand total levels.
   - NULL in a higher-level column means a subtotal for all items at that level.
5. **Ordering Results**
   - Use ORDER BY to control the appearance of subtotals and grand totals.
   - Ordering helps clarify the structure, especially when many levels are grouped.

STORED PROCEDURES

1. **Definition**
   - A stored procedure is a set of SQL statements saved under a name and stored in the database.
   - It allows for reusable code that can be executed multiple times.
2. **Advantages**
   - **Modularity**: Groups complex logic into a single procedure.
   - **Performance**: Precompiled and cached, which improves execution speed.
   - **Security**: Access can be controlled; users can execute without direct table access.
   - **Maintenance**: Easier to manage and update business logic centrally.
3. **Syntax**

```
CREATE PROCEDURE ProcedureName
@Parameter1 DataType, @Parameter2 DataType
AS
BEGIN
   -- SQL statements
END;
```

   - Parameters are optional; they allow passing values into the procedure.
   - BEGIN and END encapsulate the SQL code.
4. **Execution**
   - Stored procedures are executed with the EXEC or EXECUTE command.

   EXEC ProcedureName @Parameter1 = Value1, @Parameter2 = Value2;

5. **Types of Stored Procedures**
   - **User-defined**: Created by users for custom tasks.
   - **System-defined**: Provided by SQL Server for administrative tasks (e.g., sp_help).
6. **Error Handling**
   - Use TRY...CATCH blocks within stored procedures to handle errors gracefully.
7. **Modifying and Dropping**
   - Modify with ALTER PROCEDURE.
   - Delete with DROP PROCEDURE ProcedureName.
8. **Example**

```
CREATE PROCEDURE GetCustomerOrders
@CustomerId INT
AS
BEGIN
   SELECT * FROM Orders WHERE CustomerID = @CustomerId;
END;
```

coalesce - replace missing value with some suitable value

5/11/24

5/11/24.

Database schema

Flat
hierarchical
network
relational
star
snowflake - multiple fact table

Functions

String, date&time, math, aggregate, ranking,
Configuration, metadata, security, cryptographic

Select column, 'Hourly rate' = Round(hourrate ,2) from employee;

Load function

Rank

row_number()

rank()

dense_rank()

ntile()

Select studid, grade, ROW_NUMBER() over (order by studid desc) as rank from result;

Select orderid, cost, RANK() over (order by orderid) as rank from sale;

---

dense_rank() - consecutive ranking.

dense rank 1 1 2 3 3 4 5 5 6.

(rank 1 1 3 4 5 5 7)

ntile is for grouping.

SQL Order of Execution

why order of exec is important

→ accurate result

→ query optimization

→ Efficient resource utilization

→ Reduced query complexity.

FROM (1) → where (2) → groupby (3) → having (4) → select (5) →

order by (6) → limit / offset (7)

FWGHSOL

Calculate Subtotals

Rollup, cube, grouping sets are extensions of groupby

- before 2019 till 2019;

Aggregate function

Stored procedure

Set of stmt perform defined operations.

Create procedure us_cust as
Select custid, name from customers where country = 'USA';