1.What is a view?

A view in SQL is a saved SQL query that acts as a virtual table. Unlike regular tables, views do not store data themselves. Instead, they dynamically generate data by executing the SQL query defined in the view each time it is accessed. It can fetch data from one or more tables and present it in a customized format, allowing developers to:
Simplify Complex Queries: Encapsulate complex joins and conditions into a single object.
Enhance Security: Restrict access to specific columns or rows.
Present Data Flexibly: Provide tailored data views for different users.

2.Can we update data through a view?

You can update data through a single-table view if you have the Update privilege on the view (see GRANT statement). For a view to be updatable, the query that defines the view must not contain any of the following items:

- Columns in the projection list that are aggregate values
- Columns in the projection list that use the UNIQUE or DISTINCT keyword
- A GROUP BY clause
- A UNION operator

3.What is a materialized view?

A materialized view is a duplicate data table created by combining data from multiple existing tables for faster data retrieval. For example, consider a retail application with two base tables for customer and product data. The customer table contains information like the customer's name and contact details, while the product table contains information about product details and cost. The customer table only stores the product IDs of the items an individual customer purchases. You have to cross-reference both tables to obtain product details of items purchased by specific customers. Instead, you can create a materialized view that stores customer names and the associated product details in a single temporary table. You can build index structures on the materialized view for improved data read performance.

4.Difference between view and table?

Basis	View	Table
Definition	A view is a virtual table that derives its data from one or more base tables through a SQL query.	A table is a physical object that stores data in the form of rows and columns.

Dependency	A view depends on underlying tables or other views for data retrieval.	A table is an independent data object that directly stores information.
Database space	Views do not occupy physical storage space. They only store the query structure.	Tables consume physical space to store data in a database.
Manipulate data	Data cannot be added, updated, or deleted directly from a view.	Data in tables can be added, updated, or deleted using SQL commands like INSERT, UPDATE, and DELETE.
Recreation	Views can be easily recreated or replaced using the CREATE OR REPLACE statement.	A table can only be created or dropped; data in the table can be manipulated.
Aggregation of data	Views can aggregate data from multiple tables or perform complex joins.	Tables store raw, unaggregated data and do not inherently support data aggregation.
table/view relationship	Views can combine multiple tables using joins, unions, etc.	Tables can have primary keys, foreign keys, and indexes to maintain relationships between different tables.

—----------

5.How to drop a view?

The DROP VIEW command deletes a view.

The following SQL drops the "Brazil Customers" view:

Example

```
DROP VIEW [Brazil Customers];
```

—----------

6.Why use views?

A good database should contain views for the given reasons:

- Restricting data access - Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.
- Hiding data complexity - A view can hide the complexity that exists in multiple joined tables.
- Simplify commands for the user - Views allow the user to select information from multiple tables without requiring the users to actually know how to perform a join.
- Store complex queries - Views can be used to store complex queries.
- Rename Columns - Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in a select statement. Thus, renaming helps to hide the names of the columns of the base tables.
- Multiple view facility - Different views can be created on the same table for different users.

7.Can we create indexed views?

The following steps are required to create an indexed view and are critical to the successful implementation of the indexed view:

- Verify the SET options are correct for all existing tables that will be referenced in the view.
- Verify that the SET options for the session are set correctly before you create any tables and the view.
- Verify that the view definition is deterministic.
- Verify that the base table has the same owner as the view.
- Create the view by using the WITH SCHEMABINDING option.
- Create the unique clustered index on the view.

8.How to secure data using views?

Views should be defined as secure when they are specifically designated for data privacy (i.e. to limit access to sensitive data that should not be exposed to all users of the underlying table(s)).

Secure views should not be used for views that are defined solely for query convenience, such as views created to simplify queries for which users do not need to understand the underlying data representation. Secure views can execute more slowly than non-secure views.

9.What are limitations of views?

Limitations of views are:

- You cannot pass parameters to SQL Server views
- Cannot use an Order By clause with views without specifying FOR XML or TOP
- Views cannot be created on Temporary Tables
- You cannot associate rules and defaults with views

10.How does WITH CHECK OPTION work?

In SQL, the WITH CHECK OPTION is used in the context of views to enforce data integrity. When you create a view with the WITH CHECK OPTION clause, it ensures that any INSERT or

UPDATE operation performed through the view adheres to the conditions defined in the view's WHERE clause. This prevents data from being added or modified in a way that would make it inaccessible through the view.
Example:
Without WITH CHECK OPTION:
Copy the codeCREATE VIEW ActiveEmployees AS
SELECT * FROM Employees
WHERE Status = 'Active';

- You can insert or update rows through the view that do not meet the Status = 'Active' condition. These rows will exist in the table but won't be visible in the view.

With WITH CHECK OPTION:
Copy the codeCREATE VIEW ActiveEmployees AS
SELECT * FROM Employees
WHERE Status = 'Active'
WITH CHECK OPTION;

- Now, if you try to insert or update a row through the view where Status is not 'Active', the database will reject the operation.
