Q1- 1. batch, script and transaction

1- SQL Batch

\*SQL Batch is just that a collection of commands that need to be executed without guaranteed of success or fail.

\*Batch Processing means things are put into queue and it is processed when a certain amount if items is reached, or when a certain period has passed. You can do undo/rollback in this.

\*In BATCH PROCESSING, the bank would just queue xyz's request to deposit amount. The bank would just put your request in queue with all the other requests and process them at the end of the day or when they reach a certain amount.

2-SQL Transaction

\*SQL Transaction is a collection of commands that are guaranteed to succeed or fail totally.Transactions won't complete half the commands and then fail on the rest, if one fails they all fail.

\*Transaction is like real time processing that allows you to rollback/undo changes.

\*In TRANSACTIONS, it's just like the batch, but you have the option to "cancel" it.

Scripts

\*shows you every query and every transactions user did on your databases's objects included(tables, views, columns …etc)

Q2- trigger and stored procedure

Trigger

\*Is a stored procedure that runs automatically when various events happen (eg update, insert, delete)

\*It can execute automatically based on the events

\* It cannot take input as parameter

\* we can't use transaction statements inside a trigger

\*Triggers can not return values, Only fire it's code when a specific event happens.

2- Stored procedures

\*Are a pieces of the code in written in PL/SQL to do some specific task

\* It can be invoked explicitly by the user

\* It can take input as a parameter

\* We can use transaction statements like begin transaction, commit transaction, and rollback inside a stored procedure

\* Stored procedures can return values

Q3-stored procedure and functions

| Functions | Procedures |
| --- | --- |
| A function has a return type and returns a value. | A procedure does not have a return type. But it returns values using the OUT parameters. |
| You cannot use a function with Data Manipulation queries. Only Select queries are allowed in functions. | You can use DML queries such as insert, update, select etc… with procedures. |
| A function does not allow output parameters | A procedure allows both input and output parameters. |
| You cannot manage transactions inside a function. | You can manage transactions inside a procedure. |
| You cannot call stored procedures from a function | You can call a function from a stored procedure. |
| You can call a function using a select statement. | You cannot call a procedure using select statements. |

Q4- drop, truncate and delete statement

TRUNCATE

TRUNCATE Command is a Data Definition Language operation. It is used to remove all the records from a table. It deletes all the records from an existing table but not the table itself. The structure or schema of the table is preserved.

Truncate command marks the table for deallocation. This operation removes all the data from a table bypassing a number of constraints enforced on the table. MySQL does not allow the users to truncate the table which is referenced as FOREIGN KEY in another table.

Syntax is : TRUNCATE TABLE [database\_name.]table\_name

Delete

The DELETE statement in SQL is a Data Manipulation Language(DML) Command. It is used to delete existing records from an existing table. We can delete a single record or multiple records depending on the condition specified in the query.

The conditions are specified in the WHERE clause of the DELETE statement. If we omit the WHERE clause then all of the records will be deleted and the table will be empty.

The DELETE statement scans every row before deleting it. Thus it is slower as compared to TRUNCATE command. If we want to delete all the records of a table, it is preferable to use TRUNCATE in place of DELETE as the former is faster than the latter.

DELETE is a DML Command so it can be rolled back.

The DELETE command returns the number of records that were deleted by its execution.

Syntax: DELETE FROM table\_name [WHERE conditions]

Drop

DROP statement is a Data Definition Language(DDL) Command which is used to delete existing database objects. It can be used to delete databases, tables, views, triggers, etc.

A DROP statement in SQL removes a component from a relational database management system (RDBMS).

DROP is a DDL Command. Objects deleted using DROP are permanently lost and it cannot be rolled back.

Unlike TRUNCATE which only deletes the data of the tables, the DROP command deletes the data of the table as well as removes the entire schema/structure of the table from the database.

Syntax

DROP object object\_name

Q5- select and select into statement

INSERT INTO SELECT vs SELECT INTO

INSERT INTO SELECT and SELECT INTO may be very similar commands but they have some important differences. Every now and again I see people getting confused by the similarities and missing the differences. So here is a quick check list.

Similarities

They look similar (I think this one throws people a lot.)

They are both part of larger commands.

They both insert rows into a table.

Both can be minimally logged under the right circumstances.

Differences

INSERT INTO SELECT inserts into an existing table.

SELECT INTO creates a new table and puts the data in it.

All of the columns in the query must be named so each of the columns in the table will have a name. This is the most common mistake I see for this command.

The data type and nullability come from the source query.

If one of the source columns is an identity column and meets certain conditions (no JOINs in the query for example) then the column in the new table will also be an identity.

Q6.local and global variables

Local Variable is defined as a type of variable declared within programming block or subroutines. It can only be used inside the subroutine or code block in which it is declared. The local variable exists until the block of the function is under execution. After that, it will be destroyed automatically.

Example: public int add(){

int a =4;

int b=5;

return a+b;

}

Global Variable

A Global Variable in the program is a variable defined outside the subroutine or function. It has a global scope means it holds its value throughout the lifetime of the program. Hence, it can be accessed throughout the program by any function defined within the program, unless it is shadowed.

Example:

int a =4;

int b=5;

public int add(){

return a+b;

}

Q7-convert and cast statements

CAST is part of the ANSI-SQL specification; whereas, CONVERT is not.  In fact, CONVERT is SQL implementation specific.

CONVERT differences lie in that that accepts an optional style parameter which is used for formatting.

For example, when converting a DateTime datatype to Varchar, you can specify the resulting date’s format, such as YYYY/MM/DD or MM/DD/YYYY.

Q8-DDL,DML,DCL,DQL and TCL

DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

[CREATE](https://www.geeksforgeeks.org/sql-create/)

[DROP](https://www.geeksforgeeks.org/sql-drop-truncate/)

[ALTER](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)

[TRUNCATE](https://www.geeksforgeeks.org/sql-drop-truncate/)

[COMMENT](https://www.geeksforgeeks.org/sql-comments/)

[RENAME](https://www.geeksforgeeks.org/sql-alter-rename/)

2.DQL (Data Query Language) :

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

Example of DQL:

select

3.DML(Data Manipulation Language): The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

* Insert
* Update
* Delete

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

Examples of DCL commands:

* Grant
* Revoke

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

Examples of TCL commands:

* COMMIT
* [ROLLBACK](https://www.geeksforgeeks.org/sql-transactions/)
* SAVEPOINT
* SET TRANSACTION

Q9-For xml raw and for xml auto

Generating RAW XMLSELECT store.stor\_id AS store\_id, store.stor\_name, store.city, sale.ord\_num, sale.qty FROM Stores store Inner Join Sales sale ON store.stor\_id = sale.stor\_id FOR XML RAW

SELECT store.stor\_id AS store\_id, store.stor\_name, store.city, sale.ord\_num, sale.qty FROM Stores store Inner Join Sales sale ON store.stor\_id = sale.stor\_id FOR XML AUTO, ROOT('ORDERS'), ELEMENTS

Q10-Table valued and multi statement function

Inline table valued function refers to a TVF where the function body just contains one line of select statement. ... Multi-statement table valued function refers to a TVF where it has a return table variable. Inside the function body, there will be statements populating this table variable.

Q11- Varchar(50) and varchar(max)

|  |  |  |
| --- | --- | --- |
|  | VARCHAR | VARCHAR(MAX) |
| Bytes | It consumes 1 byte per character. | It consumes 1 byte per character. |  |
| Data Length | The maximum number of characters for this data type can hold up to 8000 characters by defining as  VARCHAR(8000). | The maximum number of characters for this data type can hold unlimited by defining as NVARCHAR(MAX) MAX indicates that the maximum storage size is (2147483647) 2^31-1 bytes (2 GB). The storage size is the actual length of the data entered + 2 bytes. |  |
| Data  Storage  Type | Used to store variable length value as String. | Used to store variable length value as String. |  |

Q12- Datetime(3), datetime2(7) and datetimeoffset(7)

| Feature | datetimeoffset | datetime2 |
| --- | --- | --- |
| SQL Compliant (ANSI & ISO 8601) | Yes | Yes |
| Date Range | 0001-01-01 through 9999-12-31 | 0001-01-01 through 9999-12-31 |
| Time Range | 00:00:00 through 23:59:59.9999999 | 00:00:00 through 23:59:59.9999999 |
| Character Length | 26 positions minimum 34 maximum | 19 positions minimum 27 maximum |
| Storage Size | 8 to 10 bytes, depending on the precision\*  \* Plus 1 byte to store the precision | 6 to 8 bytes, depending on the precision\*  \* Plus 1 byte to store the precision |

Q13-Default instance and named instance

A named instance is identified by the network name of the computer plus the instance name that you specify during installation. The client must specify both the server name and the instance name when connecting.

By default, SQL Server installs in the default instance unless you specify an instance name. SQL Server Express, however, always installs in a named instance unless you force a default installation during setup.

Q-14 SQL and windows Authentication

Windows authentication means the account resides in Active Directory for the Domain. SQL Server knows to check AD to see if the account is active, password works, and then checks what level of permissions are granted to the single SQL server instance when using this account.

SQL Server Authentication means the account resides in the SQL server master database but nowhere on the Domain. The username and password are stored in the master database. If this account needs to access more than 1 SQL Server instance, then it has to be created on each instance.

Q15-Clustered and non-clustered index

A clustered index defines the order in which data is physically stored in a table. Table data can be sorted in only way, therefore, there can be only one clustered index per table. In SQL Server, the primary key constraint automatically creates a clustered index on that particular column.

A non-clustered index doesn’t sort the physical data inside the table. In fact, a non-clustered index is stored at one place and table data is stored in another place. This is similar to a textbook where the book content is located in one place and the index is located in another. This allows for more than one non-clustered index per table.

Q16- Group by rollup and group by cube

The GROUP BY clause is used to group the results of aggregate functions according to a specified column. However, the GROUP BY clause doesn’t perform aggregate operations on multiple levels of a hierarchy.

The CUBE operator is also used in combination with the GROUP BY clause, however the CUBE operator produces results by generating all combinations of columns specified in the GROUP BY CUBE clause.

Q17- Sequence object and identity

\*The IDENTITY property is tied to a particular table and cannot be shared among multiple tables since it is a table column property.

on the flip side the SEQUENCE object is defined by the user and can be shared by multiple tables since is it is not tied to any table.

Q18-Inline function and view

The principal difference between the two is that the inline table valued function can be built with a parameter where a view cannot. Most of the time, but not all the time, the inline functions are resolved out to the component tables in the same way that views are. With the addition of a filter through the parameter, this can make them perform better than views in certain circumstances.

However, test your code carefully. Functions, especially multi-statement table valued ones, can be serious performance pigs.

Q19- Table variable and temporary table

In SQL Server, based on the scope and behavior, temporary tables are of two types,

Example:

Local Temporary Tables (#temp)

Global Temporary Tables (##temp)

CREATE TABLE #StudentTemp

(

    StudentID int,

    Name varchar(50),

    Address varchar(150)

)

GO

INSERT INTO #StudentTemp VALUES ( 1, 'Dipendra','Pune');

GO

SELECT \* FROM #StudentTemp

CREATE TABLE #StudentTemp

(

    StudentID int,

    Name varchar(50),

    Address varchar(150)

)

GO

INSERT INTO #StudentTemp VALUES ( 1, 'Dipendra','Pune');

GO

SELECT \* FROM #StudentTemp

Table variable is a very useful programming construct, like that of any other variable.

 DECLARE @TStudent TABLE

 (

    RollNo INT IDENTITY(1,1),

    StudentID INT,

    Name INT

 )

 --Insert data to Table variable @TStudent

 INSERT INTO @TStudent(StudentID,Name)

 SELECT DISTINCT StudentID, Name FROM StudentMaster ORDER BY StudentID ASC

 --Select data from Table variable @TStudent

 SELECT \* FROM @TStudent

 --Next batch

 GO

 SELECT \* FROM @TStudent --gives error

DECLARE @TStudent TABLE

 (

    RollNo INT IDENTITY(1,1),

    StudentID INT,

    Name INT

 )

 --Insert data to Table variable @TStudent

 INSERT INTO @TStudent(StudentID,Name)

 SELECT DISTINCT StudentID, Name FROM StudentMaster ORDER BY StudentID ASC

 --Select data from Table variable @TStudent

 SELECT \* FROM @TStudent

 --Next batch

 GO

Q20- Row\_number() and dense\_Rank() function

The RANK, DENSE\_RANK and ROW\_NUMBER functions are used to retrieve an increasing integer value. They start with a value based on the condition imposed by the ORDER BY clause. All of these functions require the ORDER BY clause to function properly. In case of partitioned data, the integer counter is reset to 1 for each partition.