CUI -> charcter user interface

GUI-> graphical user interface

**Database Commands:**

1. **CREATE DATABASE:** Create a new database.

Constraints Key

Default Key

alter table intern add constraint DF\_stipened default 0 for stipened

Unique Key

alter table intern add constraint uk\_intern unique(email)

Drop Any Constraint

alter table intern drop constraint pk\_ID

CREATE DATABASE MyDatabase;

1. **ALTER DATABASE:** Modify database properties.

ALTER DATABASE MyDatabase SET READ\_COMMITTED\_SNAPSHOT ON;

1. **DROP DATABASE:** Delete a database.

DROP DATABASE MyDatabase;

1. **DATABASE RENAME**

sp\_renamedb 'rise','rise1'

**Table Commands:**

1. **CREATE TABLE:** Create a new table.

CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), DepartmentID INT );

1. **ALTER TABLE:** Modify an existing table.

ALTER TABLE TableName DROP COLUMN ColumnName;

ALTER TABLE Employees ADD Email VARCHAR(100);

1. **DROP TABLE:** Delete a table.

DROP TABLE Employees;

**Data Manipulation Commands:**

1. **INSERT INTO:** Insert data into a table.

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID) VALUES (1, 'John', 'Doe', 101);

1. **UPDATE:** Update existing records in a table.

UPDATE Employees SET DepartmentID = 102 WHERE LastName = 'Doe';

update staff set Emp\_ID = 2, Dep\_ID = 'DS' where Emp\_ID is NULL;

1. **Pivot Table**

---pivot

select distinct customer\_name, country, sum(profit) as profit from sales\_data\_final

group by customer\_name, country

order by customer\_name, country.

1. **DELETE:** Delete row / records from a table.

DELETE FROM Employees WHERE EmployeeID = 1;

1. TABLE RENAAME QUERY in MS SQL

sp\_rename 'OldTableName', 'NewTableName';

1. Table Rename Column

sp\_rename Table\_Name.Column Name', 'New\_column\_Name';

DATABASE RENAME

alter database (OLD NAME) modify name = (NEW NAME)

1. **SELECT:** Retrieve data from a table.

SELECT FirstName, LastName FROM Employees WHERE DepartmentID = 102;

**Query Commands:**

1. **SELECT DISTINCT:** Retrieve distinct values.

SELECT DISTINCT DepartmentID FROM Employees;

1. **FROM:** Specify the source of data for a query.

SELECT FirstName, LastName FROM Employees;

1. **JOIN:** Combine rows from multiple tables based on a related column.

SELECT Employees.FirstName, Departments.DepartmentName FROM Employees INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

1. **WHERE:** Filter rows based on a condition.

SELECT FirstName, LastName FROM Employees WHERE DepartmentID = 102;

1. **GROUP BY:** Group rows based on a column or expression.

SELECT DepartmentID, COUNT(\*) AS EmployeeCount FROM Employees GROUP BY DepartmentID;

1. **HAVING:** Filter grouped rows.

SELECT DepartmentID, COUNT(\*) AS EmployeeCount FROM Employees GROUP BY DepartmentID HAVING COUNT(\*) > 5;

1. **ORDER BY:** Sort rows.

SELECT FirstName, LastName FROM Employees ORDER BY LastName, FirstName;

1. **LIMIT / OFFSET:** Limit the number of rows returned (used in some SQL variants).

SELECT FirstName, LastName FROM Employees ORDER BY LastName, FirstName OFFSET 10 ROWS FETCH NEXT 5 ROWS ONLY;

1. **TOP:** Retrieve a specified number of rows from the beginning.

SELECT TOP 5 FirstName, LastName FROM Employees ORDER BY LastName, FirstName;

**Aggregate Functions:**

1. **COUNT:** Count the number of rows.

SELECT COUNT(\*) AS TotalEmployees FROM Employees;

1. **SUM:** Calculate the sum of values in a column.

SELECT SUM(Salary) AS TotalSalary FROM EmployeeSalaries;

1. **AVG:** Calculate the average of values in a column.

SELECT AVG(Age) AS AverageAge FROM Persons;

1. **MIN:** Find the minimum value in a column.

SELECT MIN(Price) AS LowestPrice FROM Products;

1. **MAX:** Find the maximum value in a column.

SELECT MAX(Salary) AS HighestSalary FROM EmployeeSalaries;

**Subqueries:**

1. **IN:** Check if a value matches any value in a subquery result.

SELECT ProductName FROM Products WHERE CategoryID IN (SELECT CategoryID FROM Categories WHERE CategoryName = 'Electronics');

1. **EXISTS:** Check if a subquery returns any rows.

SELECT OrderID FROM Orders WHERE EXISTS (SELECT \* FROM OrderDetails WHERE OrderDetails.OrderID = Orders.OrderID);

1. **NOT EXISTS:** Check if a subquery returns no rows.

SELECT CustomerID FROM Customers WHERE NOT EXISTS (SELECT \* FROM Orders WHERE Orders.CustomerID = Customers.CustomerID);

1. **ANY / ALL:** Compare a value to all values in a subquery.

SELECT ProductName FROM Products WHERE Price > ALL (SELECT Price FROM Products WHERE CategoryID = 1);

1. **Subquery in SELECT:** Use a subquery to retrieve data for a column.

SELECT ProductName, (SELECT AVG(Price) FROM Products) AS AvgPrice FROM Products;

**String Functions:**

1. **CONCAT:** Concatenate strings.

SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM Employees;

1. **SUBSTRING:** Extract a substring from a string.

SELECT SUBSTRING(ProductName, 1, 10) AS ShortName FROM Products;

1. **CHARINDEX:** Find the position of a substring within a string.

SELECT CHARINDEX('SQL', 'Learn SQL Server') AS Position;

1. **LEN:** Calculate the length of a string.

SELECT LEN(ProductName) AS NameLength FROM Products;

1. **LOWER / UPPER:** Convert a string to lowercase or uppercase.

SELECT UPPER(ProductName) AS UppercaseName FROM Products;

1. **LTRIM / RTRIM:** Remove leading or trailing spaces.

SELECT LTRIM(RTRIM(CompanyName)) AS TrimmedName FROM Customers;

**Date and Time Functions:**

1. **GETDATE:** Get the current date and time.

SELECT GETDATE() AS CurrentDateTime;

1. **DATEADD:** Add an interval to a date or time.

SELECT DATEADD(DAY, 7, OrderDate) AS NewOrderDate FROM Orders;

1. **DATEDIFF:** Calculate the difference between two dates or times.

SELECT DATEDIFF(YEAR, Birthdate, GETDATE()) AS Age FROM Employees;

1. **FORMAT:** Format a date or time value.

SELECT FORMAT(OrderDate, 'yyyy-MM-dd HH:mm:ss') AS FormattedDate FROM Orders;

**NULL Handling:**

1. **IS NULL:** Check for NULL values.

SELECT FirstName, LastName FROM Employees WHERE Email IS NULL;

1. **IS NOT NULL:** Check for non-NULL values.

SELECT ProductName FROM Products WHERE Description IS NOT NULL;

1. **COALESCE:** Return the first non-NULL value in a list.

SELECT COALESCE(ShippingAddress, BillingAddress, 'N/A') AS Address FROM Orders;

**Indexes and Constraints:**

1. **CREATE INDEX:** Create an index on one or more columns.

CREATE INDEX IX\_LastName ON Employees (LastName);

1. **DROP INDEX:** Delete an index.

DROP INDEX IX\_LastName ON Employees;

1. **PRIMARY KEY:** Define a primary key constraint.

ALTER TABLE Orders ADD CONSTRAINT PK\_Orders PRIMARY KEY (OrderID);

1. **FOREIGN KEY:** Define a foreign key constraint.

ALTER TABLE OrderDetails ADD CONSTRAINT FK\_OrderDetails\_OrderID FOREIGN KEY (OrderID) REFERENCES Orders(OrderID);

1. **UNIQUE:** Define a unique constraint.

ALTER TABLE Customers ADD CONSTRAINT UQ\_Email UNIQUE (Email);

**Views and Synonyms:**

* **Create mirror table with live update**

Create view table\_name from select \* from Emp

1. **CREATE VIEW:** Create a virtual table based on a query.

CREATE VIEW HighSalaryEmployees AS SELECT FirstName, LastName FROM Employees WHERE Salary > 50000;

1. **ALTER VIEW:** Modify an existing view.

ALTER VIEW HighSalaryEmployees AS SELECT FirstName, LastName FROM Employees WHERE Salary > 60000;

1. **DROP VIEW:** Delete a view.

DROP VIEW HighSalaryEmployees;

1. **CREATE SYNONYM:** Create an alias for an object.

CREATE SYNONYM SalesSummary FOR AdventureWorks2019.dbo.SalesSummary;

**Stored Procedures and Functions:**

1. **CREATE PROCEDURE:** Create a stored procedure.

CREATE PROCEDURE GetEmployeeByID @EmployeeID INT AS BEGIN SELECT FirstName, LastName FROM Employees WHERE EmployeeID = @EmployeeID; END;

1. **ALTER PROCEDURE:** Modify an existing stored procedure.

ALTER PROCEDURE GetEmployeeByID @EmployeeID INT AS BEGIN SELECT FirstName, LastName, DepartmentID FROM Employees WHERE EmployeeID = @EmployeeID; END;

1. **DROP PROCEDURE:** Delete a stored procedure.

DROP PROCEDURE GetEmployeeByID;

1. **CREATE FUNCTION:** Create a user-defined function.

CREATE FUNCTION CalculateTotalPrice (@Quantity INT, @UnitPrice DECIMAL(10, 2)) RETURNS DECIMAL(10, 2) AS BEGIN RETURN @Quantity \* @UnitPrice; END;

1. **ALTER FUNCTION:** Modify an existing user-defined function.

ALTER FUNCTION CalculateTotalPrice (@Quantity INT, @UnitPrice DECIMAL(10, 2)) RETURNS DECIMAL(10, 2) AS BEGIN RETURN @Quantity \* @UnitPrice \* 1.1; -- Apply a 10% tax END;

1. **DROP FUNCTION:** Delete a user-defined function.

DROP FUNCTION CalculateTotalPrice;

**Transactions:**

1. **BEGIN TRANSACTION:** Start a new transaction.

BEGIN TRANSACTION;

1. **COMMIT:** Save changes and end the current transaction.

COMMIT;

1. **ROLLBACK:** Undo changes and end the current transaction.

ROLLBACK;

**Security Commands:**

1. **GRANT:** Grant permissions to users or roles.

GRANT SELECT, INSERT, UPDATE ON Employees TO User1;

1. **REVOKE:** Revoke previously granted permissions.

REVOKE DELETE ON Orders FROM User2;

**Backup and Restore:**

1. **BACKUP DATABASE:** Create a backup of a database.

BACKUP DATABASE MyDatabase TO DISK = 'C:\Backup\MyDatabase.bak';

1. **RESTORE DATABASE:** Restore a database from a backup.

USE master; RESTORE DATABASE MyRestoredDatabase FROM DISK = 'C:\Backup\MyDatabase.bak';

Please note that the above examples are illustrative and may need to be adapted to your specific database structure and data. Always refer to the official documentation for Microsoft SQL Server and practice caution when using SQL commands, especially in a production environment.

**Query Commands:**

1. Retrieve distinct values:

SELECT DISTINCT DepartmentID FROM Employees;

1. Specify the source of data for a query:

SELECT FirstName, LastName FROM Employees;

1. Combine rows from multiple tables with INNER JOIN:

SELECT Employees.FirstName, Departments.DepartmentName FROM Employees INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

1. Filter rows using WHERE:

SELECT FirstName, LastName FROM Employees WHERE DepartmentID = 102;

1. Group rows using GROUP BY:

SELECT DepartmentID, COUNT(\*) AS EmployeeCount FROM Employees GROUP BY DepartmentID;

1. Filter grouped rows using HAVING:

SELECT DepartmentID, COUNT(\*) AS EmployeeCount FROM Employees GROUP BY DepartmentID HAVING COUNT(\*) > 5;

1. Sort rows using ORDER BY:

SELECT FirstName, LastName FROM Employees ORDER BY LastName, FirstName;

1. Limit the number of rows using OFFSET and FETCH (SQL Server 2012+):

SELECT FirstName, LastName FROM Employees ORDER BY LastName, FirstName OFFSET 10 ROWS FETCH NEXT 5 ROWS ONLY;

1. Retrieve a specified number of rows using TOP (SQL Server):

SELECT TOP 5 FirstName, LastName FROM Employees ORDER BY LastName, FirstName;

**Aggregate Functions:**

1. Count the number of rows:

SELECT COUNT(\*) AS TotalEmployees FROM Employees;

1. Calculate the sum of values in a column:

SELECT SUM(Salary) AS TotalSalary FROM EmployeeSalaries;

1. Calculate the average of values in a column:

SELECT AVG(Age) AS AverageAge FROM Persons;

1. Find the minimum value in a column:

SELECT MIN(Price) AS LowestPrice FROM Products;

1. Find the maximum value in a column:

SELECT MAX(Salary) AS HighestSalary FROM EmployeeSalaries;

**Subqueries:**

1. Check if a value matches any value in a subquery result using IN:

SELECT ProductName FROM Products WHERE CategoryID IN (SELECT CategoryID FROM Categories WHERE CategoryName = 'Electronics');

1. Check for existence using EXISTS:

SELECT OrderID FROM Orders WHERE EXISTS (SELECT \* FROM OrderDetails WHERE OrderDetails.OrderID = Orders.OrderID);

1. Check for non-existence using NOT EXISTS:

SELECT CustomerID FROM Customers WHERE NOT EXISTS (SELECT \* FROM Orders WHERE Orders.CustomerID = Customers.CustomerID);

1. Compare a value to all values in a subquery using ANY / ALL:

SELECT ProductName FROM Products WHERE Price > ALL (SELECT Price FROM Products WHERE CategoryID = 1);

1. Use a subquery in SELECT to retrieve data for a column:

SELECT ProductName, (SELECT AVG(Price) FROM Products) AS AvgPrice FROM Products;

**String Functions:**

1. Concatenate strings using CONCAT:

SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM Employees;

1. Extract a substring using SUBSTRING:

SELECT SUBSTRING(ProductName, 1, 10) AS ShortName FROM Products;

1. Find the position of a substring within a string using CHARINDEX:

SELECT CHARINDEX('SQL', 'Learn SQL Server') AS Position;

1. Calculate the length of a string using LEN:

SELECT LEN(ProductName) AS NameLength FROM Products;

1. Convert a string to uppercase using UPPER:

SELECT UPPER(ProductName) AS UppercaseName FROM Products;

1. Remove leading or trailing spaces using LTRIM / RTRIM:

SELECT LTRIM(RTRIM(CompanyName)) AS TrimmedName FROM Customers;

**Date and Time Functions:**

1. Get the current date and time using GETDATE:

SELECT GETDATE() AS CurrentDateTime;

1. Add an interval to a date or time using DATEADD:

SELECT DATEADD(DAY, 7, OrderDate) AS NewOrderDate FROM Orders;

1. Calculate the difference between two dates or times using DATEDIFF:

SELECT DATEDIFF(YEAR, Birthdate, GETDATE()) AS Age FROM Employees;

1. Format a date or time value using FORMAT (SQL Server 2012+):

SELECT FORMAT(OrderDate, 'yyyy-MM-dd HH:mm:ss') AS FormattedDate FROM Orders;

**NULL Handling:**

1. Check for NULL values using IS NULL:

SELECT FirstName, LastName FROM Employees WHERE Email IS NULL;

1. Check for non-NULL values using IS NOT NULL:

SELECT ProductName FROM Products WHERE Description IS NOT NULL;

1. Return the first non-NULL value in a list using COALESCE:

SELECT COALESCE(ShippingAddress, BillingAddress, 'N/A') AS Address FROM Orders;

* Insert Bulk data

BULK INSERT TargetTable FROM 'C:\Path\To\Your\File.csv' WITH ( FIELDTERMINATOR = ',', ROWTERMINATOR = '\n', FIRSTROW = 2 -- Skip header row if applicable );

1. Update single / multiple cell value

update employee set emp\_name = 'vineeta singh', where emp\_id = 28;

1. Check Available database in MS SQL

SELECT name

FROM sys.databases

WHERE database\_id >4;

1. To check tables name in database

SELECT name

FROM sys.tables;

Identity Key

* To turn off auto increment

Set identity Tablename on

* To turn on auto increment

Set identity Tablename oFF

-- Using UNION (removes duplicates)

SELECT column1, column2, ...

FROM table1

WHERE condition

* **UNION**

SELECT column1, column2, ...

FROM table2

WHERE condition;

-- Using UNION ALL (includes duplicates)

SELECT column1, column2, ...

FROM table1

WHERE condition

UNION ALL

SELECT column1, column2, ...

FROM table2

WHERE condition;

OPERATOR COMMAND

OR QUERY

select \* from staff where Dep\_ID = 'DA' and stack\_ID ='DS'

NOT Query

select \* from staff where not Dep\_ID = 'DA'

DATE Functions IN MS SQL

* For todays date below

Select getdate()

* To calculate difference between two dates

Datediff(Days/Month/Year, start\_date, end\_date/getdate())

* To add value in mentioned date

select dateadd(month,10,'2023-06-20')

* To get a part of date

select datepart(month,'2023-06-20')

* ORDER BY ASC /DESC

select\* from intern order by identity\_no asc

select\* from intern order by identity\_no dsc

--- Left Join

select \* from intern left join staff on intern.Dep\_ID = staff.HOD\_Name order by identity\_no asc

--- LEFT Outer Join

select \* from intern left join staff on intern.Dep\_ID = staff.HOD\_Name where Emp\_ID is null order by identity\_no ASC

--- Right JOin

select \* from intern right join staff on intern.Dep\_ID = staff.HOD\_Name order by identity\_no asc

- Right Outer Join

select \* from intern right join staff on intern.Dep\_ID = staff.HOD\_Name where Emp\_ID is null order by identity\_no asc

----Self Join

select a.enrolmenet\_id,b.studen\_name

from table1 a, table1 b

where a.enrolmenet\_id< b.enrolmenet\_id

--- EQUIE JOin

select table1.studen\_name, table1.enrolmenet\_id,table2.student\_dob,table2.contact\_no

from table1 join table2 on table1.enrolmenet\_id=table2.enrolmenet\_id

---NON EQUIE JOIN

select \* from table1 join table2 on table1.enrolmenet\_id<table2.enrolmenet\_id

create database joins

create table order\_status(

order\_ID bigint,

customer\_id bigint,

order\_date date

)

create table customer\_detail(

order\_ID bigint,

Customer\_Name nvarchar(40),

order\_date date

)

bulk insert customer\_detail from 'D:\DATABASE\Customer Detaisl.csv' with

(format ='CSV', fieldterminator = ',', firstrow = 2, rowterminator ='\n')

bulk insert order\_status from 'D:\DATABASE\order detail.csv' with

(format ='CSV', fieldterminator = ',' , firstrow = 2, rowterminator ='\n')

select \* from order\_status

select \* from customer\_detail

alter table customer\_detail alter column Order\_ID bigint not null

alter table customer\_detail add constraint PK\_Order\_ID primary key (order\_ID)

alter table order\_status add constraint FK\_Order\_ID foreign key ( Order\_ID )

references customer\_detail(order\_ID) on update cascade on delete cascade

sp\_help order\_status

sp\_rename 'customer\_detail.order\_ID','Customer\_ID'

select order\_status.order\_ID, customer\_detail.Customer\_Name, customer\_detail.customer\_ID from

order\_status inner join customer\_detail on order\_status.customer\_id=customer\_detail.customer\_ID