SQL Assignment 4

---

1. **Explain different types of views. Demonstrate with suitable examples.**
   Ans-
   - SQL has a special version of tables called View, which is a virtual table that is compiled in runtime.
   - A View is just an SQL statement, and the data associated with it is not physically stored in the view but is stored in its base tables of it.
   - It can contain all the rows and columns of a table or only a few selected rows and columns if there is a need to restrict access.
   - **Type of SQL views:-**

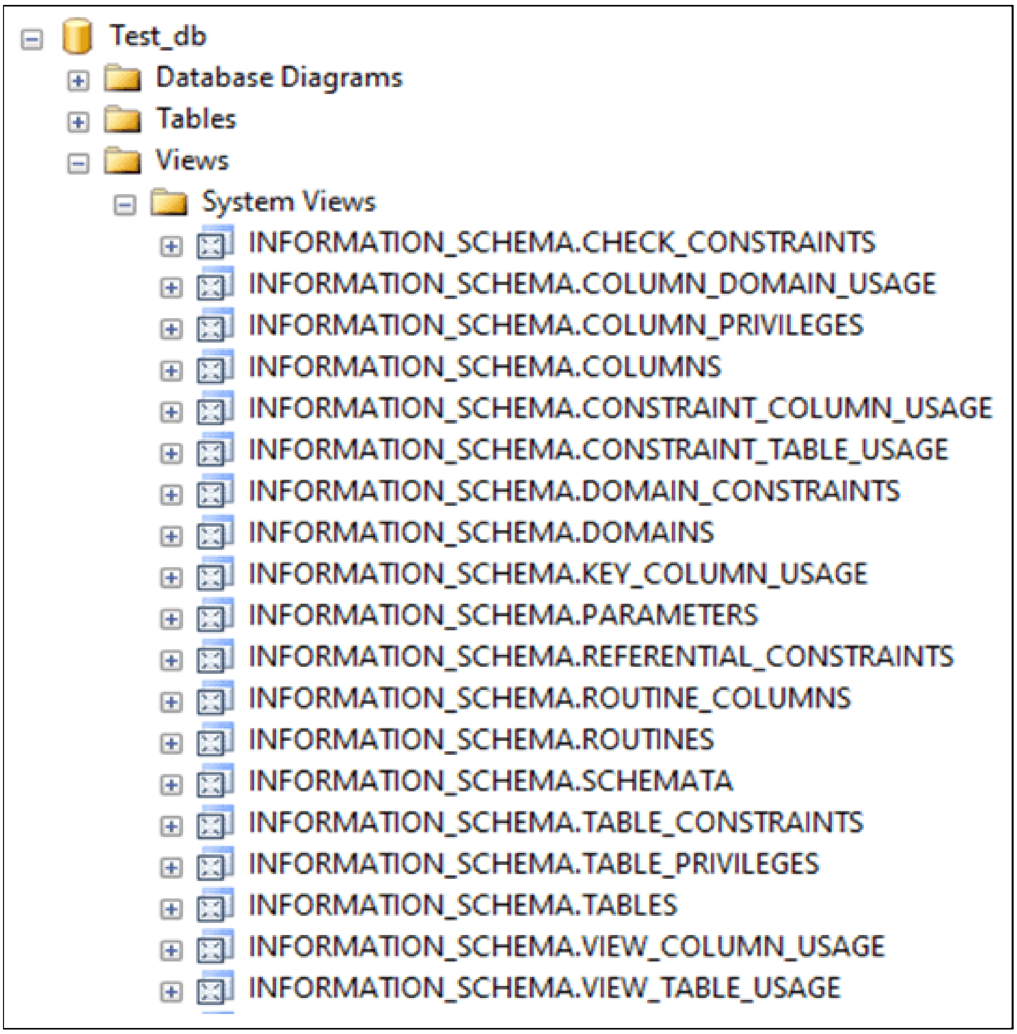
     1- User Defined Views
     2- System Defined Views
   - **1- User Defined Views-** These are the types of views that are defined by the users. There are two types under User Defined views, **Simple View** and **Complex View**.
     - **Simple View-**These views can only contain a single base table or can be created only from one table. Group functions such as MAX(), COUNT(), etc., cannot be used here, and it does not contain groups of data.
     - DML operations can be performed. Insert, delete, and update are directly possible, but Simple View does not contain group by, pseudocolumn like rownum, distinct, columns defined by expressions. Simple view also does not include NOT NULL columns from the base tables.
     - **Example-** A view based on the employee table.

       ```
       CREATE VIEW employee_dept1_view AS
       SELECT * FROM employee
       WHERE department_id = 'D01';
       ```
     - **Complex View-** Complex views as the name suggest are a bit complicated are created on more than one database table.
     - It 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.
     - **Example-** A view based on the employee and department table.

       ```
       CREATE VIEW employee_details AS
       ```

```
SELECT e.employee_name, e.salary,
e.highest_qualification, d.department_name,
d.location
FROM employee AS e JOIN departments as d
ON e.department_id = d.department_id;
```

- **2-System Defined Views-** 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. Here is an image from SQL server management studio that gives us an understanding of what system views are:
- **Example-**



```
⊟ Test_db
    ⊞ Database Diagrams
    ⊞ Tables
    ⊟ Views
        ⊟ System Views
            ⊞ INFORMATION_SCHEMA.CHECK_CONSTRAINTS
            ⊞ INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE
            ⊞ INFORMATION_SCHEMA.COLUMN_PRIVILEGES
            ⊞ INFORMATION_SCHEMA.COLUMNS
            ⊞ INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
            ⊞ INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE
            ⊞ INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS
            ⊞ INFORMATION_SCHEMA.DOMAINS
            ⊞ INFORMATION_SCHEMA.KEY_COLUMN_USAGE
            ⊞ INFORMATION_SCHEMA.PARAMETERS
            ⊞ INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
            ⊞ INFORMATION_SCHEMA.ROUTINE_COLUMNS
            ⊞ INFORMATION_SCHEMA.ROUTINES
            ⊞ INFORMATION_SCHEMA.SCHEMATA
            ⊞ INFORMATION_SCHEMA.TABLE_CONSTRAINTS
            ⊞ INFORMATION_SCHEMA.TABLE_PRIVILEGES
            ⊞ INFORMATION_SCHEMA.TABLES
            ⊞ INFORMATION_SCHEMA.VIEW_COLUMN_USAGE
            ⊞ INFORMATION_SCHEMA.VIEW_TABLE_USAGE
```

SQL Assignment 4

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

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.

Syntax-
CREATE FUNCTION function_name [(parameter datatype [, parameter datatype])]
RETURNS return_datatype
BEGIN
  declaration_section
   executable_section
END;
 	
Ex-
DELIMITER //
CREATE FUNCTION CalcIncome (starting_value INT)
RETURNS INT
BEGIN
  DECLARE income INT; SET income = 0;
  label1: WHILE income <= 3000 DO
  SET income = income + starting_value;
  END WHILE label1;
 RETURN income;
END; // DELIMITER ;
	Syntax-
	CREATE PROCEDURE procedure_name
	AS
	Sql_statement
	
	Ex-
	CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
	AS
	SELECT * FROM Customers WHERE City = @City

SQL Assignment 4

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

   Ans-

   - An index is a schema object. It is **used by the server to speed up the retrieval of rows by using a pointer**.
   - It can reduce disk I/O(input/output) by using a rapid path access method to locate data quickly. An **index helps to speed up select queries and where clauses**, but it **slows down data input**, with the update and the insert statements.
   - Indexes can be created or dropped with no effect on the data.

   - **Different types of indexes in SQL:-**

   - **Cluster Index-** Clustered indexes **sort and store the data rows in the table or view based on their key values**. These are the 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.
   - The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
   - **Non-Cluster Index-** Nonclustered indexes **have a structure separate from the data rows**. A nonclustered index contains the nonclustered index key values and each key-value entry has a pointer to the data row that contains the key value.
   - The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
   - **Unique Index-** A unique index **ensures the index key columns do not contain any duplicate values**. A unique index may consist of one or many columns.
   - If a unique index has one column, the values in this column will be unique. In case the unique index has multiple columns, the combination of values in these columns is unique.

- **Filter Index-** Filtered Index is a new feature in SQL SERVER. A filtered index is used **to index a portion of rows in a table which means it applies a filter on INDEX which improves query performance, reduces index maintenance costs**, and reduces index storage costs compared with full-table indexes.
- **Columnstore Index-** Columnstore index is a new type of index introduced in SQL Server 2012. It is a **column-based non-clustered index geared toward increasing query performance** for workloads that involve large amounts of data, typically found in data warehouse fact tables.
- **Hash Index-** Basically, a hash index is **an array of N buckets or slots, each one containing a pointer to a row**. Hash indexes use a hash function F(K, N) in which given a key K and the number of buckets N, the function maps the key to the corresponding bucket of the hash index.

4. Showcase an example of exception handling in SQL stored procedure.

Ans-

- SQL Server also has an **exception model to handle exceptions and errors that occurs in T-SQL statements**. To handle exceptions in SQL Server we have **TRY..CATCH blocks**.
- We put T-SQL statements in the TRY block and to handle exceptions we write code in a CATCH block. If there is an **error in code within the 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.
- **TRY..CATCH Syntax:-**

```
BEGIN TRY
--T-SQL statements
--or T-SQL statement blocks
END TRY
BEGIN CATCH
--T-SQL statements
--or T-SQL statement blocks
END CATCH
```

- In **exception handling all T-SQL statements are put into a try block**. If all statements execute without any error then everything is OK else control will go to the catch block.
- SQL Server contains the following **two types of exceptions**:

SQL Assignment 4

1. **System Defined Exceptions-** In a System Defined Exception, the exceptions (errors) are generated by the system.

   Example:-

   ```
   Declare val1 int;
   Declare val2 int;
   BEGIN TRY
   Set val1=8;
   Set val2=val1/0;    /* Error Occur Here */
   END TRY
   BEGIN CATCH
   Print 'Error Occur that is:'
   Print Error_Message()
   END CATCH
   ```

   Output:-

   ```
   Error Occur that is:
   Divide by zero error encountered
   ```

2. **User-Defined Exceptions-** This type of exception is user generated, not system generated.

   Example:-

   ```
   Declare val1 int;
   Declare val2 int;
   BEGIN TRY
   Set val1=8;
   Set val2=val1%2;
   if val1=1
   Print ' Error Not Occur'
   else
   Begin
   Print 'Error Occur';
   Throw 60000,'Number Is Even',5
   End

   END TRY
   BEGIN CATCH
   Print 'Error Occur that is:'
   Print Error_Message()
   END CATCH
   ```

Output:-

```
Error Occur
Error Occur that is:
Number Is Even
```

5. **Create a SQL function to split strings into rows on a given character?**
   Input String: Stephen; peter; berry; Olivier; caroline;

Stephen
Peter
Berry
Oliver
Caroline

Ans-

```
SELECT value
FROM
STRING_SPLIT('Stephen, peter, berry, Olivier, caroline ', ' ');
```

Output:-

Stephen
Peter
Berry
Oliver
Caroline