Data Engineering 101

SQL Basics

All the concepts to get started

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SELECT

USED TO SELECT DATA FROM A DATABASE.

```
SELECT *
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

FROM

SPECIFIES THE TABLE TO SELECT OR DELETE DATA FROM.

```
SELECT Name
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

WHERE

FILTERS RECORDS.

```
SELECT *
FROM Employees
WHERE Age > 30;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

INSERT

INSERTS NEW DATA INTO A TABLE.

```
INSERT INTO Employees (Name, Age)
VALUES ('John', 28);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

UPDATE

MODIFIES EXISTING DATA IN A TABLE.

```
UPDATE Employees
SET Age = 30
WHERE Name = 'John';
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DELETE

DELETES DATA FROM A TABLE.

```
DELETE FROM Employees
WHERE Name = 'John';
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CREATE TABLE

CREATES A NEW TABLE.

```
CREATE TABLE
Employees (
ID int,
Name varchar(255)
);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DROP TABLE

DELETES A TABLE.

```
DROP TABLE Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ALTER TABLE

MODIFIES AN EXISTING TABLE.

```
ALTER TABLE Employees
ADD Salary int;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

INNER JOIN

RETURNS RECORDS WITH MATCHING VALUES IN BOTH TABLES.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LEFT JOIN

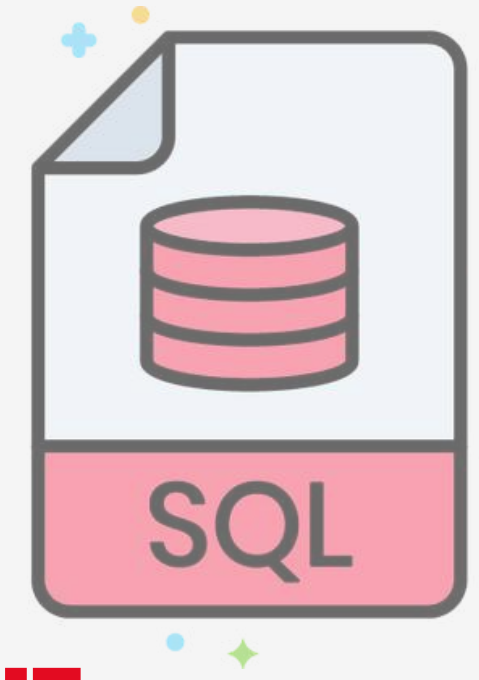
RETURNS ALL RECORDS FROM THE LEFT TABLE, AND MATCHED RECORDS FROM THE RIGHT TABLE.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

RIGHT JOIN



RETURNS ALL RECORDS FROM THE RIGHT TABLE, AND MATCHED RECORDS FROM THE LEFT TABLE.

```
SELECT *
FROM Employees
RIGHT JOIN Departments
ON Employees.DeptID =
Departments.ID;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

FULL JOIN

RETURNS ALL RECORDS WHEN THERE IS A MATCH IN EITHER LEFT OR RIGHT TABLE.

```
SELECT *
FROM Employees
FULL JOIN Departments
ON Employees.DeptID =
Departments.ID;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CROSS JOIN

RETURNS THE CARTESIAN PRODUCT OF THE TWO TABLES.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

GROUP BY

GROUPS ROWS THAT HAVE THE SAME VALUES INTO SUMMARY ROWS.

```
SELECT COUNT(*), Department
FROM Employees
GROUP BY Department;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

HAVING

FILTERS RECORDS THAT WORK ON SUMMARIZED GROUP BY RESULTS.

```
SELECT COUNT(*),
Department
FROM Employees
GROUP BY Department
HAVING COUNT(*) > 5;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ORDER BY

SORTS THE RESULT SET IN ASCENDING OR DESCENDING ORDER.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DISTINCT

SELECTS ONLY DISTINCT (DIFFERENT) VALUES.

```
SELECT DISTINCT Department
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LIMIT

SPECIFIES THE NUMBER OF RECORDS TO RETURN.

```
SELECT *
FROM Employees
LIMIT 10;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

OFFSET

SPECIFIES THE OFFSET OF THE FIRST ROW TO RETURN.

```
SELECT *
FROM Employees
LIMIT 5 OFFSET 10;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

UNION

COMBINES THE RESULT SET OF TWO OR MORE SELECT STATEMENTS.

```
SELECT City
FROM Customers
UNION
SELECT City
FROM Suppliers;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

UNION ALL

COMBINES THE RESULT SET OF TWO OR MORE SELECT STATEMENTS, INCLUDING DUPLICATES.

```
SELECT City
FROM Customers
UNION ALL
SELECT City
FROM Suppliers;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

INTERSECT

RETURNS THE INTERSECTION OF TWO OR MORE SELECT STATEMENTS.

```
SELECT City
FROM Customers
INTERSECT
SELECT City
FROM Suppliers;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

EXCEPT

RETURNS THE DIFFERENCE BETWEEN TWO SELECT STATEMENTS.

```
SELECT City
FROM Customers
EXCEPT
SELECT City
FROM Suppliers;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

IN

CHECKS FOR VALUES WITHIN A SET.

```
SELECT *
FROM Employees
WHERE Department IN ('HR', 'Finance');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

BETWEEN

SELECTS VALUES WITHIN A GIVEN RANGE.

```
SELECT *
FROM Employees
WHERE Age BETWEEN 25 AND 30;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LIKE

SEARCHES FOR A SPECIFIED PATTERN IN A COLUMN.

```
SELECT *
FROM Employees
WHERE Name LIKE 'J%';
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

IS NULL

TESTS FOR EMPTY (NULL) VALUES.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

IS NOT NULL

TESTS FOR NON-EMPTY (NOT NULL) VALUES.

```
SELECT *
FROM Employees
WHERE Age IS NOT NULL;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CASE

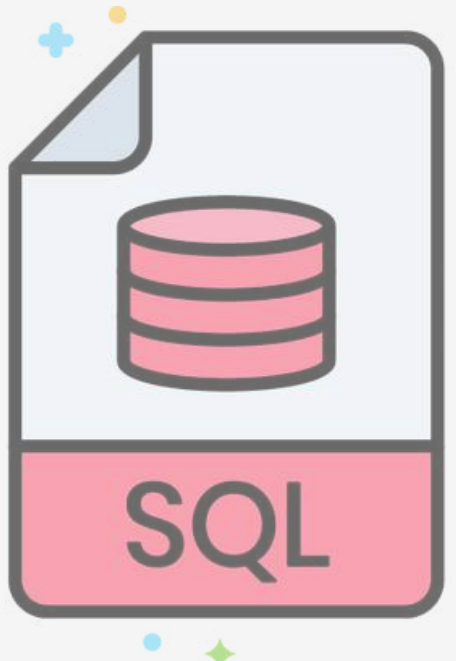
RETURNS VALUE BASED ON A CONDITION.

```
SELECT Name, Age,
CASE WHEN Age > 30 THEN 'Senior'
ELSE 'Junior'
END
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

COALESCE



RETURNS THE FIRST NON-NULL VALUE IN A LIST.

```
SELECT
COALESCE(Address, 'No Address')
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

NULLIF

RETURNS NULL IF TWO EXPRESSIONS ARE EQUAL.

```
SELECT NULLIF(Salary, 0)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CAST

CONVERTS A DATA TYPE INTO ANOTHER DATA TYPE.

```
SELECT CAST(Age AS varchar)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CONVERT

CONVERTS A DATA TYPE INTO ANOTHER DATA TYPE WITH STYLE OPTIONS.

```
SELECT CONVERT(varchar, Age, 1)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SUBSTRING

EXTRACTS CHARACTERS FROM A STRING.

```
SELECT SUBSTRING(Name, 1, 2)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LENGTH

RETURNS THE LENGTH OF A STRING.

```
SELECT LENGTH(Name)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

TRIM

REMOVES SPACES OR SPECIFIED CHARACTERS FROM BOTH ENDS OF A STRING.

```
SELECT TRIM(Name)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

UPPER

CONVERTS A STRING TO UPPERCASE.

```
SELECT UPPER(Name)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LOWER

CONVERTS A STRING TO LOWERCASE.

```
SELECT LOWER(Name)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

REPLACE

REPLACES OCCURRENCES OF A SPECIFIED STRING.

```
SELECT REPLACE(Name, 'John', 'Jon')
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CHARINDEX

RETURNS THE POSITION OF A SUBSTRING IN A STRING.

```
SELECT CHARINDEX('a', Name)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ROUND

ROUNDS A NUMBER TO A SPECIFIED NUMBER OF DECIMAL PLACES.

```
SELECT ROUND(Salary, 2)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

AVG

RETURNS THE AVERAGE VALUE OF A NUMERIC COLUMN.

```
SELECT AVG(Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

COUNT

RETURNS THE NUMBER OF ROWS THAT MATCHES A SPECIFIED CRITERION.

```
SELECT COUNT(*)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SUM

RETURNS THE TOTAL SUM OF A NUMERIC COLUMN.

```
SELECT SUM(Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

MAX

RETURNS THE MAXIMUM VALUE IN A SET.

```
SELECT MAX(Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

MIN

RETURNS THE MINIMUM VALUE IN A SET.

```
SELECT MIN(Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

IFNULL

RETURNS A SPECIFIED VALUE IF THE EXPRESSION IS NULL.

```
SELECT IFNULL(Salary, 0)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CONCAT

CONCATENATES TWO OR MORE STRINGS.

```
SELECT CONCAT(FirstName, ' ',
LastName)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

IF

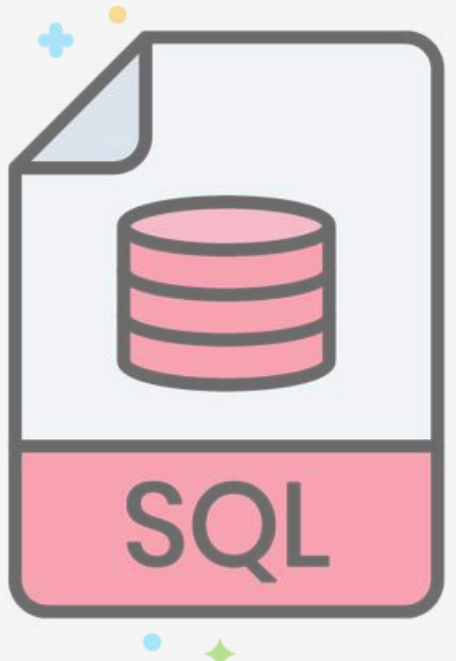
RETURNS A VALUE BASED ON A CONDITION.

```
SELECT Name,
IF(Age > 30, 'Senior', 'Junior')
 FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

EXISTS



CHECKS FOR THE EXISTENCE OF ANY RECORD IN A SUBQUERY.

```
SELECT *
FROM Employees
WHERE
EXISTS (SELECT 1 FROM Departments
WHERE Employees.DeptID =
Departments.ID);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ALL

COMPARES A VALUE TO ALL VALUES IN ANOTHER VALUE SET.

```
SELECT *
FROM Employees
WHERE Salary > ALL (SELECT Salary
FROM Employees
WHERE Department = 'HR');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ANY

COMPARES A VALUE TO ANY VALUE IN ANOTHER VALUE SET.

```
SELECT *
FROM Employees
WHERE Salary > ANY (SELECT Salary
FROM Employees
WHERE Department = 'HR');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SOME

SYNONYM FOR ANY.

```
SELECT *
FROM Employees
WHERE Salary > SOME (SELECT Salary
FROM Employees
WHERE Department = 'HR');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SUBQUERY

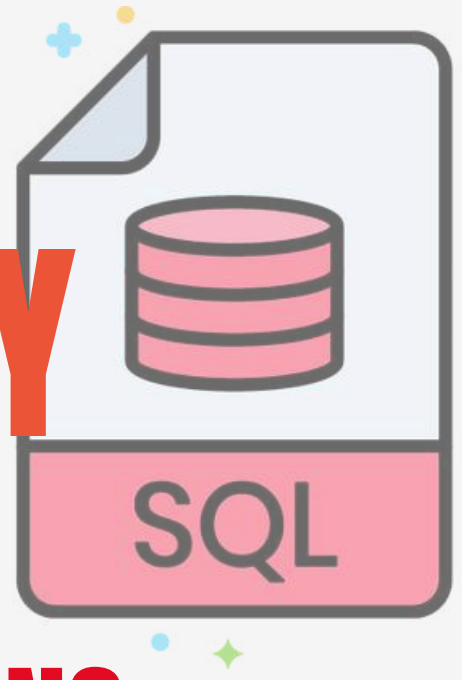
A QUERY NESTED INSIDE ANOTHER QUERY.

```
SELECT *
FROM Employees
WHERE DeptID = (SELECT ID
FROM Departments
WHERE Name = 'HR');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CORRELATED SUBQUERY



A SUBQUERY THAT REFERENCES COLUMNS FROM THE OUTER QUERY.

```
SELECT Name
FROM Employees E1
WHERE Salary > (SELECT AVG(Salary)
FROM Employees E2
WHERE E1.DeptID = E2.DeptID);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

VIEW

A VIRTUAL TABLE BASED ON THE RESULT-SET OF AN SQL STATEMENT.

```
CREATE VIEW EmployeeView
AS
SELECT Name, Age
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

INDEX

USED TO SPEED UP THE PERFORMANCE OF QUERIES.

```
CREATE INDEX idx_name
ON Employees (Name);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

TRIGGER

EXECUTES A BATCH OF SQL CODE WHEN AN INSERT, UPDATE OR DELETE COMMAND IS RUN AGAINST A SPECIFIC TABLE.

```
CREATE TRIGGER trg_after_insert
ON Employees AFTER INSERT
AS
BEGIN
PRINT 'New Employee Inserted';
END;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

PROCEDURE

A STORED SUBROUTINE AVAILABLE TO APPLICATIONS ACCESSING A RELATIONAL DATABASE SYSTEM.

```
CREATE PROCEDURE GetEmployee @ID int
AS
BEGIN
SELECT * F
ROM Employees
WHERE ID = @ID;
END;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

FUNCTION

A SUBROUTINE AVAILABLE TO APPLICATIONS ACCESSING A RELATIONAL DATABASE SYSTEM THAT RETURNS A SINGLE VALUE.

```
CREATE FUNCTION GetEmployeeName (@ID int)
RETURNS varchar(255)
AS
BEGIN
DECLARE @Name varchar(255);
SELECT @Name = Name F
ROM Employees
WHERE ID = @ID;
 RETURN @Name; END;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CURSOR

A DATABASE OBJECT USED TO RETRIEVE DATA ROW-BY-ROW.

```
DECLARE cursor_name
 CURSOR
FOR SELECT Name
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

FETCH

RETRIEVES ROWS ONE AT A TIME, OR IN A BLOCK, FROM THE RESULT SET OF A MULTI-ROW QUERY.

```
FETCH NEXT
FROM cursor_name
INTO @Name;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CLOSE

CLOSES THE CURSOR AND RELEASES THE CURRENT RESULT SET.

```
CLOSE cursor_name;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DEALLOCATE

REMOVES A CURSOR REFERENCE AND RELEASES RESOURCES.

```
DEALLOCATE cursor_name;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DECLARE

DECLARES A VARIABLE OR CURSOR.

```
DECLARE @Age int;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SET

INITIALIZES OR ASSIGNS A VALUE TO A VARIABLE.

```
SET @Age = 30;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

RAISERROR

RETURNS A USER-DEFINED ERROR MESSAGE.

```
RAISERROR('This is an error message', 16, 1);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

TRY...CATCH

HANDLES EXCEPTIONS IN T-SQL CODE.

```
BEGIN TRY;
SELECT 1/0;
END TRY
BEGIN CATCH;
PRINT 'Error';
END CATCH;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

TRANSACTION

A SEQUENCE OF OPERATIONS PERFORMED AS A SINGLE LOGICAL UNIT OF WORK.

```
BEGIN TRANSACTION;
UPDATE Employees
SET Age = 30
WHERE Name = 'John';

COMMIT;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

COMMIT

SAVES THE CHANGES MADE IN THE TRANSACTION.

```
COMMIT TRANSACTION;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ROLLBACK

UNDOES THE CHANGES MADE IN THE TRANSACTION.

```
ROLLBACK TRANSACTION;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SAVEPOINT

SETS A POINT WITHIN A TRANSACTION TO WHICH A ROLLBACK CAN OCCUR.

```
SAVE TRANSACTION savepoint_name;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

SET TRANSACTION ISOLATION LEVEL

SETS THE ISOLATION LEVEL FOR THE CURRENT SESSION.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

BULK INSERT

IMPORTS A LARGE AMOUNT OF DATA INTO A TABLE.

```
BULK INSERT Employees
FROM 'datafile.txt'
WITH (FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

TEMPORARY TABLE

A TABLE THAT IS CREATED AND CAN BE AUTOMATICALLY DELETED WHEN NO LONGER USED.

```
CREATE TABLE #TempTable
(ID int,
Name varchar(255));
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

RECURSIVE CTE

COMMON TABLE EXPRESSIONS THAT REFER TO THEMSELVES.

```
WITH RECURSIVE CTE AS
(SELECT 1 AS n
UNION ALL
SELECT n+1
FROM CTE
WHERE n < 10)
SELECT *
FROM CTE;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

WINDOW FUNCTION

PERFORMS A CALCULATION ACROSS A SET OF TABLE ROWS RELATED TO THE CURRENT ROW.

```
SELECT Name,
Salary,
AVG(Salary) OVER (PARTITION BY
Department)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

RANK

ASSIGNS A RANK TO EACH ROW WITHIN THE PARTITION OF A RESULT SET.

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

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DENSE_RANK

ASSIGNS RANKS TO ROWS IN AN ORDERED PARTITION WITHOUT GAPS IN RANK VALUES.

```
SELECT Name,
Salary,
DENSE_RANK()
OVER (ORDER BY Salary DESC)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ROW_NUMBER

ASSIGNS A UNIQUE SEQUENTIAL INTEGER TO ROWS WITHIN A PARTITION.

```
SELECT Name,
Salary,
ROW_NUMBER() OVER (ORDER BY
Salary DESC)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

NTILE

DISTRIBUTES ROWS OF AN ORDERED PARTITION INTO A SPECIFIED NUMBER OF GROUPS.

```
SELECT Name,
Salary,
NTILE(4) OVER (ORDER BY Salary DESC)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LAG

ACCESSES DATA FROM A PREVIOUS ROW IN THE SAME RESULT SET.

```
SELECT Name,
Salary,
LAG(Salary, 1) OVER (ORDER BY Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

LEAD

ACCESSES DATA FROM A SUBSEQUENT ROW IN THE SAME RESULT SET.

```
SELECT Name,
Salary,
LEAD(Salary, 1) OVER (ORDER BY Salary)
FROM Employees;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

PIVOT

TRANSFORMS ROWS INTO COLUMNS.

```
SELECT *
FROM (SELECT Department, Salary
        FROM Employees)
PIVOT (AVG(Salary)
FOR Department IN ([HR], [Finance]))
AS PVT;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

UNPIVOT

TRANSFORMS COLUMNS INTO ROWS.

```
SELECT *
FROM (SELECT Department, Salary
      FROM Employees)
UNPIVOT (Salary FOR Department
IN ([HR], [Finance]))
AS UPVT;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CROSS APPLY

APPLIES A TABLE-VALUED FUNCTION TO EACH ROW OF AN OUTER TABLE.

```
SELECT *
FROM Employees
CROSS APPLY
GetEmployeeDetails(Employees.ID);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CTE FOR HIERARCHICAL DATA

USES CTES TO HANDLE HIERARCHICAL DATA.

```
WITH CTE AS (
SELECT ID, ParentID, Name
FROM Employees
WHERE ParentID IS NULL
UNION ALL
 SELECT e.ID, e.ParentID, e.Name
FROM Employees e
INNER JOIN CTE c ON e.ParentID = c.ID)
SELECT * FROM CTE;
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

STRING_SPLIT

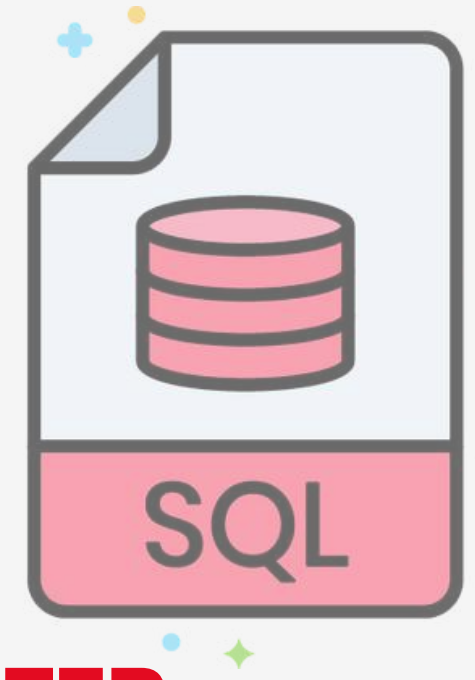
SPLITS A STRING INTO A TABLE OF SUBSTRINGS.

```
SELECT value
FROM
STRING_SPLIT('a,b,c', ',');
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

DYNAMIC SQL



SQL STATEMENTS THAT ARE CONSTRUCTED AND EXECUTED AT RUNTIME.

```
EXEC sp_executesql N'SELECT *
FROM Employees
WHERE Name = @name', N'@name
NVARCHAR(50)', @name = N'John';
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

CUBE SUBCLAUSE

A WAY TO GENERATE SUBTOTALS FOR ALL COMBINATIONS OF THE SPECIFIED COLUMNS.

```
SELECT Department, Year, SUM(Salary)
FROM Employees
GROUP BY CUBE (Department, Year);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

Data Engineering 101: SQL Basics

ROLLUP SUBCLAUSE

A WAY TO GENERATE SUBTOTALS THAT ROLL UP FROM THE MOST DETAILED LEVEL TO A GRAND TOTAL.

```
SELECT Department, Year, SUM(Salary)
FROM Employees
GROUP BY ROLLUP (Department, Year);
```

Shwetank Singh
GritSetGrow - GSGLearn.com

THANK you