Difference Between in SQL Interview Questions

Primary Key	**Foreign Key**
Avoids duplicates and nulls.	References the Primary Key in the parent table.
Acts as a combination of UNIQUE and NOT NULL constraints.	Allows duplicates.
Always attached to the parent table.	Always attached to the child table.
Creates a unique index by default.	Does not create an index by default.
Can be added at the table creation, alter, or column level.	Can be added at the table or alter level only.

DELETE	**TRUNCATE**
Removes specific rows based on a condition.	Removes all rows from the table.
Can be rolled back if used within a transaction.	Cannot be rolled back once executed.
Slower as it logs individual row deletions.	Faster as it does not log individual deletions.
Activates triggers if present.	Does not activate triggers.
Leaves table structure and indexes intact.	Resets table structure and reclaims space.

WHERE	**HAVING**
Filters rows before grouping or aggregation.	Filters groups after aggregation.
Used with SELECT, UPDATE, and DELETE statements.	Used only with SELECT statements.
Cannot use aggregate functions (e.g., SUM, COUNT).	Can use aggregate functions.
Filters individual rows based on conditions.	Filters groups of rows based on aggregate conditions.
Applied before any grouping is done in a query.	Applied after the GROUP BY clause in a query.

RANK	**DENSE_RANK**
Assigns a unique rank to each row within a partition of a result set.	Similar to RANK but without gaps in ranking.
Skips ranks when there are ties (e.g., 1, 2, 2, 4).	Does not skip ranks; consecutive ranks (e.g., 1, 2, 2, 3).
Introduces gaps if there are ties in the ranking.	No gaps; ranks increase consecutively.
Syntax: RANK() OVER (PARTITION BY ... ORDER BY ...).	Syntax: DENSE_RANK() OVER (PARTITION BY ... ORDER BY ...).
Ideal when gaps in rank numbers are acceptable.	Ideal when continuous ranking without gaps is needed.

LEAD	LAG
Retrieves data from the next row in the same result set.	Retrieves data from the previous row in the same result set.
Helps compare the current row with future rows.	Helps compare the current row with previous rows.
Syntax: LEAD (column, offset, default) OVER (PARTITION BY ... ORDER BY ...).	Syntax: LAG (column, offset, default) OVER (PARTITION BY ... ORDER BY ...).
Useful for calculating future trends or forecasts.	Useful for calculating past trends or comparisons.
Offset is by default 1 if not specified.	Offset is by default 1 if not specified.

Primary Key	Surrogate Key
A natural identifier for a record, often derived from real-world data.	An artificial, system-generated identifier, usually numeric (e.g., an auto-increment column).
Must be unique and not null.	Must be unique and not null, often a sequential number.
Values have business meaning and can be used in queries by users.	Values have no business meaning, used only for database management.
Can be a single column or a combination of columns.	Typically, a single column, often an integer.
Can be affected by changes in the data source (e.g., name changes).	Not affected by changes in the data source; remains stable.

VIEW	MATERIALIZED VIEW
Has a logical existence; does not store data.	Has a physical existence; stores data.
Cannot perform DML operations on complex views.	DML operations can be performed.
Fetches data from the base table when queried.	Fetches data from the stored, materialized view.
Cannot be scheduled to refresh automatically.	Can be scheduled to refresh automatically.
Does not store aggregated data.	Can store aggregated data, often used for reporting purposes.

Sub-Query	Correlated Sub-Query
Executed once for the entire parent query.	Executed once for each row of the parent query.
Independent of the parent query.	Depends on the parent query for its values.

Example:
SELECT * FROM Emp
WHERE Deptno IN
(SELECT Deptno FROM Dept);
	Example:
	SELECT e.* FROM Emp e
	WHERE Sal >=
	(SELECT AVG(Sal) FROM Emp a
	WHERE a.Deptno = e.Deptno
	GROUP BY a.Deptno);

Stored Procedure	Function
May or may not return values.	Must return at least one output value.
Can return multiple values using OUT parameters.	Typically returns a single value.
Used to implement complex business logic and operations.	Primarily used for calculations and data manipulation.
Pre-compiled and optimized for performance.	Not pre-compiled; parsed and compiled at runtime.
Can accept multiple arguments.	Can accept multiple arguments but returns a single result.
Mainly used to process tasks and perform operations.	Mainly used to compute values.
Cannot be directly invoked from SQL statements (e.g., SELECT).	Can be invoked from SQL statements (e.g., SELECT).

Triggers	Stored Procedures
No need to execute manually; fired automatically based on events.	Need to be executed manually or called explicitly.
Execute automatically when an INSERT, UPDATE, or DELETE is issued.	Typically used for performing tasks and operations.
Primarily used for tracing, auditing, and enforcing business rules.	Used to encapsulate logic and perform operations on data.
Part of DML events on the table.	Can run independently and be executed from various contexts.
Cannot be executed from stored procedures.	Can have parameters and be executed from other stored procedures.
Cannot have parameters.	Can have parameters.
Event-driven and attached to a specific table or database object.	A compiled collection of SQL statements or programs.

OLTP	OLAP
Designed for managing transactional data.	Designed for analytical and reporting purposes.
Handles a large number of short online transactions (e.g., INSERT, UPDATE, DELETE).	Handles complex queries involving data aggregation and analysis.
Data is highly normalized to reduce redundancy.	Data is often denormalized to optimize query performance.
Focuses on speed and efficiency for daily operations.	Focuses on query performance and data retrieval speed for analysis.
Typical operations include order entry, payments, and customer management.	Typical operations include data mining, forecasting, and business reporting.
Databases are typically smaller in size.	Databases are typically large, integrating data from multiple sources.
Examples: Banking systems, e-commerce websites.	Examples: Data warehouses, business intelligence systems.

Data Mart	Data Warehouse
Usually sponsored at the department level with a specific focus or subject.	A "Subject-Oriented, Integrated, Time-Variant, Non-Volatile" collection of data supporting decision-making.
Used at a business division or department level.	Used at an enterprise level.
A subset of data from a Data Warehouse, built for specific user groups.	An integrated consolidation of data from various sources for strategic and tactical decision-making.
Provides a focused subset of data, which can improve privacy, performance, and clarity.	Provides a comprehensive and coherent view of the business, integrating data across the enterprise.
Developed to address specific issues or needs within a department.	Developed to support overall business intelligence and decision-making across the organization.

Star Schema	Snowflake Schema
Simplest data warehouse schema.	More complex data warehouse model.
Each dimension is represented in a single table; no hierarchies between dimensions.	At least one hierarchy exists between dimension tables.
Contains a central fact table surrounded by dimension tables.	Contains a central fact table surrounded by normalized dimension tables.
Only one join is needed between the fact table and dimension tables.	Requires multiple joins due to relationships between dimension tables.
Optimizes performance with simple queries and fast response times.	Normalizes dimensions to eliminate redundancy, leading to more complex queries and potentially reduced performance.