SQL REFRESHER

Step 1: Create a New Database

```
CREATE DATABASE LibraryDB;
GO
```

Refresh the **Databases** folder to see LibraryDB.

Step 2: Create a Table

```
USE LibraryDB;
GO

CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    PublishedYear INT
);
```

Step 3: Insert Data

```
INSERT INTO Books (BookID, Title, Author, PublishedYear)
VALUES
(1, 'The Alchemist', 'Paulo Coelho', 1988),
(2, '1984', 'George Orwell', 1949);
```

Step 4: Retrieve Data

```
SELECT * FROM Books;
```

Step 5: Update and Delete

```
-- Update
UPDATE Books
```

```
SET Author = 'George Orwell Jr.'
WHERE BookID = 2;

-- Delete
DELETE FROM Books
WHERE BookID = 1;
```

Step 6: Use WHERE, LIKE, AND, OR

```
SELECT * FROM Books
WHERE Author LIKE '%Orwell%' AND PublishedYear > 1900;
```

Step 7: Aggregate & Grouping

```
SELECT Author, COUNT(*) AS TotalBooks
FROM Books
GROUP BY Author;
```

Step 8: Joins (Using 2 Tables)

```
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    Name VARCHAR(100)
);

CREATE TABLE BorrowedBooks (
    BorrowID INT PRIMARY KEY,
    BookID INT,
    MemberID INT,
    BorrowDate DATE,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);
```

-- Join Example

SELECT m.Name, b.Title, br.BorrowDate

FROM BorrowedBooks br

JOIN Books b ON br.BookID = b.BookID

JOIN Members m ON br.MemberID = m.MemberID;

Step 9: Constraints

- **PRIMARY KEY** – Unique identifier
- **FOREIGN KEY** – Referencing another table
- **NOT NULL**, **UNIQUE**, **CHECK**, **DEFAULT**

Example:

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY,

Name VARCHAR(100) UNIQUE NOT NULL

);

Step 10: Date & String Functions

-- String Function

SELECT UPPER(Author), LEN(Title) FROM Books;

-- Date Function

SELECT GETDATE() AS CurrentDate;

Step 11: Subqueries and Set Operators

-- Subquery

SELECT Title FROM Books

WHERE PublishedYear = (SELECT MAX(PublishedYear) FROM Books);

-- UNION Example

```
SELECT Title FROM Books
UNION
SELECT Name FROM Members;
```

Step 12: Dropping Objects

```
DROP TABLE BorrowedBooks;
DROP DATABASE LibraryDB;
```

Notes:

Task	Command
View all databases	SELECT name FROM sys.databases;
View all tables	SELECT * FROM sys.tables;
View DB files	EXEC sp_helpfile;
Rename DB/Table	Use GUI or ALTER DATABASE/ALTER TABLE
Backup/Restore	Use GUI or BACKUP DATABASE, RESTORE DATABASE

String Functions in MS SQL Server (with Examples)

- **1. LEN()**

Returns the number of characters in a string (excluding trailing spaces).

```
SELECT LEN('Hello SQL'); -- Output: 9
```

- **2. DATALENGTH()**

Returns the number of bytes used to represent any expression (including trailing spaces).

```
SELECT DATALENGTH('Hello SQL'); -- Output: 9 (for VARCHAR)
```

- **3. LEFT(string, number)**

Returns the left part of a string with the specified number of characters.

```
SELECT LEFT('Database', 4); -- Output: 'Data'
```

- **4. RIGHT(string, number)**

Returns the right part of a string with the specified number of characters.

SELECT RIGHT('Database', 4); -- Output: 'base'

- **5. SUBSTRING(string, start, length)**

Extracts part of a string starting from a specific position.

SELECT SUBSTRING('SQL Server', 5, 6); -- Output: 'Server'

- **6. CHARINDEX(substring, string)**

Returns the starting position of a substring within a string.

SELECT CHARINDEX('S', 'SQL Server'); -- Output: 1

SELECT CHARINDEX('Server', 'SQL Server'); -- Output: 5

- **7. PATINDEX('%pattern%', string)**

Returns the starting position of a pattern using wildcards (%).

SELECT PATINDEX('%ver%', 'SQL Server'); -- Output: 6

- **8. REPLACE(string, find, replace_with)**

Replaces all occurrences of a substring.

SELECT REPLACE('Hello World', 'World', 'SQL'); -- Output: 'Hello SQL'

- **9. REPLICATE(string, number)**

Repeats a string a specified number of times.

SELECT REPLICATE('*', 5); -- Output: '*****'

- **10. SPACE(number)**

Inserts specified number of spaces.

SELECT 'Hello' + SPACE(3) + 'SQL'; -- Output: 'Hello   SQL'

- **11. LTRIM(string)**

Removes leading spaces.

SELECT LTRIM('   Hello'); -- Output: 'Hello'

- **12. RTRIM(string)**

Removes trailing spaces.

SELECT RTRIM('Hello   '); -- Output: 'Hello'

- **13. LOWER(string)**

Converts all characters to lowercase.

SELECT LOWER('HeLLo SQL'); -- Output: 'hello sql'

- **14. UPPER(string)**

Converts all characters to uppercase.

SELECT UPPER('HeLLo SQL'); -- Output: 'HELLO SQL'

- **15. CONCAT(string1, string2, ...)**

Concatenates two or more strings.

SELECT CONCAT('SQL', ' ', 'Server'); -- Output: 'SQL Server'

- **16. FORMAT(value, format)**

Formats a string or number or date (like .NET style formatting).

SELECT FORMAT(1234.56, 'C', 'en-US'); -- Output: '$1,234.56'

- **17. QUOTENAME(string)**

Returns a Unicode string with delimiters added (useful for dynamic SQL).

SELECT QUOTENAME('tableName'); -- Output: [tableName]

- **18. STRING_AGG(expression, separator)**

Concatenates values from multiple rows into a single string (SQL Server 2017+).

```
SELECT STRING_AGG(Name, ', ') AS Names
FROM Employees;
-- Output: 'Alice, Bob, Charlie'
```

Combining Multiple String Functions

```
SELECT
    UPPER(LEFT('database', 1)) + LOWER(SUBSTRING('database', 2, LEN('database'))) AS Capitalized;
-- Output: 'Database'
```

DATE FUNCTIONS in SQL Server

These functions allow manipulation and formatting of **date and time values**.

Function	Description	Example
GETDATE()	Returns current date and time	SELECT GETDATE();
SYSDATETIME()	Returns current system datetime (more precision)	SELECT SYSDATETIME();
CURRENT_TIMESTAMP	Same as GETDATE()	SELECT CURRENT_TIMESTAMP;
GETUTCDATE()	Returns current UTC datetime	SELECT GETUTCDATE();
DATEPART(part, date)	Extracts part (year, month, etc.) from date	SELECT DATEPART(YEAR, GETDATE());
DATENAME(part, date)	Returns part of date as string	SELECT DATENAME(MONTH, GETDATE());
DAY(date)	Returns day of the month	SELECT DAY('2025-07-17');

Function	Description	Example
MONTH(date)	Returns month number	SELECT MONTH('2025-07-17');
YEAR(date)	Returns year	SELECT YEAR('2025-07-17');
DATEADD(part, number, date)	Adds interval to date	SELECT DATEADD(DAY, 5, GETDATE());
DATEDIFF(part, start, end)	Returns difference between two dates	SELECT DATEDIFF(DAY, '2025-07-01', '2025-07-17');
EOMONTH(date)	Returns last day of month	SELECT EOMONTH('2025-07-17');
SWITCHOFFSET()	Changes time zone offset	SELECT SWITCHOFFSET(SYSDATETIMEOFFSET(), '-05:00');
FORMAT(date, format)	Formats date like .NET	SELECT FORMAT(GETDATE(), 'dd-MMM-yyyy');

MATHEMATICAL FUNCTIONS

Used for arithmetic operations, rounding, etc.

Function	Description	Example
ABS(x)	Returns absolute value	SELECT ABS(-25); → 25
CEILING(x)	Rounds up to next integer	SELECT CEILING(4.3); → 5
FLOOR(x)	Rounds down to previous integer	SELECT FLOOR(4.8); → 4
ROUND(x, d)	Rounds to d decimal places	SELECT ROUND(123.4567, 2); → 123.46
POWER(x, y)	x raised to power y	SELECT POWER(2, 3); → 8
SQRT(x)	Square root	SELECT SQRT(16); → 4
SIGN(x)	-1 for negative, 0 for zero, 1 for positive	SELECT SIGN(-25); → -1
EXP(x)	e raised to the power of x	SELECT EXP(1); → 2.71828
LOG(x)	Natural log (base e)	SELECT LOG(10);

Function	Description	Example
LOG10(x)	Log base 10	SELECT LOG10(100); → 2
PI()	Returns value of π	SELECT PI();
RAND()	Returns a random float between 0 and 1	SELECT RAND();
DEGREES(x)	Converts radians to degrees	SELECT DEGREES(PI()); → 180
RADIANS(x)	Converts degrees to radians	SELECT RADIANS(180); → 3.14
SQUARE(x)	x squared	SELECT SQUARE(5); → 25

SYSTEM FUNCTIONS

Provide information about **system state**, **metadata**, **user**, etc.

Function	Description	Example
@@VERSION	SQL Server version	SELECT @@VERSION;
@@SPID	Current session ID	SELECT @@SPID;
@@TRANCOUNT	Number of active transactions	SELECT @@TRANCOUNT;
@@IDENTITY	Last inserted identity value (any table)	SELECT @@IDENTITY;
SCOPE_IDENTITY()	Last identity in current scope	SELECT SCOPE_IDENTITY();
IDENT_CURRENT('table')	Last identity in a specific table	SELECT IDENT_CURRENT('Employees');
ISNULL(expr, replacement)	Replaces NULL with replacement	SELECT ISNULL(NULL, 'Default');
COALESCE(expr1, expr2,...)	Returns first non-null value	SELECT COALESCE(NULL, NULL, 'OK');

Function	Description	Example
NEWID()	Generates a new uniqueidentifier (GUID)	SELECT NEWID();
HOST_NAME()	Returns machine name	SELECT HOST_NAME();
SYSTEM_USER	Current user login	SELECT SYSTEM_USER;
USER_NAME()	Database user name	SELECT USER_NAME();
DB_NAME()	Current database name	SELECT DB_NAME();
OBJECT_NAME(id)	Object name for object ID	SELECT OBJECT_NAME(OBJECT_ID('Employees'));
ERROR_MESSAGE()	Returns error message from TRY...CATCH	Used in error handling block

---

Combine Functions

-- Find days until end of the current month

SELECT DATEDIFF(DAY, GETDATE(), EOMONTH(GETDATE())) AS DaysLeft;

-- Format today's date with time

SELECT FORMAT(GETDATE(), 'dd-MM-yyyy hh:mm tt') AS FormattedDate;

Implementing Data Integrity in MS SQL

- **Constraints:**

```
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Age INT CHECK (Age >= 18),
```

```
    Email VARCHAR(100) UNIQUE,
    DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID)
);
```

Using Functions to Customize the Result Set

```
SELECT UPPER(Name) AS UpperName,
       GETDATE() AS CurrentDate,
       ROUND(Salary, 2) AS RoundedSalary
FROM Employees;
```

Using String Functions

```
SELECT
  Name,
  UPPER(Name) AS UpperCase,
  LEN(Name) AS NameLength,
  LEFT(Name, 3) AS First3Letters,
  REPLACE(Email, '.com', '.org') AS UpdatedEmail
FROM Employees;
```

Using Date Functions

```
SELECT
  GETDATE() AS CurrentDateTime,
  YEAR(HireDate) AS HireYear,
  DATEDIFF(YEAR, HireDate, GETDATE()) AS YearsWorked
FROM Employees;
```

Using Mathematical Functions

```
SELECT
  Salary,
  ROUND(Salary, 0) AS RoundedSalary,
```

```
  CEILING(Salary) AS CeilValue,
  FLOOR(Salary) AS FloorValue,
  POWER(Salary, 2) AS SalarySquared
FROM Employees;
```

Using System Functions

```
SELECT
  SYSTEM_USER AS LoggedInUser,
  HOST_NAME() AS Host,
  DB_NAME() AS CurrentDatabase,
  NEWID() AS RandomGUID;
```

Summarizing and Grouping Data

- **Aggregate Functions**

```
SELECT
  COUNT(*) AS TotalEmployees,
  AVG(Salary) AS AvgSalary,
  MAX(Salary) AS MaxSalary,
  MIN(Salary) AS MinSalary,
  SUM(Salary) AS TotalSalary
FROM Employees;
```

- **Grouping Data**

```
SELECT DepartmentID, COUNT(*) AS Total
FROM Employees
GROUP BY DepartmentID;
```

Hands-on Exercises

- **1. Filtering Data using SQL Queries**

```
SELECT * FROM Employees
WHERE Age > 30 AND DepartmentID = 2;
```

🔸 2. Total Aggregations

```
SELECT
  COUNT(*) AS TotalEmployees,
  SUM(Salary) AS TotalSalary
FROM Employees;
```

🔸 3. Group By Aggregations

```
SELECT DepartmentID, AVG(Salary) AS AvgDeptSalary
FROM Employees
GROUP BY DepartmentID;
```

🔸 4. Order of Execution of SQL Queries

Correct Order (Conceptual Execution):

1. FROM
2. JOIN
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. ORDER BY

Example:

```
SELECT DepartmentID, AVG(Salary) AS AvgSalary
FROM Employees
WHERE Age > 25
GROUP BY DepartmentID
HAVING AVG(Salary) > 40000
ORDER BY AvgSalary DESC;
```

🔸 5. Rules and Restrictions to Group and Filter

- Columns in SELECT must either be:
    - Aggregated: AVG(Salary)
    - Or in GROUP BY
- HAVING is used **after** GROUP BY to filter aggregates
- WHERE is used **before** GROUP BY to filter rows

🔸 6. Filter Data using Group By and Having

```
SELECT DepartmentID, COUNT(*) AS EmpCount
FROM Employees
GROUP BY DepartmentID
HAVING COUNT(*) > 3;
```