SQL Interview questions

**Commends in SQL**

---------------------------

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.

Examples of DDL commands:Create ,Drop ,Alter, Truncate ,Rename.

Alter to add column: Alter table <table Name> add <column name> datatype

Drop column: Alter table <table name> drop column <column Name>.

Change data type: Alter table <table Nmae> alter column <column Name><datatype>

DML(Data Manipulation Language) :The SQL commands that deals with the manipulation of data present in database

Examples of DML commands :Select, Insert, Update, Delete.

DCL(Data Control Language) : DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands:

GRANT-gives user’s access privileges to database.

REVOKE-withdraw user’s access privileges given by using the GRANT command

Grant: Grant create table,create view to <role name>.

Revoke; Revoke create table from <role name>

TCL(transaction Control Language) **:** TCL commands deals with the [transaction within the database](https://www.geeksforgeeks.org/sql-transactions/).

Examples of TCL commands:

COMMIT– commits a Transaction.

[ROLLBACK](https://www.geeksforgeeks.org/sql-transactions/)– rollbacks a transaction in case of any error occurs.

SAVEPOINT–sets a savepoint within a transaction.

**Functions:**

**--------------**

A function is a database object which accepts one parameter.

Types:: User defined and system defined.

User defined:

Table Valued Functions: Which returns the table:

Create function <function name>

Returns table

As

Begin

return (selete \*from emp)

end

Scalar Valued Functions:scalar functions operate against a single value and return a single value.

Example:Upper(string)

Aggregation functions/system functions: operates on multiple valyes returns single value:

Example:SUM(<colum name>)

**Views in sql**

**-----------------**

A view is actually a composition of a table. It is a just select statement. **Views** are used for security purposes because they provide encapsulation of the name of the table

If main table update view automatically updated.

We cant alter view . retrieving is fast in views because they store temporary data.

Syntax:: CREATE VIEW <view name> AS SELECT empid,emp\_name FROM employee;

**INDEXS in SQL:**

**--------------------**

Indexs are used to improve the performance in reqtriving.

Syntax:: create index <index name> on <table name> (<columns names>).

Types:

Clustred index: It determines the physical order of data and sorts the data. A primary key creates default clustered idex on column.

But one clustred index has possible for more columns called composit clustred index.

Create index <index name> on table <table name>(column1 ASC, column2 desc)

Non-Clustred index: Here the index will creates at one point and table data will be in another point.

The index has pointer which referece the tables data. Like text bboks

Create NONClustred index <index name> on <table name> (columns)

**Triggeres:**

**--------------**

A trigger is a special kind of Stored Procedure or stored program that is automatically fired or executed when some event (insert, delete and update) occurs.

Types:

DML Triggers:

After triggers/For triggers : they will fire after insert update insert will fired.

CREATE TRIGGER <trigger name>

ON INV\_<table name>

AFTER update/delete/insert

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

declare @hostname varchar(30)

select @hostname = hostname FROM sys.sysprocesses

WHERE spid = @@SPID

Print 'INV\_ITEM\_WAREHOUSE is updated' + ' At ' +

cast(Getdate() as nvarchar(20))+' Using Hostname

'+@hostname

-- Insert statements for trigger here

END

GO

Instead off triggers: they will fired before the insert update delete

CREATE TRIGGER <trigger name>

ON INV\_<table name>

INSTEAD OF update/delete/insert

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

declare @hostname varchar(30)

select @hostname = hostname FROM sys.sysprocesses

WHERE spid = @@SPID

Print 'INV\_ITEM\_WAREHOUSE is updated' + ' At ' +

cast(Getdate() as nvarchar(20))+' Using Hostname

'+@hostname

-- Insert statements for trigger here

END

GO

DDl triggers: this triggers will fire on ddl commends.

CREATE/ALTER TRIGGER <trigger name>

ON database

For CREATE\_TABLE,ALTER\_TABLE,DROP\_TABLE

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

declare @hostname varchar(30)

select @hostname = hostname FROM sys.sysprocesses

WHERE spid = @@SPID

Print 'INV\_ITEM\_WAREHOUSE is updated' + ' At ' +

cast(Getdate() as nvarchar(20))+' Using Hostname

'+@hostname

-- Insert statements for trigger here

END

GO

LOGON triggers:

This triggers will fires in response of login events. Used fro tracking activits , restricts

to login for sql servers, limits number of sessions for specific login.

Query to limits user session for 4 times::::::::::

ALTER TRIGGER <trigger name>

ON ALL SERVER

FOR LOGON

AS

BEGIN

DECLARE @logname varchar(100)

SET @logname =ORIGINAL\_LOGIN()

PRINT(@logname)

IF(SELECT COUNT(\*) FROM sys.dm\_exec\_Sessions WHERE

original\_login\_name=@logname and is\_user\_process=1)>=3

BEGIN

PRINT 'FOURTH CONNECTION

ROLLBACK

END

END

---is\_user\_process tells us where connection is created by user process

DISABLE TRIGGER <trigger-name> ON ALL SERVER --------------only for logon triggers

Find the rank of Below set:

---------------------------

{25,25,10,11,12,15,15,19} out put {1,1,3,4,5,6,6,8}

Find the Dens of Below set:

---------------------------

{25,25,10,11,12,15,15,19} out put {1,1,2,3,4,5,5,6}

Print a column in folloeing path ‘namw1,name2,mae3,name4…’

-----------------------------------------------------------

Select <column\_ame> from table for xml path(‘’)

CTE(Common table expression)

------------------------------

A cet is a temparery result set that can be a reference with in a select insert delete statements. Used for recursive query

Delete duplicates using cte::::

with temp(rank1,id ,fname,lname)

as

(

select row\_number() over ( partition by ID, FirstName, LastName order

by ID, FirstName, LastName ) , \*

from duplicateTest

)

select \* from temp where rank1 > 1;

To find nth higest salary

WITH CTE AS

(

SELECT Name, Year, Sales,

RN = ROW\_NUMBER() OVER (ORDER BY Sales DESC)

FROM employee

)

SELECT Name, Year, Sales

FROM CTE

WHERE RN = 2

To find nth higest salary based on department

WITH CTE AS

(

SELECT Name, Year, Sales,

DN = ROW\_NUMBER OVER (PARTITION BY Year ORDER BY Sales DESC)

FROM employee

)

SELECT Name, Year, Sales,DN

FROM CTE

WHERE DN = 1

select row\_number() over ( partition by ID, FirstName, LastName order by ID, FirstName, LastName ),\* from duplicateTest

Recursion using cte: Recursion uses union all

with weekdy (n,wekday) as

(

select 0,DATENAME(dw,0)

union all

select n+1,DATENAME(dw,n+1) from weekdy where n<6

)

select wekday from weekdy

**backeup database**

---------------

BACKUP DATABASE <Name>TO DISK='<File Path>'

RESTORE DATABASE AdventureWorks FROM DISK = 'C:\AdventureWorks.BAK'

**Stored Procedure**

**----------------**

**A stored procedure is a group of T-Sql statements used for re-usability**

**Syntax:**

**create procedure <procedure name>[params] as**

**Begin**

**End**

**Cursores**

**-----------**

**It is a database object to retrieve data from a result set one row at a time. It is useful when we want to manipulate the record of a table in a singleton method**

**Types:::**

**Implicit Cursor**

**These types of cursors are generated and used by the system during the manipulation of a DML query (INSERT, UPDATE and DELETE). An implicit cursor is also generated by the system when a single row is selected by a SELECT command.**

**Explicit Cursor**

**This type of cursor is generated by the user using a SELECT command. An explicit cursor contains more than one row, but only one row can be processed at a time. An explicit cursor moves one by one over the records. An explicit cursor uses a pointer that holds the record of a row. After fetching a row, the cursor pointer moves to the next row.**

**Main components of Cursors**

**Each cursor contains the followings 5 parts,**

1. **Declare Cursor: In this part we declare variables and return a set of values.**
2. **Open: This is the entering part of the cursor.**
3. **Fetch: Used to retrieve the data row by row from a cursor.**
4. **Close: This is an exit part of the cursor and used to close a cursor.**
5. **Deallocate: In this part we delete the cursor definition and release all the system resources associated with the cursor**

**Sytax:::**

**DECLARE ItemCursor CURSOR**

**FOR**

**SELECT CAMPAIGN\_NAME,DESCRIPTION,BUDGETED\_COST**

**FROM CRM\_CAMPAIGN;**

**--Open cursor**

**OPEN ItemCursor**

**--FETCH THE RECORD INTO THE VARIABLES.**

**FETCH NEXT FROM ItemCursor INTO**

**@unitPrice, @quantity, @amount**

**WHILE @@FETCH\_STATUS = 0**

**BEGIN**

**--PRINT CURRENT RECORD.**

**PRINT CONCAT(@unitPrice, SPACE(0.5),@quantity,SPACE(0.5), @amount)**

**--FETCH THE NEXT RECORD INTO THE VARIABLES.**

**FETCH NEXT FROM ItemCursor INTO**

**@unitPrice, @quantity, @amount**

**END**

**--CLOSE THE CURSOR.**

**CLOSE ItemCursor**

**DEALLOCATE ItemCursor**

**GO;**

**DEALLOCATE ItemCursor**

**Types of cursors**

**Microsoft SQL Server supports the following 4 types of cursors.**

1. ***STATIC CURSOR*  
   A static cursor populates the result set during cursor creation and the query result is cached for the lifetime of the cursor. A static cursor can move forward and backward.**
2. ***FAST\_FORWARD*This is the default type of cursor. It is identical to the static except that you can only scroll forward.**
3. ***DYNAMIC*In a dynamic cursor, additions and deletions are visible for others in the data source while the cursor is open.**
4. ***KEYSET*This is similar to a dynamic cursor except we can't see records others add. If another user deletes a record, it is inaccessible from our record set**

## Cursor Scope

**Microsoft SQL Server supports the GLOBAL and LOCAL keywords on the DECLARE CURSOR statement to define the scope of the cursor name.**

1. **GLOBAL - specifies that the cursor name is global to the connection.**
2. **LOCAL - specifies that the cursor name is local to the Stored Procedure, trigger or query that holds the cursor.**

**Data Fetch Option in Cursors**

**Microsoft SQL Server supports the following two fetch options for data:**

1. **FORWARD\_ONLY - Specifies that the cursor can only be scrolled from the first to the last row.**
2. **SCROLL - It provides 6 options to fetch the data (FIRST, LAST, PRIOR, NEXT, RELATIVE and ABSOLUTE)**

**Types of Locks**

**--------------**

**Locking is the process by which a DBMS restricts access to a row in a multi-user environment. When a row or column is exclusively locked, other users are not permitted to access the locked data until the lock is released. It is used for data integrity. This ensures that two users cannot simultaneously update the same column in a row.**

**Microsoft SQL Server supports the following three types of Locks.**

1. ***READ ONLY*Specifies that the cursor cannot be updated.**
2. ***SCROLL\_LOCKS*Provides data integrity into the cursor. It specifies that the cursor will lock the rows as they are read into the cursor to ensure that updates or deletes made using the cursor will succeed.**
3. ***OPTIMISTIC*Specifies that the cursor does not lock rows as they are read into the cursor. So, the updates or deletes made using the cursor will not succeed if the row has been updated outside the cursor.**

**Pivot**

**------**

**PIVOT is used when we want to transfer data from row level to column level**

**<syntax>**

**SELECT <non-pivoted column>, ---------- first select pivot columns**

**<list of pivoted column>**

**FROM**

**(<SELECT query to produces the data>)**

**AS <alias name> ---------------- select data from normal table into pivote**

**PIVOT**

**(**

**<aggregation function>(<column name>)**

**FOR**

**[<column name that become column headers>] ---------------------- use aggregations to form pivot**

**IN ( [list of pivoted columns])**

**) AS <alias name for pivot table>**

**example**

**--------**

**SELECT Year, Pankaj,Rahul,Sandeep FROM**

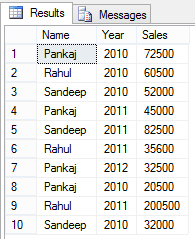
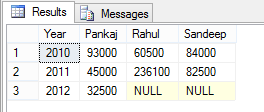
**(SELECT Name, [Year] , Sales FROM Employee )Tab1**

**PIVOT**

**(**

**SUM(Sales) FOR Name IN (Pankaj,Rahul,Sandeep)) AS Tab2**

**ORDER BY [Tab2].[Year]**

 out put

**Merge in sql**

**---------------**

The **MERGE** statement is used to make changes in one table based on values matched from anther. It can be used to combine insert, update, and delete operations into one statement.

1. MERGE [**AS** TARGET]
2. USING [**AS** SOURCE]
3. **ON**
4. [**WHEN** MATCHED
5. **THEN** ]
6. [**WHEN** NOT MATCHED [**BY** TARGET]
7. **THEN** ]
8. [**WHEN** NOT MATCHED **BY** SOURCE
9. **THEN** ];

Merge <table> t1

Using <table> t2

On t1.id=t2.id

**Joins**

**-----**

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

**10. What are the types of join and explain each?**

There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

* **Inner Join.**

Inner join return rows when there is at least one match of rows between the tables.

* **Right Join.**

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table.

* **Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

* **Full Join.**

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

**How can you create an empty table from an existing table?**

**-----------------------------------------------------------------------------**

**Example will be -.**

**Select \* into studentcopy from student where 1=2**

**How to fetch common records from two tables?**

**--------------------------------------------------------------------**

**Common records result set can be achieved by -.**

**Select studentID from student.**

**INTERSECT**

**Select StudentID from Exam**

**How to fetch alternate records from a table?**

**---------------------------------------------------------**

**Records can be fetched for both Odd and Even row numbers -.**

**To display even numbers-.**

**Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0**

**To display odd numbers-.**

**Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1**

**How to select unique records from a table?**

**------------------------------------------**

**Select unique records from a table by using DISTINCT keyword.**

**Select DISTINCT StudentID, StudentName from Student.**

**. Which operator is used in query for pattern matching?**

**---------------------------------------------------------------------------**

**LIKE operator is used for pattern matching, and it can be used as -.**

**% - Matches zero or more characters.**

**\_(Underscore) – Matching exactly one character.**

**Example -.**

**Select \* from Student where studentname like 'a%'**

**Select \* from Student where studentname like 'ami\_'**

**What is the difference between CHAR and VARCHAR2 datatype in SQL?**

**-------------------------------------------------------------------**

Both Char and Varchar2 are used for characters datatype but varchar2 is used for character strings of variable length whereas Char is used for strings of fixed length. For example, char(10) can only store 10 characters and will not be able to store a string of any other length whereas varchar2(10) can store any length i.e 6,8,2 in this variable.

**What are STUFF and REPLACE function?**

**------------------------------------**

**STUFF Function**: This function is used to overwrite existing character or inserts a string into another string. Syntax:

STUFF(string\_expression,start, length, replacement\_characters)

where,  
string\_expression: it is the string that will have characters substituted

start: This refers to the starting position  
length: It refers to the number of characters in the string which are substituted.

replacement\_string: They are the new characters which are injected in the string.

## What do you mean by Collation?

Collation is defined as a set of rules that determine how data can be sorted as well as compared. Character data is sorted using the rules that define the correct character sequence along with options for specifying case-sensitivity, character width etc.

## Q60. What are the different types of Collation Sensitivity?

Following are the different types of collation sensitivity:

* Case Sensitivity: A and a and B and b.
* Kana Sensitivity: Japanese Kana characters.
* Width Sensitivity: Single byte character and double-byte character.
* Accent Sensitivity.

Apart from this SQL Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for a structured training from edureka! Click below to know more.

## Q61. What are Local and Global variables?

### **Local variables:**

These variables can be used or exist only inside the function. These variables are not used or referred by any other function.

### **Global variables:**

These variables are the variables which can be accessed throughout the program. Global variables cannot be created whenever that function is called.

Bulk insert:

---------------

To insert huge data from csv files.

Bulk insert <table name> from ‘<file path>’ with ( rowteriminator=’\n’, fieldterimiator=’,’ , TABLOCK)

TABLOCK—to lock table while data loading to improve prefornance

**OPENROWSET**

**--------------------------------**

**to read large files into SQL Server**

**insert into <tablename> select \*from OPENROWSET(‘add database oled<file path>’,<sheetname’)**

SQL query to get organization hierarchy

-----------------------------------------------------

Goto url <https://www.youtube.com/watch?v=Kd3HTph0Mds>

Here is the problem definition: 1. Employees table contains the following columns a) EmployeeId, b) EmployeeName c) ManagerId

1. If an EmployeeId is passed, the query should list down the entire organization hierarchy i.e who is the manager of the EmployeeId passed and who is managers manager and so on till full hierarchy is listed.
2. For example, Scenario 1: If we pass David's EmployeeId to the query, then it should display the organization hierarchy starting from David.
3. Scenario 2: If we pass Lara's EmployeeId to the query, then it should display the organization hierarchy starting from Lara.

Declare @id int;

set @id =7;

with cte as

(

select EmployeeID,EmployeeName,ManagerID

from Employees where EmployeeID=@id

union all

select Employees.EmployeeID,Employees.EmployeeName,Employees.ManagerID

from Employees join cte on Employees.EmployeeID=cte.ManagerID

)

select c1.EmployeeName,c2.EmployeeName from cte c1 left join cte c2 on c1.ManagerID= c2.EmployeeID