SQL

Interview Questions and Answers (Ver. 1.0)

For

Data Analysis & Business Intelligence

By

Ombir Rathee

(Business Intelligence Analyst)

SQL stands for **Structured query language**. SQL is a standard language that was designed to query and manage the data in RDBMS i.e Relational Database Management Systems. Microsoft provides **T-SQL** (Transact-SQL) as dialect of SQL in **Microsoft SQL Server** Data management software. In this document I mainly focused on **T-SQL**. However, most of SQL syntax remains same for other RDBMS software such as Oracle, MySQL, IBM DB2, PostgreSQL etc.

Ques 1. What are the different types of SQL commands/statements?

Ans. There are mainly 3 types of SQL statements

a. DDL: Data Definition Language (DDL) statements defines data structures.

• **CREATE** • **ALTER** • **DROP** • **RENAME**

b. DML: Data Manipulation Language (DML) affect the information stored in the database.

• **SELECT** • **INSERT** • **UPDATE** • **DELETE** • **TRUNCATE** • **MERGE**

Note: Oracle consider TRUNCATE as a DDL statement.

c. DCL: Data Control Language (DCL) deals with permissions on various objects.
• **GRANT** • **REVOKE**

Ques 2. What are the Constraints and its different types?

Ans. Constraints are used to enforce the Data integrity in a Table

• **UNIQUE**: Ensure all values in a column are different

• **NOT NULL**: Column cannot contain NULL values

• **PRIMARY KEY** : Combination of NOT NULL and UNIQUE

• **CHECK** : Specify data values that are acceptable in a Column (Ex: Salary >1000)

• **DEFAULT** : Set a Default value to Column when No Value is Specified

• **FOREIGN KEY** : Used to establish and enforce a link between the data in two tables to control the data that can be stored in the foreign key table

Ques 3. What is the Order of Execution of a SQL statement?

Ans.

1. FROM
2. JOIN
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT

7. DISTINCT
8. ORDER BY
9. TOP/OFFSET-FETCH

Ques 4. What are the different types of Joins in SQL?

Ans.

1. **INNER JOIN:** Return rows only when there is a match in both tables.
2. **LEFT OUTER JOIN:** Returns all rows from Left Table even there isn't any match in Right table. Place NULL in Right table when there isn't any match with Left Table.
3. **RIGHT OUTER JOIN:** Opposite to LEFT OUTER JOIN. Returns all rows from Right Table even if there isn't any match in Left table. Place NULL in Left table when there isn't any match with Right Table.
4. **FULL OUTER JOIN**: Its combination of LEFT and RIGHT JOIN. It returns all rows from both tables and place NULL when there isn't any match in both tables.
5. **CROSS JOIN**: Return the Cartesian Product of both Tables. i.e M*N rows.

Note: INNER/OUTER Keyword is optional

Ques 5. You have two tables with below structure and data. What will be the output in Number of rows for different joins on basis of ID as Join condition?

Table A

ID	Name
1	John
1	Peter
2	Veronica
5	Natasha

Table B

ID	Sale
1	10
1	20
2	30
7	40

Ans.

1. **INNER JOIN**: 5 Rows
2. **LEFT JOIN**: 6 Rows
3. **RIGHT JOIN**: 6 Rows
4. **FULL JOIN:** 7 Rows
5. **CROSS JOIN**: 16 Rows

Ques 6. What is Subquery and its types?

Ans. A subquery is a query that is nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery. A subquery is also called an inner query or inner select, while the statement containing a subquery is called an outer query or outer select.

1. **Self-Contained**: It has no dependency on Outer query that it belongs to.
2. **Correlated**: It has dependency on outer query.

Both Self-Contained and Correlated subqueries can be **Single-valued, Multi valued or table-valued.**

Ques 7. What is table expression and its types?

Ans. Table expression is a **named** query that represents a valid relational table. These are virtual and not physically stored anywhere.

1. **Derived Tables:** Derive tables are also known as **table subqueries** and are defined in **FROM** clause of an outer query. Their scope is limited to outer query. As soon as outer query is finished, the derived table is gone.
2. **CTE:** Common Table Expressions are used to temporarily store the result of a complex query. CTE are always followed by SELECT, INSERT, DELETE, UPDATE OR MERGE statements. These are of 2 types i.e Non-recursive and Recursive. Scope of CTE is also limited to outer query similar to derived tables.
3. **VIEWS:** Unlike Derive Table and CTE, VIEW is a reusable table expression whose definition is stored in database as objects. Just like other database objects we can control the access to View with permissions such as SELECT, INSERT, UPDATE, DELETE. Views are generally used for security purposes as they can display only selected data.
4. **Inline Table-Valued Functions:** These are also reusable table expressions that supports input parameters. TVFs are similar to views in all aspects except support for Input Parameters. They can be called as parameterized views.

Ques 8. What are the different types of SET operators?

Ans. A set operator compares complete rows between the result sets of two input queries. T-SQL supports 3 types of SET operators.

1. **UNION (A ∪ B)**

   UNION unifies the results of two input queries and eliminates the duplicate rows.
   UNION ALL also unifies the results of two input queries but doesn't remove duplicates. Hence it is faster than **UNION**.

2. **INTERSECT (A ∩ B)**

   INTERSECT operator first eliminates duplicate rows from two input queries and then returns the common rows between two input queries.

3. **EXCEPT (A-B)**

   EXCEPT operator first eliminates duplicate rows from two input queries and return rows that appear in first query but not in second

The two queries involved in SET operators must have the same number of columns and corresponding columns must have compatible data types. Compatible means, data type that has lower precedence must be implicitly convertible to higher data type. The names of column in result set are determined by first query. Set operator considers two NULLs as equal. The two Input queries cannot contain ORDER BY clause.

Ques 9. What is the difference between TRUNCATE and DELETE statement?

Ans.

TRUNCATE	**DELETE**
Remove all rows from Table	Can remove selected rows based on condition in WHERE clause
It is Minimally logged. Performance is faster than delete.	It is fully logged. Hence slower than Truncate.
Reset the Identity column if present to seed value.	Doesn't affect the Identity column.
Executed using Table Lock.	Executed using Row level Lock.
ALTER permission is required to Truncate the Table.	DELETE permission is required to perform the Delete operation.
DDL for Oracle; DML for MS SQL Server	DML command
Can be Rolled back only in Transaction if Transaction is not Committed or Transaction Session is not closed. Can't be Rolled back in Oracle even if it is in Transaction.	Can be rolled back

Ques 10. What is the difference between Store Procedure and Function?

Ans.

Store Procedure	**Function**
Function can be called from Store Procedure.	Store Procedure can't be called from Function.
Try-Catch statement is allowed for error handling.	Try-Catch is not allowed.
SP is not allowed in Select, From, Where, Having statement.	We can use Function with Select, From, Where, Having statement.
Transaction is allowed in SP.	Transaction is not allowed.
Can have Input and Output parameters.	Only Input parameters are allowed.
May or May not return a value.	Must return a value.
We can use DML statements inside SP.	DML operation is allowed but only on table variables that are within local scope of function. DML operation is not allowed on permanent tables.

Ques 11. What is the difference between Table variable and Temp Table?

Ans.

Temp Table (#/##)	Table Variable (@)
Scope of Local Temp table(#temp) is limited to the connection while Global Temp Table(##temp) is shared across all connections.	A **table** variable behaves like a local variable. It has a well-defined scope. This is the function, stored procedure, or batch that it is declared in.
Temp tables are cleared when session/Connection is terminated.	**Table** variables are automatically cleaned up at the end of the function, stored procedure, or batch in which they are defined.
Can be created using SELECT INTO and INSERT EXEC statements	Can't be created using SELECT INTO and INSERT EXEC. Their definition must be explicitly defined using CREATE TABLE statement.
Can be Truncated and do carry statistics.	Can't be Truncated and don't maintain Statistics. Due to lack of statistics, Optimizer will not be able to deter the best execution for complex queries on large dataset.
Can be Indexed like normal Table	Can't be Indexed via CREATE INDEX statement. Allow Index only via PRIMARY and UNIQUE constraints.
Non-Clustered Index can be created along with Clustered and Unique Index.	Can't create NON-Clustered Index on Table variable.
Recommended for larger data set.	Recommended for smaller data set i.e. upto 1000 rows for better performance.
Cannot be used inside functions.	Can be used inside Function.

Ques 12. What is the difference between View and Store Procedure?

Ans.

View	Store Procedure
Does not accept parameters	Accept Parameters
Can be used in FROM clause. Can be used a building block in larger query	Cannot be used in FROM clause. Hence, cannot be used a building block in larger query
Contains only Select query	Can contains several statements, IF-ELSE, Loop etc.
Cannot perform modification to any table	Can perform modification to one or several tables

Ques 13. What is the difference between COUNT(*) and COUNT(ColName)?

Ans. **COUNT(*)** count the number of rows in result set while **COUNT(ColName)** count the number of values in column ignoring NULL values.

Ques 14. What is Index in SQL?

Ans. Indexes speed up the querying process by providing swift access to rows in the data tables, similarly to the way a book's index helps you find information quickly within that book. When a SQL Server has no index to use for searching, the result is similar to the reader who looks at every page in a book to find a word; the SQL engine needs to visit every row in a table. In database terminology we call this behavior a table scan, or just scan. Indexes can be created on basically all columns except Large object data types such as **Image, Text and Varchar (Max).** However, **varchar(max)**, **varbinary(max)**, and **xml** data types can participate in a non-clustered index as non-key index columns.

Ques 15. What are different types of Indexes?

Ans.

1. **Clustered Index:** Clustered indexes sort and store the data rows in the table or view based on their key values. Key values are the column (or columns) included in the index definition. There can be only one clustered index per table, because the data rows themselves can be stored in only one order.
2. **Non-Clustered Index:** Non-clustered indexes cannot be sorted like clustered indexes. However, you can create more than one non-clustered index per table or view. The index in the back of a book is an example of a non-clustered index. A non-clustered index has the indexed columns and a pointer or bookmark pointing to the actual row. In the case of our example it contains a page number. Another example could be a search done on Google or another of the search engines. The results on the page contain links to the original web pages.

   In addition to an index being clustered or non-clustered, it can be configured in other ways:

a. **Composite index**: An index that contains more than one column. You can include up to 16 columns in an index, as long as the index doesn't exceed the 900-byte limit. Both clustered and non-clustered indexes can be composite indexes.
b. **Unique Index:** An index that ensures the uniqueness of each value in the indexed column. If the index is a composite, the uniqueness is enforced across the columns as a whole, not on the individual columns. For example, if you were to create an index on the FirstName and LastName columns in a table, the names together must be unique, but the individual names can be duplicated.
c. **Covering Index:** When all of the required columns in SELECT list are part of the index, it is called a covering index. It is created using INCLUDE statement and can be created only with Non-Clustered Index. It can include Non-key columns in in the Index to significantly improves the query performance because the query optimizer can locate all the column values within the index, table or clustered index data is not accessed resulting in fewer disk I/O operations.

Note: A table that has a clustered index is referred to as a clustered table. A table that has no clustered index is referred to as a heap.

Ques 16. What are the disadvantages of using an Index?

Ans.

1. **Disk Space:** Indexes are stored on the disk, and the amount of space required will depend on the size of the table, and the number and types of columns used in the index. Disk space is generally cheap enough to trade for application performance, particularly when a database serves a large number of users.
2. **Data Modification:** Any time a query modifies the data in a table (INSERT, UPDATE, or DELETE), the database needs to update all of the indexes where data has changed. Hence, providing too many indexes to update can actually hurt the performance of data modifications.

Ques 17. What are the factors to be considered while creating an Index?

Ans. Well-designed indexes can reduce disk I/O operations and consume fewer system resources therefore improving query performance. Consider the following guidelines when you design an Index.

1. Understand the characteristics of the columns used in the queries. For example, an index is ideal for columns that have an integer data type and are also unique or nonnull columns.
2. Create non-clustered indexes on the columns that are frequently used in predicates(Where, Having) and join conditions in queries. However, you should avoid adding unnecessary columns. Adding too many index columns can adversely affect disk space and index maintenance performance.
3. Avoid over-indexing heavily updated tables and keep indexes narrow, that is, with as few columns as possible.
4. Use many indexes to improve query performance on tables with low update requirements, but large volumes of data. . Large numbers of indexes can help the performance of queries that do not modify data, such as SELECT statements.
5. Indexes on views can provide significant performance gains when the view contains aggregations, table joins, or a combination of aggregations and joins
6. Determine which index options might enhance performance when the index is created or maintained. For example, creating a clustered index on an existing large table would benefit from the ONLINE index option. The ONLINE option allows for concurrent activity on the underlying data to continue while the index is being created or rebuilt.
7. Determine the optimal storage location for the index. A non-clustered index can be stored in the same filegroup as the underlying table, or on a different filegroup. The storage location of indexes can improve query performance by increasing disk I/O performance. For example, storing a non-clustered index on a filegroup that is on a different disk than the table filegroup can improve performance because multiple disks can be read at the same time. Alternatively, clustered and non-clustered indexes can use a partition scheme across multiple filegroups. Partitioning makes large tables or indexes more manageable by letting you access or manage subsets of data quickly and efficiently, while maintaining the integrity of the overall collection.

Ques 18. What is Partitioning and its benefits?

Ans. SQL Server supports table and index partitioning. Partitioning is a way to divide a large table into smaller, more manageable parts without having to create separate tables for each part. Data in a partitioned table is physically stored in groups of rows called partitions and each partition can be accessed and maintained separately. Partitioning is not visible to end users, a partitioned table behaves like one logical table when queried.

Benefits:

1. You can transfer or access subsets of data quickly and efficiently, while maintaining the integrity of a data collection
2. You can perform maintenance operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table.

It is mostly intended to aid in maintenance on larger tables and to offer fast ways to load and remove large amounts of data from a table. Partitioning can enhance query performance, but there is no guarantee.

Ques 19. What is a Computed Column?

Ans. A computed column is a virtual column that is not physically stored in the table, unless the column is marked PERSISTED. A computed column expression can use data from other columns to calculate a value for the column to which it belongs.

```
CREATE TABLE dbo.Products
(
   ProductID int IDENTITY (1,1) NOT NULL
  , QtyAvailable smallint
  , UnitPrice money
  , InventoryValue AS QtyAvailable * UnitPrice  ---Computed Column
)
```

Ques 20. What is the difference between PRIMARY AND UNIQUE KEY?

Ans.

PRIMARY	**UNIQUE**
Primary Key is used to identify a row (record) in a table.	Unique-key is used to prevent duplicate values in a column (with the exception of a null entry)
There can be only one primary key in a table.	Can be more than one unique key in a table
By Default Primary Key create clustered Index.	By Default Unique Key creates Non-Clustered Index.
Cannot contains NULL values	Can contains NULL values.

Ques 21. What is a Candidate Key?

Ans. A table can have multiple column (or Combination of Columns) which can uniquely identify a row. This column (or Combination of Columns) are referred as **candidate keys**. There can be multiple Candidate Keys in one table. Each Candidate Key can qualify as Primary Key.

Ques 22. What is the difference between Varchar and Nvarchar data type ?

Ans.

1. Nvarchar stores UNICODE data. If you have requirements to store UNICODE or multilingual data, nvarchar is the choice. Varchar stores ASCII data and should be your data type of choice for normal use.
2. Nvarchar uses 2 bytes per character, whereas Varchar uses 1.

Ques 23. What is the difference between ISNULL, IFNULL and NULLIF?

Ans.

1. **NULLIF (exp1,exp2):** Return NULL if exp1=exp2 else return exp1.
2. **IFNULL(exp1,exp2):** Return exp1 if it is not NULL else return exp2.
3. **ISNULL(exp1,value):** Return **value** if exp1 is NULL else return exp1.

Ques 24. What is Collation?

Ans. Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case-sensitivity, accent marks, kana character types and character width.

- **Case sensitivity:** A and a, B and b, etc. are treated in the same way then it is case-insensitive
- **Accent sensitivity:** If a and á, o and ó are treated in the same way, then it is accent-insensitive
- **Kana Sensitivity:** When Japanese kana characters Hiragana and Katakana are treated differently, it is called Kana sensitive
- **Width sensitivity:** When a single-byte character (half-width) and the same character when represented as a double-byte character (full-width) are treated differently then it is width sensitive.

  SYNTAX: **COLLATE** Latin1_General_CS_AS_KS_WS ASC

  The COLLATE clause can be applied only for the **char**, **varchar**, **text**, **nchar**, **nvarchar**, and **ntext** data types. SQL Server allows the users to create databases, tables and columns in different collations.

Ques 25. Which operator is used for Pattern Matching?

Ans. **LIKE** operator is used for pattern matching. It supports below wildcards.

- % : Matches any string of zero or more characters.
- _ : Matches any single character.
- [] : Matches any single character within the specified range ([a-f]) or set ([abcdef]).
- [^] : Matches any single character not within the specified range ([^a-f]) or set ([^abcdef]).

Ques 26. How to Select Unique Records from a Table?

Ans. Using **DISTINCT** keyword in Select Query

Ques 27. What is a Cursor?

Ans. A cursor is a database object which is used to manipulate data in a row-to-row manner

Cursor follow steps as given below

- Declare Cursor
- Open Cursor
- Retrieve row from the Cursor
- Process the row
- Loop until last row
- Close Cursor
- Deallocate Cursor

Ques 28. What is the difference between WHERE and HAVING clause?

Ans.

- Both are used to filter the dataset but WHERE is applied first and HAVING is applied at later stage of query execution.
- WHERE can be used in any SELECT query, while HAVING clause is only used in SELECT queries, which contains aggregate function or group by clause.
- Apart from SELECT queries, you can use WHERE clause with UPDATE and DELETE clause but HAVING clause can only be used with SELECT query

Ques 29. What is the use of IDENTITY?

Ans. An identity column in the SQL automatically generates unique numeric values. We can define a start and increment value of identity column.

Ques 30. What is the difference between RENAME and ALIAS?

Ans. Rename is used to give a permanent name to a table or column whereas Alias is a temporary name given to a table or column.

Ques 31. What is the use of COALESCE function?

Ans. Evaluates the arguments in order and returns the first value that is NOT NULL.
For example:

```
SELECT COALESCE(NULL, NULL, 'third_value', 'fourth_value')-- returns the third  value
```

Ques 32. What are the different Ranking Window functions in SQL Server?

Ans.

SYNTAX:

```
ROW_NUMBER|RANK|DENSE_RANK|NTILE () OVER ([PARTITION BY <partition_column>]
                                        ORDER BY <order_by_column>)
```

The PARTITION BY clause is optional. If not used, data will be ranked based on a single partition.

1. **ROW_NUMBER():** Always generate unique values without any gaps, even if there are ties.

2. **RANK():** Ranks each row in the result set. Can have gaps in its sequence and when values are the same, they get the same rank.

3. **DENSE_RANK():** It also returns the same rank for ties, but doesn't have any gaps in the sequence.

4. **NTILE():** Divides the rows in roughly equal sized buckets. Suppose you have 20 rows and you specify NTILE(2). This will give you 2 buckets with 10 rows each. When using NTILE(3), you get 2 buckets with 7 rows and 1 bucket with 6 rows.

Ques 33. What is the use of VALUES constructor?

Ans. The T-SQL table value constructor allows multiple rows of data to be specified in a single DML statement. The table value constructor can be specified in the VALUES clause of the INSERT statement, in the USING <source table> clause of the MERGE statement, and in the definition of a derived table in the FROM clause.

- ```
  INSERT INTO Products (Name, ListPrice)
  VALUES ('Helmet', 25.50),
         ('Wheel', 30.00)
  ```

- ```
  SELECT a, b
  FROM (VALUES (1, 2), (3, 4), (5, 6), (7, 8), (9, 10) ) AS MyTable(a, b)
  ```

Ques 34. What are the different types of CASE statement?

Ans. CASE statement has two formats. Both allows up-to 10 levels of Nesting.

1. Simple CASE: It allows only equality operator.

```
CASE YEAR(OrderDT)
                WHEN 2014 THEN 'Year 1'
                WHEN 2013 THEN 'Year 2'
                WHEN 2012 THEN 'Year 3'
                ELSE 'Year 4 and beyond' END AS YearType
```

2. Searched CASE: It also supports other operators such as >, < ,>=, <=, <>

```
CASE
        WHEN YEAR(OrderDT) = 2014 THEN 'Year 1'
        WHEN YEAR(OrderDT) = 2013 THEN 'Year 2'
        WHEN YEAR(OrderDT) = 2012 THEN 'Year 3'
        WHEN YEAR(OrderDT) < 2012 THEN 'Year 4 and beyond'
END AS YearType
```

Ques 35. How to check the number of rows affected by Last statement?

Ans. @@ROWCOUNT return the number of rows affected by Last statement.

Ques 36. What is use of SET NOCOUNT ON statement?

Ans. Stops the message that shows the count of the number of rows affected by a Transact-SQL statement or stored procedure. It is used to improve the performance in a Store procedure as it turns off the messages that SQL Server sends back to the client after each T-SQL statement is executed.

Ques 37. What are the different Analytic functions available in SQL Server?

Ans.

1. **FIRST_VALUE():** Returns the first value in an ordered set of values. If **Partition By** clause is specified then it returns First Value in each partition after ordering the partition by **Order By** Clause.

2. **LAST_VALUE():** Returns the Last value in an ordered set of values. If **Partition By** clause is specified then it returns Last Value in each partition after ordering the partition by **Order By** Clause.

3. **LAG():** Provides access to a row at a given physical offset that comes before the current row. Use this function in a **SELECT** statement to compare values in the current row with values in a previous row as specified by offset. Default offset is 1 if not specified. If **Partition By** clause is specified then it returns the offset Value in each partition after ordering the partition by **Order By** Clause.

4. **LEAD():** Provides access to a row at a given physical offset that comes after the current row. Use this function in a **SELECT** statement to compare values in the current row with values in a subsequent row as specified by offset. Default offset is 1 if not specified. If **Partition By** clause is specified then it returns the offset Value in each partition after ordering the partition by **Order By** Clause.

5. **PERCENT_RANK():** Used to evaluate the relative standing of a value within a query result set or partition. The range of values returned by PERCENT_RANK is greater than 0 and less than or equal to 1.

Query Based Questions

Ques 1. How to check Indexes created on a table?

Ans. `EXEC sp_helpindex TblName`

Ques 2. How to check definition of Store procedure?

Ans. `EXEC sp_helptext 'SpName'`

Ques 3. How can you Create a Table without Create table statement?

Ans.
```
SELECT * INTO NEWTABLE FROM SOURCETABLE
```

Ques 4. How can you Create a Table without Create table statement without any Data?

Ans.
```
SELECT * INTO NEWTABLE FROM SOURCETABLE WHERE 1=2
```

Ques 5. How to convert DateTime data type to Date?

Ans.
```
SELECT CONVERT(date, GETDATE())
```

Ques 6. Write a query to calculate the exact age in years from DOB?

Ans.
```
DECLARE @DOB DATE='1990-01-06' --YYYY-MM-DD
SELECT DATEDIFF(hour,@DOB,CAST(GETDATE() as Date))/8766.0
```

Ques 7. Write the query to calculate the following?

Ans.

- **First day of previous month**
  ```
  SELECT DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) - 1, 0)
  ```
- **First day of next month**
  ```
  SELECT DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) + 1, 0)
  ```
- **Last day of Previous month**
  ```
  SELECT DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()), -1)
  ```
- **Last day of Next month**
  ```
  SELECT DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) + 2, -1)
  ```
- **First day of previous quarter**
  ```
  SELECT DATEADD(QUARTER, DATEDIFF(QUARTER, 0, GETDATE()) -1, 0)
  ```
- **Last day of previous quarter**
  ```
  SELECT DATEADD(QUARTER, DATEDIFF(QUARTER, 0, GETDATE()), -1)
  ```
- **First day of current quarter**

```
SELECT DATEADD(QUARTER, DATEDIFF(QUARTER, 0, GETDATE()), 0)
```

- **Last day of current quarter**

```
SELECT DATEADD(QUARTER, DATEDIFF(QUARTER, 0, GETDATE()) + 1, -1)
```

- **Last day of 1st quarter of current year**

```
SELECT DATEADD(QUARTER, 1, DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), -1))
```

- **Last day of 4th quarter of previous year**

```
SELECT DATEADD(QUARTER, 4, DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()) - 1, -1))
```

Ques 8. Write a Query to return the Rows sorted randomly?

Ans.

```
SELECT * FROM TblName ORDER BY NEWID()
```

Ques 9. Write a SQL query to generate Random Number between two numbers?

Ans.

```
DECLARE @TOP INT = 10
DECLARE @BOTTOM INT = 20
SELECT CAST(ROUND((@TOP-@BOTTOM)* RAND() +@BOTTOM,0) as INT)
```

Ques 10. How can you add a column with default value be added to existing table?

Ans.

```
ALTER TABLE TblName
      ADD ColName INT NULL
 CONSTRAINT Constraint_Name
      DEFAULT (1)
WITH VALUES
```

Ques 11. How to get all Column names from a Table?

Ans.

```
SELECT *
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='TblName'
```

Ques 12. How to set a default value to existing column?

Ans.

```
ALTER TABLE TblName
      ADD CONSTRAINT SomeName
      DEFAULT 'Value' FOR ColumnName
```

Ques 13. How to check if a column exists in table?

Ans.

1.
```
IF EXISTS(SELECT 1 FROM sys.columns
WHERE Name = 'ColName'
AND Object_ID = Object_ID('TblName'))
BEGIN
        -- Column Exists
END
```

2.
```
IF COL_LENGTH('TblName', 'ColName') IS NOT NULL
BEGIN
        -- Column Exists
END
```

Ques 14. How to check if Table exists in Database?

Ans.

a. Normal Table

```
IF OBJECT_ID (TblName, 'U') IS NOT NULL
BEGIN
        --Table Exists
END
```

b. Temp Table

```
IF OBJECT_ID('tempdb..#TblName') IS NOT NULL
BEGIN
        -- Table Exists
END
```

Ques 15. Write a Update query with Join statement?

Ans.

```
UPDATE A
SET ColB = ColB + B.ColB,
        ColC= B.ColC
FROM Table1 AS A
        INNER JOIN Table2 AS B
ON A.ColA = B.ColA
```

Ques 16. Write a Delete query with Join statement?

Ans.

```
DELETE A
 FROM
     Table1 AS A
  INNER JOIN Table2 AS B
     ON A.ID = B.ID
 WHERE  B.Sales > 10000
```

Ques 17. Write a query to generate the numbers between 1 and N?

Ans.

```
SELECT TOP 1000 --Replace 1000 with Any Number
ROW_NUMBER() OVER (ORDER BY A.OBJECT_ID) AS N
FROM SYS.ALL_OBJECTS AS A
CROSS JOIN SYS.ALL_OBJECTS AS B
```

Ques 18. Write a query to sort a column in a user defined order? For ex: If a field contains name of different colors then Red to appear first, Blue to appear second and rest of colors in ascending order of Color. IF NULL is available in column then it should appear last.

Ans.

INPUT

ID	Color
1	PURPLE
2	RED
3	GREEN
4	BLUE
5	VIOLET
6	RED
7	YELLOW
8	NULL

OUTPUT

ID	Color
2	RED
6	RED
4	BLUE
3	GREEN
1	PURPLE
5	VIOLET
7	YELLOW
8	NULL

```
SELECT * FROM TblName
ORDER BY
        CASE
                WHEN Color='RED' THEN '0'
                WHEN Color='BLUE' THEN '1'
                WHEN COLOR IS NULL THEN 'zz'
                ELSE COLOR
        END
```

Ques 19. Write a query to find the employee with Nth highest salary?

Ans.

INPUT	ID	Salary
	1	50000
	2	60000
	3	70000
	4	80000
	5	100000
	6	100000
	7	110000
	8	120000

OUTPUT	ID	Category
Highest	8	120000
3rd Highest	5	100000
	6	100000
6th Highest	2	60000

```
;WITH CTE AS
(
        SELECT *,
        DENSE_RANK() OVER (ORDER BY SALARY DESC) AS RANKING
        FROM TblName
)

SELECT * FROM CTE WHERE RANKING=1 -- HIGHEST
--SELECT * FROM CTE WHERE RANKING=3 –-3rd HIGHEST
--SELECT * FROM CTE WHERE RANKING=6 –-6TH HIGHEST
```

Ques 20. Write a query to find the Manager of Manager of an employee?

Ans.

INPUT	EmpID	ManID
	Emp_1	Emp_3
	Emp_2	Emp_3
	Emp_3	Emp_5
	Emp_4	Emp_5
	Emp_5	Emp_8
	Emp_6	Emp_8
	Emp_7	Emp_8
	Emp_8	Emp_9

OUTPUT	EmpID	Manager of Manager
	Emp_1	Emp_5
	Emp_2	Emp_5
	Emp_3	Emp_8
	Emp_4	Emp_8
	Emp_5	Emp_9
	Emp_6	Emp_9
	Emp_7	Emp_9
	Emp_8	NULL

```
SELECT
A.EMPID,
B.MANID AS [MANAGER OF MANAGER]
```

```
FROM Table1 AS  A
LEFT JOIN Table1 AS B
    ON A.MANID=B.EMPID
```

Ques 21. Write a query to find the Top N students in each Subject?

Ans.

INPUT	Subject	SID	Marks
	PHY	1	90
	PHY	2	50
	PHY	3	60
	PHY	4	70
	CHE	1	80
	CHE	2	90
	CHE	3	40
	CHE	4	60
	MAT	1	85
	MAT	2	85
	MAT	3	60
	MAT	4	70

OUTPUT	Subject	SID	Marks
	PHY	1	90
	PHY	4	70
	CHE	2	90
	CHE	1	80
	MAT	1	85
	MAT	2	85
	MAT	4	70

```
;WITH CTE AS
(
SELECT *,
DENSE_RANK() OVER(PARTITION BY SUB ORDER BY MARKS DESC) AS RANKING
FROM TblName
)
SELECT * FROM CTE WHERE RANKING<=2 ---Top 2 in each Subject
```

Ques 22. Write a query to Delete the Duplicate records from a table?

Ans.

Input	Id	Age	Country
	1	25	India
	1	25	India
	2	26	USA
	3	28	Australia
	4	26	UK
	4	26	Canada
	5	25	Switzerland
	5	25	Switzerland
	5	25	Switzerland

Output	Id	Age	Country
	1	25	India
	2	26	USA
	3	28	Australia
	4	26	UK
	4	26	Canada
	5	25	Switzerland

```
;WITH CTE AS
(
  SELECT ROW_NUMBER() OVER (PARTITION BY ID, AGE, COUNTRY
                            ORDER BY ( SELECT 0)) As RowNo
  FROM  TblName
)
DELETE FROM CTE WHERE  RowNo > 1
```

Ques 23. When to use MERGE statement? Write its Syntax.

Ans. MERGE statement is used to perform INSERT, UPDATE and DELETE operations on a TARGET table by matching the records from the SOURCE table.

```
MERGE Products AS TARGET
USING UpdatedProducts AS SOURCE
ON TARGET.ProductID = SOURCE.ProductID
        --When records are matched, update
        --the records if there is any change
WHEN MATCHED AND TARGET.ProductName <> SOURCE.ProductName
OR TARGET.Rate <> SOURCE.Rate THEN
UPDATE SET TARGET.ProductName = SOURCE.ProductName,
TARGET.Rate = SOURCE.Rate
        --When no records are matched, insert
        --the incoming records from source
        --table to target table
WHEN NOT MATCHED BY TARGET THEN
INSERT (ProductID, ProductName, Rate)
VALUES (SOURCE.ProductID, SOURCE.ProductName, SOURCE.Rate)
        --When there is a row that exists in target table and
        --same record does not exist in source table
        --then delete this record from target table
WHEN NOT MATCHED BY SOURCE THEN
DELETE
```

Ques 24. Write a Query to join rows values with comma as Delimiter?

Ans.

INPUT	ID	Category
	1	Apple
	1	Orange
	1	Banana
	3	Chair
	3	Table
	2	Paper
	2	Pencil
	2	Eraser

OUTPUT	ID	Category
	1	Apple, Banana, Orange
	2	Chair, Table
	3	Eraser, Paper, Pencil

```
SELECT DISTINCT
ID,
STUFF((SELECT ','+CATEGORY
FROM TblName B WHERE A.ID=B.ID
ORDER BY CATEGORY
FOR XML PATH('')
),1,1,'') AS CATEGORY
FROM TblName AS A
GROUP BY ID
ORDER BY ID
```

Expert Level Questions

Ques 1. Write a query to find the number of days of Gap between a Date range?

Ans:

Input	DayOnline
	5/1/2018
	5/2/2018
	5/3/2018
	5/4/2018
	5/5/2018
	5/7/2018
	5/8/2018
	5/9/2018
	5/15/2018
	5/16/2018
	5/17/2018
	5/19/2018
	5/20/2018

Output	GapStart	GapEnd	Days
	5/6/2018	5/6/2018	1
	5/10/2018	5/14/2018	5
	5/18/2018	5/18/2018	1

```
SELECT
   GapStart = DATEADD(DAY,1,[current]),
   GapEnd   = DATEADD(DAY,-1,[next]),
   [Days]     = DATEDIFF(day,DATEADD(DAY,1,[Current]),DATEADD(DAY,-1,[Next]))+1
FROM
(
SELECT
   [Current] = [DayOnline],
   [Next]    = LEAD([DayOnline]) OVER (ORDER BY [DayOnline])
FROM TblName
) A
WHERE DATEDIFF(DAY,[Current],[Next]) > 1
```

Ques 2. Write a query to find the number of days between continuous dates in a Date Range?

Ans:

Input	SomeDate
	1/1/2012
	1/1/2012
	1/3/2012
	1/5/2012
	1/6/2012
	1/10/2012
	1/10/2012
	1/11/2012
	1/11/2012
	1/11/2012
	1/12/2012

Output	StartDate	EndDate	Days
	1/1/2012	1/1/2012	1
	1/3/2012	1/3/2012	1
	1/5/2012	1/6/2012	2
	1/10/2012	1/12/2012	3

```
;WITH GroupedDates AS
(
 SELECT UniqueDate = SomeDate,
        DateGroup  = DATEADD(dd, - ROW_NUMBER() OVER (ORDER BY SomeDate), SomeDate)
 FROM TblName
 GROUP BY SomeDate
)
SELECT StartDate = MIN(UniqueDate),
       EndDate   = MAX(UniqueDate),
       Days      = DATEDIFF(day,MIN(UniqueDate),MAX(UniqueDate))+1
FROM GroupedDates
GROUP BY DateGroup
ORDER BY StartDate
```

Ques 3. Write a query to create a Pivot with Rows and Columns Total?

Ans:

Input	ID	Category	Value
	1	A	16
	1	A	20
	1	B	12
	1	C	19
	2	A	13
	2	B	15
	2	B	10
	2	C	15
	3	A	17
	3	B	18
	3	C	11
	3	C	12

Output		A	B	C	Row Total
	1	36	12	19	67
	2	13	25	15	53
	3	17	18	23	58
	Col Total	66	55	57	178

```
SELECT
[ ] =  ISNULL(ID, 'Col Total'), --ID is of Varchar type
A= SUM(A),
B= SUM(B),
C= SUM(C),
[Row Total]= SUM(A+B+C)
FROM TblName AS A
PIVOT
(
        SUM(VALUE) FOR CATEGORY IN([A],[B],[C])
) AS A
GROUP BY ROLLUP(ID)
```

Ques 4. Write a Query to split the concatenated values with comma as Delimiter into rows?
Ans:

INPUT	ID	Category
	1	Apple, Banana, Orange
	2	Eraser, Paper, Pencil
	3	Chair, Table

OUTPUT	ID	Category
	1	Apple
	1	Banana
	1	Orange
	2	Eraser
	2	Paper
	2	Pencil
	3	Chair
	3	Table

```
SELECT A.ID,
     Split.a.value('.', 'VARCHAR(100)') AS Data
 FROM
 (
     SELECT ID,
         CAST ('<M>' + REPLACE(CATEGORY, ',', '</M><M>') + '</M>' AS
XML) AS Data
     FROM TblName
 ) AS A CROSS APPLY Data.nodes ('/M') AS Split(a)
```