Explain different types of views. Demonstrate with suitable examples.

Answer A view based on multiple tables, which contain GROUP BY clause and functions. Inline View: A view based on a subquery in FROM Clause, that subquery creates a temporary table and simplifies the complex query. Materialized View: A view that stores the definition as well as data.

In SQL, we can have two types of views, namely system-defined views and user-defined views. Within user-defined views, the two types of views that are widely known:

- **Simple View**: Simple views are views that are created on a single table. We can perform only basic SQL operations in simple views.
- **Complex View**: Complex views as the name suggest are a bit complicated compared to simple views. Complex views are created on more than one database table. We can perform analytical and aggregate operations in complex views, but unlike simple views, we cannot perform insert, delete, and update directly from a complex view.
- In some databases like SQL server, we have some system-defined views too. They are views for routines, schemas, table privileges, table privileges, check constraints, etc. They are automatically created when we create a database.

What is the difference between function and stored procedure? Write syntax for creating functions and stored procedures.

Answer - Functions cannot change anything and must have at least one parameter. Also, it must return a result. Stored procedures take no parameters, can modify database objects, and need not return results. Stored procedures join SQL queries into transactions and communicate with the outside world.
Stored procedures take no parameters, can modify database objects, and need not return results.

Functions can only have input parameters, whereas procedures can have either input or output parameters.

Functions can be called from procedures, but procedures cannot be called from functions.

Exceptions can be handled in try-catch blocks within procedures, but try-catch blocks cannot be used within functions.

Procedures cannot be used in SELECT statements, but functions can be embedded in SELECT statements.

What is an index in SQL? What are the different types of indexes in SQL?

Answer- Indexes in SQL are the individual lookup tables, which are utilized by the data set internet searcher to accelerate the general information recovery. The use of the index in SQL is to rapidly

discover the data in a data set table without looking through each row of it. The use of the index in SQL is to rapidly discover the data in a data set table without looking through each row of it. In SQL Index, it is basic to keep up more extra storage to make a copy duplicate of the data set. Tables in SQL server is contained inside database item holders that are called Schemas. The schema likewise fills in as a security limit, where you can restrict data set client authorizations to be on a particular schema level as it were. To know what the different types of Indexes in SQL Server are, then read this article to explore them and have a better understanding of them.

Different Types of Indexes in SQL Server

There are various types of indexes in SQL server:

1. Clustered Index
2. Non-Clustered Index
3. Column Store Index
4. Filtered Index
5. Hash Index
6. Unique Index

Showcase an example of exception handling in SQL stored procedure.

Answer- To handle exception in SQL Server we have TRY..CATCH blocks. We put T-SQL statements in TRY block and to handle exception we write code in CATCH block. If there is an error in code within TRY block then the control will automatically jump to the corresponding CATCH blocks. In Sql Server, against a Try block, we can have only one CATCH block.

What is a temporary and a variable table? Write suitable syntax to create temporary tables and variable tables.

Answer - Temp Variables are also used for holding data temporarily just like a temp table. Temp Variables are created using a “DECLARE” statement and are assigned values using either a SET or SELECT command. After declaration, all variables are initialized as NULL, unless a value is provided as part of the declaration.

```
1. Declare @My_vari TABLE
2. (
3.    IID int,
4.    Name Nvarchar(50),
5.    Salary Int ,
6.    City_Name Nvarchar(50)
7. )
```