**#1**

**Server Authentication**

* Windows Authentication
* SQL Authentication

**#2**

**System Database**

1. **Master**
2. **Model**
3. **Msdb**
4. **Tempdb**

**#3**

**Whenever we create a database on SQL Server, it will generate two database files are**

1. .mdf (master data file)
2. .ldf(Log data file)

**#4**

**Types in Database**

1. **Interger** Data Type [TinyInt(1), SmallInt(2), Int(4), BigInt(8)]
2. **Decimal** Data Type [Decimal (P, S), Numeric (P, S)]
3. **Money** (SmallMoney(4), Money(8))
4. **DateTime** (Date(yy/mm/dd), Time(hh/mm/ss.ms), DateTime)
5. **Character**

* **Char**(n) (1 byte) (1-8000) (non-unicode)
* **VarChar**(n/max) (1 byte) (1-8000) (non-unicode)
* **Text** (It is same as Varchar(max) data type)
* **NChar**(n) (2 byte) (1-4000) (unicode)
* **Nvarchar**(n/max) (2 byte) (1-4000) (unicode)
* **Ntext**(It is same as nvarchar(max))

1. **Binary**

* **Binary**(n) (1-8000)
* **VarBinary**(n/max) (1-8000)
* **Image** (it is same as varbinary(max))

**#5**

Structured Query Language is divided into 4 Sub Language

1. **DDL** (Data Definition Language) [Drop, SP\_Rename, Create, Alter, Truncate]
2. **DML** (Data Manipulation Language) [Select, Update, Delete, Insert]
3. **TCL** (Transaction Control Language) [Commit, RollBack, Save Transaction]
4. **DCL** (Data Control Language) [Grant, Revoke]

**#6**

**DDL (Data Definition Language)**

This Contains 5 commands

1. **Create**
2. **Alter**
3. **SP**\_**Rename**
4. **Truncate**
5. **Drop**

#Query:



**#7**

**Rules for Creating Table in DB**

* Table name must be unique
* Column names should not repeat
* Table name never starts with number or special char
* No space in column or table name
* Every Object can contain only (1-128) characters
* Max no of columns can be 1024

#Query:



**#8**

**DML** (Data Manipulation Language)

This contains 4 commands

1. **Insert**
2. **Update**
3. **Delete**
4. **Select**

#Query:

****

**#9**

**Difference between Truncate and Delete**

|  |  |
| --- | --- |
| **Truncate** | **Delete** |
| DDL | DML |
| Permanent | Temporary |
| Specific is not possible | Specific possible |
| No "Where" | "Where" |
| Can't Rollback | Can Rollback |
| Reset the identity value | Don't Reset the identity value |

**#10**

Using select command we can retrieve the data from the table in 3 ways

1. Projection (Retrieving data from specific column is known as projection)
2. Selection (Retrieving data from condition is known as selection)
3. Joins

**#11**

**Where, Alias, Identity, Distinct**

**#12**

**Built in Functions in SQL (System Functions)**

* **Aggregate Function**
* Configuration Function
* Curser Function
* **Date and Time Function**
* **Mathematical Function**
* Metadata Function
* Other Function
* Hierarchy Id Function
* Rowset Function
* Security Function
* String Function
* System statistical function
* Text and Image Function

**#13**

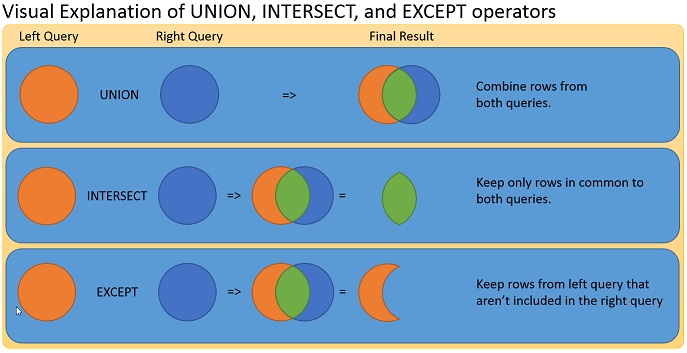
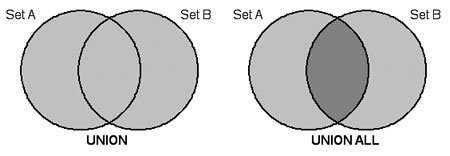
**Operators In SQL**

1. Assignment (=)
2. Arithmetic (+, -, /, %)
3. Comparison (>, <, >=, <=, != (or <>), !<, !>)
4. Logical (And, OR, Not, Between, Not Between, Like, Not Like, IN, Not IN, Exists, Not Exists, Any, All, Some)
5. Set (Union (non-duplicate), Union All(duplicate), Intersect, Except)

#Query:



**# Difference between Union and Union all**



**#14**

**Clauses In SQL**

1. Where (Used to Filter)
2. Order By (Used to sort or arrange data)
3. TOP N Clause (fetch top N)
4. Group By (to arrange similar data into groups)
5. Having (Filtering and restring the data)

#Query:



**#15**

**Difference between where and having**

|  |  |
| --- | --- |
| **Where** | **Having** |
| Used to restrict the records before grouping | Used to restrict the records after grouping |
| Can apply without group by clause | Can’t apply without group by clause |

**#16**

**Syntax to create table from existing table**

* Select \* into <NewTable> from <OldTable>
* Select EmpID, Name into <NewTable> from <OldTable>
* Select \* into Dummy\_Emp from employee where 1=2 (Dummy Table without data in it)

**#17**

**Copying data from one existing table to another table**

* Insert into <DummyTable> select \* from <TableName>

**#18**

**Constraint in SQL**

**Why Constraint in SQL:**

* Constraint is using to restrict the insertion of unwanted data in any columns.
* We can create single or multiple columns of any table

Types of Constraint

1. Unique Key
2. Not Null
3. Check
4. Primary Key
5. Foreign Key

#Query:



**#19**

1. **On Update cascade**,
2. **On Delete Cascade**

**#20**

**JOINS IN SQL**

Joins can be classified into the following types

1. **EQUI JOIN**
2. **INNER JOIN** (Inner join return only those records that match in both table)
3. **OUTER JOIN** (It will retrieve or get matching data from both table as well as UN matching data from left hand side table)
4. **LEFT OUTER JOIN** (It will retrieve or get matching data from both table as well as UN matching data from right hand side table)
5. **RIGHT OUTER JOIN** (It will retrieve or get matching data from both table as well as UN matching data from left hand side table plus right-hand side table also.)
6. **FULL OUTER JOIN** (If we join tables with any condition other than equality condition then we call as a non Equi join.)
7. **NON EQUI JOIN** (Joining a table by itself is known as self-join. When we have some relation between the columns within the same table then we use self-join.)
8. **SELF JOIN**
9. **CROSS JOIN** (Cross join is used to join more than two tables without any condition we call as a cross join. In cross join each row of the first table join with each row of the second table.)
10. **NATURAL JOIN** (It is used to avoid duplicate column from the tables)



#Query:



**#21**

**TCL (TRANSACTION CONTROLL LANGUAGE)**

**TRANSACTION**: A transaction is a unit of work that is performed against a database or set of statement (Insert, Update and Delete) which should be executed as one unit.

To manage transaction, we have provided with transaction control language that provides a command like

1. **BEGIN TRANSACTION** (Begin Transaction command is used to start the transaction)
2. **COMMIT** (Commit command is used to end the transaction and save the data permanent part of the database)
3. **ROLLBACK** (Rollback command is used to undo the transactions and gets back to the initial state where transaction started.)
4. **SAVE** **POINT** (Save point is used for dividing (or) breaking a transaction into multiple units)

#Query:



**#22**

**Sub Query**: A select query contains another select query is called sub Query

**Syntax**: select \* from <Table Name> where (condition) (select \* from…….. (Select \* from….. (select \* from…….)));

Disadvantage of SubQuery is its Execution is slow

**#23**

**INDEXES IN SQL:**

* **Why** - One of the most important routes to high performance in a SQL Server database is the index. Indexes speed up the querying process by providing quickly access to rows in the data tables
* Index is a database object which is used for the quick retrieving of the data from the table
* An index contains keys built from one or more columns in the table and map to the storage location of the specified data.
* By using indexes, we can save time and can improve the performance of database queries and applications
* When we create an indexes on any column, SQL server internally maintain a separate table called index table. So that whenever user trying to retrieve the data from existing table depends on index table SQL server directly go to the table and retrieve required data very quickly.
* In a table we can use max 250 indexes
* Two Types of Indexes

1. Clustered
2. Non-Clustered

* **Clustered**:
* A table can have only 1 clustered index on it
* The only time the data rows in a table are stored in sorted (ascending order only) order structure is when the table contains a clustered index.
* **Non-Clustered**
* In a table we can create 249 non-clustered index
* If we dont mention clustered indexes in a table then default is stored as non-clustered indexes.
* **Syntax**: Create Index <Index Name> on <Table Name> (Column Name);

#Query:



**#24**

**Views In SQL:**

* View is database object which is like table but logical.
* We can call it as a logical or virtual table because it does not have a physical existence.
* It is a logical table use to get the required information from the table. View will be created by using select statement and table used for the creation of the view is called as base table.
* View will not store records in it and will not occupy memory space with help of structure existing in it and records will be displayed from table.
* **Views are used for security purpose in databases**
* views restrict the user from viewing certain column and rows means by using view we can apply the restriction
* **Why**? - To protect the data
* View is classified into two types

1. Simple view (Updatable view) (one table)
2. Complex view (Non-Updatable view) (more than one table)

* **Syntax**: create view <view name> as select \* from <table name>

#Query:



**#25**

**SYNONYM**:

* synonym is database object which can be created as an “alias” for any object like table, view, procedure etc.
* If we apply any DML operations on synonym the same operations automatically effected to corresponding base table and vice versa
* If we create a synonym, the synonym will be created on entire table. It is not possible to create the synonym on partial table.
* **Syntax**: Create synonym <synonym name> for <object name>
* **Syntax to drop a synonym**: Drop synonym <synonym name>

**#26**

**T/SQL Programming (Transact Structure Query Language)**

* SQL language will not provide reusability facilities where as T/SQL language will provide reusability facilities by defining objects such as Procedures and Functions
* T/SQL Program blocks can be divided into two types. Those are

1. Anonymous Blocks
2. Sub-Program Blocks

* **Anonymous**: - Unnamed block
* **Declare variable**
* **Syntax**: Declare @<var> [as] <datatype> [size]......
* **Assigning value to variable**
* **Syntax**: Set @<var>=<value>
* **Print value**
* **Syntax**: print @<var>
* Conditional statements:

1. IF-ELSE
2. CASE
3. While

* **Sub**-**Program**: Named block of code

1. Stored Procedure
2. Functions

* **Stored** **procedure**:
* A stored procedure is a database object which contains precompiled queries.
* Stored Procedures are a block of code designed to perform a task whenever we called.
* Advantages of SP?
* As this is precompiled burden on DB will reduce
* Application performance will improve
* code reusability
* **Create**:

**Syntax**:

Create Procedure <SP\_Name>

(Passing parameters)

As

Begin

<stmts>

End

* **Call:**

Syntax: Exec <SP\_Name>

* **Function**:
* Function is a block of code similar to a stored procedure which is also used to perform an action and returns result as a value.
* Function can be divided into two types

1. **Scalar-Valued Function**

* In this case we can return an attribute datatype as an output from the function.
* **Create Syntax**:

Create Function <Function Name> (@parameter <Data Type> [size])

Returns <return attribute data type>

As

Begin

<Function Body>

Return <return attribute name>

End

* **Call** **Syntax**:

Select dbo.<Function Name> (value)

1. **Table**-**Valued** **Function**

* In this case we can return a table as an output from the function.
* **Create Syntax**:

Create Function <Function Name> (@parameter <Data Type> [size])

Returns <Table>

As

Return <return select statement>

* **Call** **Syntax**:

select \* from functionname(value)

* **Drop** **Syntax**: Drop Function <Function \_Name>
* **Difference between Stored Procedure and Function**

|  |  |
| --- | --- |
| **Function** | **Stored Procedure** |
| Returns value | Don't return value |
| Can't have output parameter | Can have output parameter |
| Use to perform select only | Use for all DML operations |
| Call with **Select** | Call with **Execute** |

**#27**

**Triggers**

* A trigger is a special type of procedure that will used to provide restrict on the tables when a language events executed.
* Two Types of Triggers

1. **DML**
2. DML triggers execute when the user tries to modify or change data through data manipulation language events. Those are Inserting, Update and Delete statements on the table.
3. DML triggers can be used to enforce business rules and data integrity. With the help of a DML trigger we can enforce integrity which cannot be done with constraints.
4. **Create Syntax**:

Create Trigger <Trigger Name> on <table Name>

For [Insert, Update, Delete]

AS

Begin

<Statements>

End

1. **Drop** **Syntax**:

Drop <Trigger> <Trigger Name>

1. **DDL**
2. DDL triggers fire in response to a data definition language event like create, Alter, drop etc. A DDL triggers is a special type of procedure that executes in response to a server scoped or database scoped events.
3. Create Syntax:

Create Trigger <Trigger Name> on database after <Event type>

As

Begin

<Statements>

End

1. Drop Syntax:

Drop <Trigger> <Trigger Name> on Database

#Query:



**#28**

**Magic** **Tables**:

* SQL Server allows you to define a Magic Table. Magic Tables are invisible tables or virtual tables. You can see them only with the help Triggers in SQL Server.
* Magic table are those tables which allows you to hold inserted, deleted, updated values during DML operations on a Table in SQL server
* Basically, there are two types of magic table in SQL Server

1. Inserted Magic Table
2. Deleted Magic Table

* **Inserted Magic Table**: Whenever you insert a record on that table, that record will be shown in the Inserted Magic table.
* **Deleted Magic Table**: Whenever you delete a record on that table, that record will be shown in the Inserted Magic table.
* Update the record in Table: For update record we need to use both Magic tables one shows the inserted and one shows the deleted table

#Query:



**#29**

**Exception** **Handling**:

* We handle errors of a program both in a programming language as well as databases also. whereas handling an error in a programming language needs stopping the abnormal termination and allowing the statements which are not related with the error to execute whereas handling as error in sql server means stopping the execution of statements which are related with the error
* Handling errors in SQL Server: we can handle errors with TRY\_CATCH

Begin Try

<stmts>

End Try

Begin Catch

<stmts>

End Catch

* Error Message (): It is used to display the information about the error occurred.

**#30**

**Cursor:**

* Cursor is a memory location for storing database tables.
* cursor is a temporary work area allotted to the client at server when a SQL statement is executed.
* A cursor contains information on a select statement and the rows of data accessed by it.
* This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the Result set.
* There are Two Types of Curser

1. Implicit Cursors
2. Explicit Cursors

* **Implicit** **Cursor**: These cursors will be created by SQL server by default when select statements will show records in the table as set or result set.
* **Explicit Cursor**: When user can create a memory location to store the tables then it is called as Explicit Cursors. These cursors will access the records in the table record by record or one by one only. Whenever we want to go for record by record manipulation then explicit cursors will be used.
* **Steps to Create Cursor**

1. **Declare**: Declare <CursorName> cursor for <select statement>
2. **Opening**: Open <CursorName>
3. **Fetch Data from Cursor**: Fetch first/last/next/prior/absolute n/relative n from <CursorName> into variable
4. **Closing**: Close <Cursor>
5. **De**-**allocate**: Deallocate <CursorName>

* **Forward only & static Cursors**: If a cursor is declared as forward only it allows you to navigate only to the next records in sequential order and more over it supports only a singleton fashion method that is fetch next(one-by-one) where as a scroll cursor allows you to navigate/fetch Bidirectional that is top- bottom or bottom-top also. And it supports six different fetch methods are Fetch Next, Fetch First, Fetch Last, Fetch Prior, Fetch Absolute, Fetch Relative.
* **Static & Dynamic Cursors**: If a cursor is declared as static after opening the cursor any modifications that are performed to the data in the table will not be reflected into cursor so the cursor contains old values only in it.

**#31**

**Data Control Language:**

* **Authentication**: Authentication is a process of verifying the credentials of a user to login into the system.
* **Authorization**: Authorization is process of verifying whether the user as permissions to perform any operation on the database.
* **Data Control Language**: DCL commands are used to enforce database security in multiple users’ database environment. These are two types.

1. GRANT
2. REVOKE

* ***GRANT***: Grant command is used for giving a privilege or permission for a user to perform operations on the database.
* **Syntax**:

GRANT <Privilege Name> on <object name>

To {User} [With GRANT OPTION]

* **Privilege Name**: Used to granted permission to the users for some rights are ALL, EXECUTE and SELECT.
* **Object** **Name**: It is the name of database objects like Table, Views and Stored Procedure etc.
* **User**: Used for to whom an access rights is being granted.
* **With** **Grant** **Option**: Allows a user to grant access rights to other users.
* ***REVOKE***: Revoke command removes user access rights / privileges to the database OR taking back the permission that is given to a user.
* **Syntax**: Revoke <privilege name> on <object name > from {user}

**#32**

**Normalization:**

* Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process are,

1. Eliminating redundant data (for example, storing the same data in more than one table) and
2. Ensuring data dependencies make sense (only storing related data in a table).

* Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.
* There are several benefits for using Normalization in Database.

1. Eliminate data redundancy
2. Improve performance
3. Query optimization
4. Faster update due to a smaller number of columns in one table
5. Index improvement

* There are different types of Normalizations form available in the Database.

1. 1NF (First Normal Form)
2. 2NF
3. 3NF
4. 4NF
5. 5NF

**#Interview Write A Query:**

WAQ for finding employee with Nth highest salary



WAQ for deleting duplicate records

