INDEX & SUMMARY

SCHEMA- A Schema is the Database design and Collection of common database objects like Tables, Stored Procedures, Triggers.

INSERT - It is used to store data in database.

```
INSERT INTO Users (Name, Email, Phone, Password)
VALUES
('Prashant Rai', 'prashant@gmail.com', '7359050908', 'abcdef12345')
```

SELECT - It is used to select data from a table W/WO Condition.

```
SELECT * FROM Country
WHERE DialingCode = 91
```

UPDATE - It is used to Update the record in the table.

```
UPDATE  City
SET CityName = 'Dhrol'
WHERE District = "Jamnagar'
```

DELETE - It is used to delete the record from table

```
DELETE FROM  City
WHERE CityName = 'Morbi'
```

Foreign Key - It is used to link the Primary key of one table to a connected column of another table. Data can be joined from both the table using that foreign key which is common on both the tables. There can be multiple foreign keys in one table and also called as child table

```
SELECT Country, State, City
FROM Country
INNER JOIN State ON Country.CountryId = State.CountryID
INNER JOIN City ON Country.CountryId = City.CountryID
```

GROUP BY - It is used to make a group of common values in a field or common rows. It is used with Aggregate Functions like Count, Min, Max, Avg, Sum. Lets example we have multiple state of same country so here in group by we can make country groups of states.

```
SELECT COUNT(StateName) AS Total, CountryID
FROM States
WHERE CountryID = 75
```

```
GROUP BY CountryID
ORDER BY CountryID DESC;
```

HAVING - It is used in place of WHERE Keyword and mostly with the aggregate functions and it is written after group by keyword to put any condition or something.

```
SELECT COUNT(Users) AS COUNTS
FROM Users
WHERE CountryID =91
GROUP BY CountryID
HAVING COUNT(USers) >5
```

UNION - It is used to combine 2 or more select statements. Every select statements must have same datatype, same order of columns and same number of columns.

```
SELECT CountryID FROM City
UNION
SELECT CountryID FROM  State
ORDER BY CountryID ASC
```

UNION ALL - It is same as UNION but it also allows duplicate values and gives union of both the queries.

INTERSECTS - It is used to find the common data between the 2 or more select queries

There are some mandatory rules for INTERSECT operations such as the number of columns, data types, and other columns must be the same in both SELECT statements for the INTERSECT operator to work correctly

```
SELECT CountryID FROM City
INTERSECTS
SELECT CountryID FROM  State
ORDER BY CountryID ASC
```

EXCEPT - it will display the distinct records only from the first table which are not in common with the records of the second table.

```
SELECT NAME, HOBBY, AGE FROM STUDENTS
EXCEPT
SELECT NAME, HOBBY, AGE FROM STUDENTS_HOBBY;
```

TOP - It gives you the top x number of results. It is useful for large datasets.

```
SELECT TOP 3 FROM Customers WHERE Salary > 10000;

SELECT TOP 50  PERCENT * FROM Customers WHERE Salary > 10000;
```

DISTINCT - it is used to return only distinct values or unique values

LIKE - It is used to match a pattern for searching or validation to follow a specific pattern to check conditions and return the record.

```
SELECT * FROM  Users
WHERE
UserName LIKE 'A%';
```

BETWEEN - it returns a record between a particular range of numbers , strings, dates, etc. Begin and end values are inclusive.

AND - It is used in conditions where all the conditions should satisfy to return the result.
OR - It is used to put in such a case where any of the following should be true to return the result.

IN-NOT IN - It is used to check whether the following data occurs in a particular varchar, range or something. It is a shorthand for multiple OR usage in a query.

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

NULL- NOT NULL- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
Whereas in case of not null it is fixed no null values will be there.

RAND() - It generates a random number between a provided range.

```
SELECT RAND()*(10-5)+5;
```

JOINS - It is used to find data from another table which is sometimes required to show and this can be only done by joining the tables together. To join tables together we need to have a common field which is a foreign key.

There are mainly 4 types of Joins but some are there like CROSS JOINS

INNER JOIN - It is used to find the common data between two tables

```
SELECT StateName, CountryName
```

```
FROM State
INNER JOIN
Country ON State.CountryID = Country.CountryId
```

FULL OUTER JOIN / FULL JOIN - It is used to return the data of two tables when it matches the column name. It returns all the data (NULL values also occur in large amount)

```
SELECT StateName, CountryName
FROM State
FULL OUTER  JOIN
Country ON State.CountryID = Country.CountryId
```

LEFT JOIN - It is used to find the common data between two tables and the full data of the first / left table not the right one.

```
SELECT StateName, CountryName
FROM State
LEFT JOIN
Country ON State.CountryID = Country.CountryId
```

RIGHT JOIN - It is used to find the common data between two tables and the full data of the second / right table not the left one.

```
SELECT StateName, CountryName
FROM State
RIGHT JOIN
Country ON State.CountryID = Country.CountryId
```

SELF JOIN - Self-join allows us to join a table itself. It is useful when a user wants to compare the data (rows) within the same table.

```
SELECT e.FirstName, m.FirstName AS ManagerFirstName
FROM Employees e
LEFT JOIN Employees m ON e.ManagerID = m.EmployeeID;
```

CROSS JOINS - Cross join allows us to join each and every row of both the tables. It is similar to the cartesian product that joins all the rows.

```
SELECT *
FROM Employees
CROSS JOIN Managers
```

VIEWS - a view is a virtual table that represents the result of a SELECT query. Views are not physical tables; they are saved SQL statements that can be used to simplify complex queries and provide a logical, abstracted representation of the data in one or more base tables.

```
CREATE VIEW EmployeeDetails AS
SELECT EmployeeID, FirstName, LastName, Department
FROM Employees;
```

```
SELECT * FROM EmployeeDetails;
```

STORED PROCEDURES - It is used to write the Business Logic at one place and used it further wherever required. It's like writing once and using it forever. We can also pass parameters to use it dynamically on runtime.

```
CREATE PROCEDURE spLoginAuth
@username VARCHAR(50),
@password VARCHAR(16)
WITH ENCRYPTION
AS
BEGIN
	SELECT * FROM Admin
	WHERE Username = @username AND Password = @password
END

EXEC spLoginAuth @usernmae = 'admin' , @password = 'admin123' ;
```

PAGINATION - it is used to show and skip data in a sequence and format. Keywords used are Offset and fetch.

OFFSET is used for skipping the values or a specified number of rows.
FETCH retrieves a specific number of rows .

Example: Get 10 rows starting from the 21st row (skip the first 20 rows)

```
SELECT *
FROM users
```

ORDER BY name
OFFSET 20 ROWS
FETCH NEXT 10 ROWS ONLY;

FUNCTIONS -

String Functions -
ASCII, CHAR, CHARINDEX, CONCAT, Concat with just +, CONCAT_WS, DATALENGTH, DIFFERENCE, FORMAT, LEFT, **LEN**, LTRIM, **LOWER**, NCHAR, PATINDEX, QUOTENAME, REPLACE, REPLICATE, **REVERSE**, RIGHT, RTRIM, SOUNDEX, SPACE, STR, STUFF, **SUBSTRING**, TRANSLATE, TRIM, UNICODE, **UPPER**

Math Functions -
ABS, **AVG**, **COUNT**, MAX, MIN, **SUM**, RAND, ROUND, SQRT, SQUARE

Date and Time Functions -

CURRENT_TIMESTAMP - Return current date and time. (its sql focused, portable, consistent for shifting to other databases)
GETDATE() - Return the current database system date and time. (its legacy oriented like old database or projects oriented)
SYSDATETIME - Returns date and time of sql server

Advanced Functions -
CAST - The CAST() function converts a value (of any type) into a specified datatype. (its sql oriented and good for portability)

CONVERT - The CONVERT () function converts a value (of any type) into a specified datatype.(its sql-server oriented only but the good part is that It not only change data type but also can change format in a required way)

COALESCE - returns the first non-null (which is not a null value) value in a list.
IIF - returns a value if a condition is TRUE, or another value if a condition is FALSE.
ISNULL - returns a specified value if the expression is NULL. If the expression is NOT NULL, this function returns the expression.

TO RUN SQL IN CMD **sqlcmd -E** -> **SELECT name FROM sys.databases GO**

DATABASE SHIPPING - it is a way to shift the database files or data from one database or server instance to another i.e, from one location to another. It can involve creating backup files, generating scripts or physically moving files from the database.

Backup file - It is the file (.bak) which is generated from SSMS to take backup in order to preserve it if any hardware failures or something drastically happpens.

Scripting DB - It is used to generate script files which will generate database schema, tables, constraints, indexes, etc which will be further required to generate a new database but with this script file only and script file is slightly different from origin query file becoz it is not showing the line by line execution of each query.

Attach DB - In this way we attach the existing db files like LDF and MDF to another database server instance.

LDF (Log Data File) - it is a transaction record file which keeps the record of all the transactions and its modifications so that whenever we have some troubles we can easily access this record for our purpose.

MDF (Master Data File) - It is a primary data file in any database which keeps the records of objects like schemas, tables, indexes, views, etc. It is a very important file and the majority of the data is stored on this file only. One database can only have a single mdf file but it can have multiple secondary files like NDF. MDF files is having extensions of .mdf at the end.

IF-ELSE - is used to check some conditions and then execute the query

```
DECLARE @Score INT = 75;

IF @Score >= 70
BEGIN
   PRINT 'Pass';
END
ELSE
BEGIN
   PRINT 'Fail';
END
```

CASE-WHEN - is used similar to if else and are more efficient than IF-ELSE

```
SELECT
   OrderID,
   ProductName,
   Quantity,
   CASE
      WHEN Quantity > 10 THEN 'High'
      WHEN Quantity > 5 THEN 'Medium'
      ELSE 'Low'
   END AS QuantityCategory
FROM Orders;
```

Virtual Table or Memory Table - a virtual table is something which is stored virtually in memory but not the actual table or something. It is used to show some intermediate results or data to the client without intervening the actual table. So it's a good option for security purposes also. It's duration is till the session of the database and then it's discarded afterwards unlike VIEW it is stored in the database and needs to be dropped or removed.

There are some differences between CREATE VIEW and CTE i.e, created with WITH keyword and that is the persistence. VIEW can remain in the database until dropped but WITH is discarded when the session is over or database is closed. So it's good bcoz it is efficient and shows data quickly.

So if we want to store for future purposes then view is good else CTE is a good option.

INDEX - It is used to search data in a table using indexing which a way to store data in a way where it is sorted and easily found which makes our searching task easy and efficiently fast than the traditional searching from a table.

Points to keep in mind before using indexing :

1) Never use indexing when data is updated frequently bcoz it can easily drop your database performance.
2) As it takes memory apart from the table memory consumption, so don't use it unnecesarily.
3) Use it when highly searching is required without any updation.

We can check whether the scanning has been done through the index or normal table scan. For that we need to write an EXPLAIN keyword before the query we need to search.
Also there is another way also to find out the time taken to execute the query like CPU time, elapsed time and total time .

```
CREATE  INDEX IX_Country
ON Country(DialingCode)

EXPALIN SELECT * FROM Country WHERE DialingCode = 91;
SET STATISTICS TIME ON SELECT * FROM Country WHERE DialingCode = 91;
DROP  INDEX Country.IX_Country
```

Clustered Indexing: we can create only single clustered indexing which is majorly a primary key or an index key of the table and also the data is stored physically. It is faster but the issue is that it is a big trouble if we change the data, it would not be a good option then. If multiple columns are there then called as composite clustered index.it is costly.Just use CLUSTERED keyword to use it.

Non-clustering index: we can have multiple non-clustering indexes and it is stored separately and occupies storage too for that. If any update operations happen then it is a little bit easy to handle as compared to the clustered one.It is by default and not costly.

FTS (Full-Text_Search) - It is used to do advanced search like fuzzy searches and natural language searches capabilities.we need to create full-text catalog which will define which table and column need to be searched. Then we need to define full-text indexes on specific table and columns. Then before using full-text-search we need to populate data into index using this command
EXEC sp_fulltext_catalog 'catalog_name', 'start_full';

And then we can make searchings using two keywords contains and freetext

```
SELECT * FROM table_name
WHERE CONTAINS(column_name, 'search_term');
                                    OR
SELECT * FROM table_name
WHERE FREETEXT(column_name, 'search_term');
```

UDF (User Defined Function) - functions which are created by developers to encapsulate sql queries and statements into reusable, modular unit. It can be used to simplify complex sql queries and to make the code organized.
3 types are Scalar, Inline Table Valued, Multi Statement table valued.

Scalar UDF - This function is used to when one or more input parameters are there and that returns single value result also called as Scalar Value. It can be of any datatype.

For example - you want to calculate GST for that we'll have some input parameters lile @cgst and @sgst , @amount , etc and then we can calculate and return final result as single result.

```
CREATE FUNCTION dbo.CalculateTax(@amount DECIMAL(10, 2), @taxRate DECIMAL(5, 2))
RETURNS DECIMAL(10, 2)
AS
BEGIN
   RETURN @amount * @taxRate;
END;
```

ITV (Inline Table Valued UDF) - It is used to encapsulate a single table and is used as a part of a query. It is called inline bcoz it is incorporated directly into the main query thus improves performance. It is used for simple scenarios.

```
CREATE FUNCTION dbo.GetHighSalaryEmployees(@salaryThreshold DECIMAL(10, 2))
RETURNS TABLE
AS
RETURN (SELECT * FROM Employees WHERE Salary >= @salaryThreshold);
```

MSTV (Multi Statement table valued) - it is used for complex scenerios. It can contain multiple sql statements also we can do several things are aggregating data, data cleansing, custom data retrieval, recursive queries.

```
CREATE FUNCTION CalculateRunningTotal()
RETURNS @Result TABLE (ID INT, Value INT, RunningTotal INT)
AS
BEGIN
   DECLARE @Sum INT = 0;

   INSERT INTO @Result (ID, Value, RunningTotal)
   SELECT ID, Value, (@Sum := @Sum + Value) AS RunningTotal
   FROM YourTable;

   RETURN;
```

END;

Ranking Functions - it is used to rank or position data in one or more columns. It is used for generating reports, get top or bottom results , etc.

ROW_NUMBER() - it is used to get the unique ranks of a particular columns. Ranking is sequential and no gaps

RANK() - it is used to get the rank in each row but it skips the if similar values occur in rows and skip next rank.

DESNE_RANK() - it is used to get the same thing but the next rank is not skip..

NTILE(N) - it is used to divide number of rows as per specified partition and assign unique value in the partition. Eg category or bucket of rows divided especially used in percentile and grade remarks like distinct, first class, etc.

TVE (Table Valued Expression) - it is a relational operator which converts a row into columns

PIVOT - it is used to change data from rows into columns
UNPIVOT - it is used to change data from columns into rows
DYNAMIC PIVOT - it is used when we want to change data dynamically in pivot and unpivot it. We create SP to do so.

TRIGGERS - it is used to automatically trigger when any modification occurs in the table, database, etc.

2 types are DML and DDL:

DML (data Modification Language trigger) - it is used at the time of modification like UPDATE, DELETE, INSERT.
It is of 2 types AFTER and INSTEAD OF.

AFTER - it is used to show or log the message after inserting or modifications.

INSTEAD OF - it is used before modification and used for custom data modification like validation and etc.

DDL (data definition language trigger) - it is used or trigger fired during the defining of table, schema, db, etc. example are CREATE, ALTER, DROP used when auditing database or performing such queries of definition.

LOGON Trigger - it is used when a user logs into the database at that time it performs.

AFTER SERVER Trigger - it is used for server level actions like automatic shutdown, or backup etc.

CTE (Common Table Expression) - it is used as a temporary reference set which is used with INSERT, UPDATE, DELETE, SELECT Command. CTE is used with the WITH command and used to make complex queries into simpler ones.

EXCEPTION HANDLING - it is used to handle the exceptions in code, errors, invalid input, external factors, etc to handle it and prevent from crashing and database failures.

```
BEGIN TRY
       --code to try
END TRY
BEGIN CATCH
       --code to run if an error occurs
--is generated in try
END CATCH
```

Try and Catch is used try for running the code and catch for handling the exception part.

```
BEGIN TRY
   SELECT *
      CASE
         WHEN Age >= 18 THEN 'Has Voting Rights'
         ELSE 'Does Not Have Voting Rights'
      END AS VotingRights
   FROM Citizens;
END TRY
BEGIN CATCH
   -- Handle the exception
   PRINT 'An exception occurred: ' + ERROR_MESSAGE();
END CATCH;
```

TRANSACTION - a transaction is a series or sequence of sql statements executed. Now the transaction should have some properties known as ACID properties.

Atomicity - either the transaction should be executed or not executed not in between or something i.e, if any transaction or query fails then it should be ROlled back completely.

Consistency - data should be consistent without any break or something i.e it should have all the checks, constraints like PK, FK, etc before and after the transaction completion.

Isolation - it means one transaction is completely separate and invisible to another transaction until and unless it is committed so this will prevent data interference.
Durability - once a transaction is committed that simply means the data is permanent and can survive in system crash and failures to recover the data.

```
BEGIN TRANSACTION;

BEGIN  TRY
        SQL query
END TRY

BEGIN CATCH
        ROLLBACK
END CATCH
```

CURSORS - it is used with the SP, triggers, etc to perform the row level changes or process data sequentially. It is used to retrieve and update data one at a time.

2 types of cursors: Implicit and Explicit.

IMPLICIT Cursor - it is created automatically by dbms and handled by triggers and sp and used for simple CRUD operations.

EXPLICIT Cursor - it is a more advanced type and provide more options to handle complex operations. Developers use DECLARE, OPEN, FETCH, CLOSE to iterate row by row. It is good to use when row level processing is required and can't be done with single SQL statement.

FILTERS - it is used to filter the data based on a condition using WHERE keyword.

```
SELECT * FROM Employees
WHERE Department = 'HR' AND Salary > 50000;
```

METADATA - it simply means saying information of a data like database, table, schema, data, db objects, etc.

Eg. - A library wants to maintain a catalog of books in its collection. Each book is represented by a metadata record that includes various pieces of information about the book.

Title, Author, Publication, Genre, Description

- This metadata allows the library's catalog system to manage and organize its collection effectively. Users can search for books by title, author, genre, or other criteria. They can also read the book's description to determine if it aligns with their interests.

- The metadata helps library staff and patrons locate the book efficiently, understand its content, and manage the library's collection by tracking details about each book.

In database terms providing information about tables columns like title, description, genre, publishing year, etc.

UDTT (User Defined Table Type) - it is a user defined table structure which contains no data just the table structure and columns details like field names, its data type and constraints. It is used as a parameter for SP and functions its just a prototype or a template.

```
CREATE TYPE EmployeeTableType AS TABLE
(
   EmployeeID INT,
   FirstName NVARCHAR(50),
   LastName NVARCHAR(50),
   Salary MONEY
);

-- Create a stored procedure that uses the UDTT as a parameter
CREATE PROCEDURE InsertEmployees
   @Employees dbo.EmployeeTableType READONLY
AS
BEGIN
   INSERT INTO EmployeeData (EmployeeID, FirstName, LastName, Salary)
   SELECT EmployeeID, FirstName, LastName, Salary
   FROM @Employees;
END;
```

Inshort it is used to send whole table data as in parameter so it is used for bulk data operations and validation.

MERGE - it is basically used for performing operations on data warehouses in a synchronous manner from different or multiple locations i.e, occurring at the same time. We can perform operations like INSERT, UPDATE, DELETE in single go .

For example if we have to make some modification in some data in a warehouse from different sources then we can perform this MERGE operation
It will insert new rows in the target table if it dont exist or if it matches the condition then it will update the data, and if it does not match with the source table then it will be deleted from the target table.

TVP (Table Valued Parameter) - In TVP whole table object is send as a parameter to functions and SP for performing operations. Majorly they are used for bulk operations where lot of data in form rows is used to perform some query and merging but directly it is not affecting the real table its READ ONLY so its useful for such scenarios where huge dataset is used for some operation but at the same time performance should also be high then TVP is best option

Ex- in our previous example of UDTT when we created a UDTT we created a SP to use it as a datatype here we will create a TVP and UDTT as its datatype then use that TVP in the SP for performing operations .

BULK DATA INSERT - it is used to insert the dataset in bulk from a source file in local PC or somewhere else and insert into the target table. It should match the conditions of the table like names, datatypes, constraints, etc. then only it will be matched and added.

BULK INSERT Users
FROM ‘file path’’

IF EXISTS - it is used to check whether a particular object or data exist in the table or not before doing any other operations, so it comes with a another sub query to check that whether exists or not

```
IF EXISTS (SELECT * FROM Employees WHERE EmployeeID = 123)
BEGIN
   PRINT 'Employee exists.';
```

```
END
ELSE
BEGIN
   PRINT 'Employee does not exist.';
END
```

JOIN HINTS - it is used as a last step or solution to be used by experienced developers or DBA to modify the joins for query optimization else sql query optimizer selects the best execution plan by default to make it optimise. But when we want to override the default behaviour we use join hints.

Join hint types : 4
Loop, Hash, Remote, Merge.

LOOP - it is used when one of the table is smaller in size as compared to another. Inshort it is used on small datasets.

HASH - it is used for large datasets where both the input sets are similar in size and unsorted so for that hash join hint will be efficient.

REMOTE - it is used when the input sets are on the different remote linked servers and need to be joined.

MERGE - it is generally used for merging 2 tables which can be large input sets in size and it should be in a pre-sorted manner for efficiency. It is used for equi-joins.

Temp Local VS Temp Global Table - temp local table is persisted until the session time is there of a SP, trigger or any other object so when it completes temp local table is dropped.
Whereas Global table is persisted until the db connection is there when it is disconnected or terminated then it will also drop off.

Global - ##
CREATE TABLE ##TableName (Column1 DataType, Column2 DataType);

Local - #
CREATE TABLE #TableName (Column1 DataType, Column2 DataType);

Temp Calculations - It is used to make temporary calculations or operations on the temporary tables or CTE which is used as intermediate results.

ORM (object relational mapping) - it is a way to map the objects of database with the backend part simply doing from backend or code first approach not from the ssms.

Scope Identity - scope identity is basically used for the same scope of operations for the same batch and it is updated when a last new row is inserted in the same session

@@IDENTITY is also used for the same purpose that is updated when a last row is inserted but it can also work when any SP or trigger is hitted from any other session.

SECURITY - protecting the database objects from sql injections and attacks by using some good code optimisation practices and following methods:

Encryption
Parameterized Queries
Access Control
Firewall On
Authentication
Backup and recovery
Databse auditing

OPTIMIZATION - we can optimize it using following ways

Indexing
Normalization
Denormalization
Query optimization
Caching
Partitioning
SP
Compression
Load Balacing
Database design review