SQL FUNDAMENTALS

Get Started

SQL

Structured Query Language, a standard language for querying and managing data in relational database management systems (RDBMS).

```
SELECT * FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

T-SQL

Transact-SQL, an extension of SQL used by Microsoft SQL Server, adding procedural programming and local variable features.

```
DECLARE @EmployeeID INT = 1;
SELECT * FROM Employees
WHERE EmployeeID = @EmployeeID;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

PREDICATE LOGIC

A branch of logic that deals with predicates, which are statements that can be true or false. It is used in SQL for filtering data.

```
SELECT * FROM Employees
WHERE Salary > 50000;
```


Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

RELATIONAL MODEL

A model for managing and manipulating data based on set theory and predicate logic. It represents data as relations (tables).

Tables in SQL Server follow the relational model, where each table represents a set of data.

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

RDBMS

Relational Database Management System, a software system that uses relational models to manage databases.

Microsoft SQL Server is an example of an RDBMS.

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

TABLES

The basic unit of data storage in SQL Server, representing a collection of related data.

```
CREATE TABLE Employees
(EmployeeID INT,
Name NVARCHAR(100),
Salary MONEY);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SCHEMAS

A way to logically group database objects such as tables, views, and procedures.

```
CREATE SCHEMA Sales;

CREATE TABLE Sales.Orders
(OrderID INT, OrderDate DATE);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DATA INTEGRITY

Ensuring the accuracy and consistency of data within a database. Enforced using constraints like primary keys and foreign keys.

```
ALTER TABLE Employees
ADD CONSTRAINT PK_EmployeeID
PRIMARY KEY (EmployeeID);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SELECT STATEMENT

A SQL command used to retrieve data from a database.

```
SELECT Name,
Salary
FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FROM CLAUSE

Specifies the table from which to retrieve the data.

```
SELECT * FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

WHERE CLAUSE

Filters the records based on a condition.

```
SELECT * FROM Employees
WHERE Salary > 50000;
```


Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

GROUP BY CLAUSE

Groups rows that have the same values into summary rows.

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

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

HAVING CLAUSE

Used to filter groups based on a condition.

```
SELECT DepartmentID, AVG(Salary)
FROM Employees
GROUP BY DepartmentID
HAVING AVG(Salary) > 50000;
```


Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ORDER BY CLAUSE

Sorts the result set in either ascending or descending order.

```
SELECT *
FROM Employees
ORDER BY Salary DESC;
```


Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

TOP CLAUSE

Limits the number of rows returned in the result set.

```
SELECT TOP 10 *
FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

OFFSET-FETCH CLAUSE

Provides a way to implement paging in a query.

```
SELECT *
FROM Employees
ORDER BY EmployeeID
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

WINDOW FUNCTIONS

Perform calculations across a set of table rows related to the current row.

```
SELECT EmployeeID, Salary,
AVG(Salary) OVER (PARTITION BY
DepartmentID) AS AvgSalary
FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CASE EXPRESSIONS

Evaluates a list of conditions and returns one of multiple possible result expressions.

```
SELECT Name,
CASE WHEN Salary > 50000
THEN 'High' ELSE 'Low'
END AS SalaryCategory
FROM Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

NULLS

A marker used in SQL to indicate that a data value does not exist in the database.

```
SELECT *
FROM Employees
WHERE MiddleName IS NULL;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ALL-AT-ONCE OPERATIONS

SQL operations are set-based and act on all rows in a result set simultaneously.

```
UPDATE Employees
SET Salary = Salary * 1.1;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CHARACTER DATA

Data type used to store text strings.

```
CREATE TABLE Products
(ProductID INT,
ProductName NVARCHAR(100));
```

Feel free to follow me
for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DATE AND TIME DATA

Data types used to store date and time values.

```
CREATE TABLE Events
(EventID INT,
EventDate DATE,
EventTime TIME);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FILTERING DATE RANGES

Querying data within a specific date range.

```
SELECT *
FROM Orders
WHERE
OrderDate BETWEEN '2023-01-01'
AND '2023-12-31';
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CROSS JOINS

Combines each row from the first table with each row from the second table.

```
SELECT *
FROM Employees
CROSS JOIN Departments;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

INNER JOINS

Combines rows from two tables based on a related column.

```
SELECT *
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID =
Departments.DepartmentID;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

OUTER JOINS

Returns all rows from one table and the matched rows from the second table.

```
SELECT * FROM
Employees
LEFT JOIN Departments
ON
Employees.DepartmentID =
Departments.DepartmentID;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

COMPOSITE JOINS

Joins based on multiple columns.

```
SELECT *
FROM Orders
INNER JOIN OrderDetails
ON Orders.OrderID =
OrderDetails.OrderID AND
Orders.ProductID =
OrderDetails.ProductID;
```

Feel free to follow me
for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

NON-EQUI JOINS

Joins based on conditions other than equality.

```
SELECT *
FROM Employees
INNER JOIN Salaries
ON Employees.Salary >
Salaries.MinSalary;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SUBQUERIES

A query nested inside another query.

```
SELECT *
FROM Employees
WHERE
DepartmentID = (SELECT
DepartmentID FROM
Departments WHERE
DepartmentName = 'Sales');
```

Feel free to follow me
for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DERIVED TABLES

Subqueries used in the FROM clause to create a temporary table.

```
SELECT
AVG(Salary)
FROM (SELECT Salary
FROM Employees
WHERE DepartmentID = 1)
AS DeptSalaries;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

COMMON TABLE EXPRESSIONS (CTES)

Temporary result sets that can be referenced within a SELECT, INSERT, UPDATE, or DELETE statement.

```
WITH EmployeeCTE
AS (SELECT *
FROM Employees
WHERE Salary > 50000)
SELECT * FROM EmployeeCTE;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

VIEWS

Virtual tables based on the result set of a SELECT statement.

```
CREATE VIEW
HighSalaryEmployees
AS
SELECT * FROM Employees
WHERE Salary > 50000;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

INLINE TABLE-VALUED FUNCTIONS

Functions that return a table data type and can be used like a table in a query.

```
CREATE FUNCTION
GetEmployeesByDepartment(@DeptID INT)
RETURNS TABLE AS RETURN
(SELECT * FROM Employees
WHERE DepartmentID = @DeptID);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

APPLY OPERATOR

Used with table-valued functions to join a table to the result of a table-valued function.

```
SELECT * FROM Departments
CROSS APPLY
GetEmployeesByDepartment(Departments.DepartmentID);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

UNION OPERATOR

Combines the result sets of two or more queries into a single result set.

```
SELECT Name
FROM Employees
UNION
SELECT Name
FROM Customers;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

INTERSECT OPERATOR

Returns the common rows between two result sets.

```
SELECT Name
FROM Employees
INTERSECT
SELECT Name
FROM Customers;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

EXCEPT OPERATOR

Returns rows from the first result set that are not in the second result set.

```
SELECT Name
FROM Employees
EXCEPT
SELECT Name
FROM Customers;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

RANKING FUNCTIONS

Window functions that assign a rank to each row within a partition of a result set.

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

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

AGGREGATE FUNCTIONS

Perform calculations on a set of values and return a single value.

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

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

PIVOTING DATA

Transforming rows into columns.

```
SELECT *
FROM (SELECT Year, SalesAmount
FROM Sales)
PIVOT
(SUM(SalesAmount) FOR Year IN
([2018], [2019], [2020])) AS
SalesPivot;
```

Feel free to follow me
for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

UNPIVOTING DATA

Transforming columns into rows.

```
SELECT Year, SalesAmount
FROM Sales
UNPIVOT
(SalesAmount FOR Year IN ([2018], [2019], [2020])) AS SalesUnpivot;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

GROUPING SETS

A powerful extension of the GROUP BY clause that allows you to create multiple groupings in the same query.

```
SELECT DepartmentID,
GroupingSet,
SUM(Salary)
FROM Employees
GROUP BY GROUPING SETS
((DepartmentID), (GroupingSet));
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

INSERT STATEMENT

Adds new rows to a table.

```
INSERT INTO Employees (EmployeeID,
Name, Salary)
VALUES (1, 'Alice', 50000);
```


Feel free to follow me
for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

UPDATE STATEMENT

Modifies existing rows in a table.

```
UPDATE Employees
SET Salary = 60000
WHERE EmployeeID = 1;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DELETE STATEMENT

Removes rows from a table.

```
DELETE
FROM Employees
WHERE EmployeeID = 1;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

MERGE STATEMENT

Performs insert, update, or delete operations on a target table based on the results of a join with a source table.

```
MERGE INTO Employees AS Target
USING NewEmployees AS Source
ON Target.EmployeeID = Source.EmployeeID
WHEN MATCHED
THEN UPDATE SET Target.Salary =
Source.Salary
WHEN NOT MATCHED
THEN INSERT (EmployeeID, Name, Salary)
VALUES (Source.EmployeeID, Source.Name,
Source.Salary);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SYSTEM-VERSIONED TEMPORAL TABLES

Tables that provide information about data changes over time.

```
CREATE TABLE Employees
(EmployeeID INT,
Name NVARCHAR(100),
Salary MONEY,
SysStartTime DATETIME2 GENERATED
ALWAYS AS ROW START,
SysEndTime DATETIME2 GENERATED
ALWAYS AS ROW END, PERIOD FOR
SYSTEM_TIME (SysStartTime,
SysEndTime))
WITH (SYSTEM_VERSIONING = ON);
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

TRANSACTIONS

A sequence of one or more SQL operations treated as a single unit of work.

```
BEGIN TRANSACTION;
UPDATE Employees
SET Salary = Salary + 1000
WHERE EmployeeID = 1;
COMMIT TRANSACTION;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

LOCKS

Mechanisms to prevent concurrent access to data in a database, ensuring data integrity.

SQL Server automatically handles locking, but you can also manage locks explicitly using hints.

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ISOLATION LEVELS

Define the degree to which the operations in one transaction are isolated from those in other transactions.

```
SET TRANSACTION ISOLATION LEVEL
READ COMMITTED;
```


Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

VARIABLES

Objects that hold a single data value of a specific type and can be used in T-SQL scripts.

```
DECLARE @EmployeeID INT = 1;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

BATCHES

Groups of one or more T-SQL statements sent to SQL Server as a single unit.

```
BEGIN TRANSACTION;
UPDATE Employees
SET Salary = Salary + 1000
WHERE EmployeeID = 1;
COMMIT TRANSACTION;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

FLOW ELEMENTS

Control-of-flow language elements that manage the execution of T-SQL statements.

```
IF @Salary > 50000
BEGIN
PRINT 'High salary';
END
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CURSORS

Database objects used to retrieve, manipulate, and navigate through the result set of a query row by row.

```
DECLARE EmployeeCursor
CURSOR FOR
SELECT EmployeeID FROM
Employees;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

TEMPORARY TABLES

Tables created temporarily within a session or transaction, which are automatically dropped when the session ends.

```
CREATE TABLE #TempEmployees
(EmployeeID INT,
Name NVARCHAR(100));
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DYNAMIC SQL

SQL statements constructed and executed at runtime.

```
EXEC sp_executesql N'SELECT *
FROM Employees
WHERE Salary > @Salary', N'@Salary
MONEY', @Salary = 50000;
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

STORED PROCEDURES

Precompiled collections of T-SQL statements that can be executed as a single unit.

```
CREATE PROCEDURE GetEmployees
@DeptID INT
AS
BEGIN
SELECT * FROM Employees
WHERE DepartmentID = @DeptID;
END
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

TRIGGERS

Special types of stored procedures that automatically execute when certain events occur in the database.

```
CREATE TRIGGER trgAfterInsert
ON
Employees AFTER INSERT
AS
BEGIN
PRINT 'A new employee record was
inserted.';
END
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ERROR HANDLING

Techniques to handle errors that occur during the execution of T-SQL statements.

```
BEGIN TRY
BEGIN TRANSACTION;
UPDATE Employees
SET Salary = Salary + 1000
WHERE EmployeeID = 1;
COMMIT TRANSACTION;
END TRY
BEGIN CATCH ROLLBACK TRANSACTION;
PRINT ERROR_MESSAGE();
END CATCH
```

Shwetank Singh
GritSetGrow - GSGLearn.com

THANK YOU!