**SQL:**

1. Can you explain the concept of query execution plans and how they can be analyzed and optimized?

Solution- A query execution plan is a roadmap or a set of steps that the database engine follows to execute a particular SQL query efficiently.

1. What are the three degrees of normalization and how is normalization done in each degree?

Solution- Normalization is the process of organizing data in a database efficiently, aiming to reduce redundancy and dependency.

**First Normal Form (1NF)**:

**Second Normal Form (2NF)**:

**Third Normal Form (3NF)**:

1. Can you explain a scenario where using derived columns can improve query performance?

Solution-OLTP- OLTP systems are designed for transaction-oriented applications, where the emphasis is on quick and efficient transaction processing. These systems are optimized for inserting, updating, and deleting small amounts of data in real-time

SELECT

(UnitPrice \* Quantity) - (UnitPrice \* Quantity \* Discount) AS TotalSaleAmount

FROM

SalesTransactions;

In this scenario, the calculation for **TotalSaleAmount** is performed dynamically in the query for each row in the **SalesTransactions** table. This can lead to increased query execution time, especially when dealing with large datasets

ALTER TABLE SalesTransactions

ADD TotalSaleAmount AS (UnitPrice \* Quantity) - (UnitPrice \* Quantity \* Discount) PERSISTED;

In this scenario, a derived column named **TotalSaleAmount** is added to the **SalesTransactions** table, and the computation logic is defined as part of the column definition. The **PERSISTED** keyword indicates that the derived column's values are physically stored in the database

Since the computation is performed and stored in advance, querying the derived column (**TotalSaleAmount**) is faster compared to calculating it dynamically for each row during query execution

1. What are the differences between OLTP and OLAP?Can you provide an example of a real-world application that requires OLTP processing and another that requires OLAP processing?

Solutions-

1. How would you determine which type of join to use based on the requirements of a specific query?
2. Solution- **Inner Join**:
   * Returns only the rows that have matching values in both tables based on the join condition.
   * Use when you only need the rows that have matches in both tables and want to exclude unmatched rows.
   * It's the most common type of join and is suitable for retrieving related information from two or more tables.
3. **Left Join (or Left Outer Join)**:
   * Returns all rows from the left table and the matched rows from the right table. If there is no match, it returns NULL values for the columns from the right table.
   * Use when you want to include all rows from the left table regardless of whether there are matches in the right table.
   * This is useful when you need to retrieve all records from one table and only matching records from the other.
4. **Right Join (or Right Outer Join)**:
   * Returns all rows from the right table and the matched rows from the left table. If there is no match, it returns NULL values for the columns from the left table.
   * Use when you want to include all rows from the right table regardless of whether there are matches in the left table.
   * It's less commonly used than left joins but can be helpful when the right table is considered the primary table.
5. **Full Outer Join**:
   * Returns all rows from both tables and matches rows from both tables where available. If there is no match, it returns NULL values for the columns from the non-matching table.
   * Use when you need to include all rows from both tables, regardless of whether there are matches.
   * It's useful for combining data from two tables where some rows may not have matches in the other table.
6. **Cross Join (or Cartesian Join)**:
   * Returns the Cartesian product of the two tables, meaning it combines every row from the first table with every row from the second table.
   * Use with caution, as it can result in a large number of rows and is typically used for specific scenarios such as generating combinations or permutations.
7. How to delete DUPLICATE records from a table Using ROWID and ROW\_NUMBER Analytic Function?

**PowerBI:**

1. How can you create a slicer in Power BI that dynamically filters multiple visuals?

Solution- **Open Power BI Desktop**: Launch Power BI Desktop and open the report want to work on.

**Create or Select Visuals**: Ensure that you have multiple visuals (charts, tables, etc.) in your report that you want to filter dynamically.

**Add a Slicer**:

* + Click on the "Slicer" visualization icon from the Visualizations pane.
  + Drag the field you want to use for filtering from the Fields pane onto the Values area of the slicer.

1. When would you choose to use Power BI DirectQuery mode instead of Import mode?

**Solution**- **Real-time Data**: DirectQuery mode is preferable when you need real-time access to the latest data in your source system. In DirectQuery mode, Power BI retrieves data directly from the data source each time a query is executed, ensuring that users always see the most up-to-date information.

**Large Datasets**: Import mode is suitable for smaller datasets that can fit into memory. However, if your dataset is large and exceeds the memory capacity of your Power BI environment, DirectQuery mode allows you to analyze data without the need to load it entirely into memory. This is especially useful for scenarios where the dataset is too large to import or where importing would be impractical due to frequent updates

1. Can you explain the concept of row context and filter context in DAX, and how they affect calculations?

Solution- **Row Context:** Row context is the context in which a calculation is evaluated row by row within a table or table expression. When DAX operates within row context, it evaluates expressions for each row in the current context. For example, when you reference columns in a calculated column or use iterators like **SUMX()** or **FILTER()**, DAX applies calculations within the row context.

**Example:** Suppose you have a table of sales transactions with columns **SalesAmount** and **Quantity**. If you create a calculated column **[TotalSales]** with the formula **[SalesAmount] \* [Quantity]**, DAX evaluates this formula for each row in the table, multiplying **SalesAmount** by **Quantity** for each transaction.

**Filter Context:** Filter context, on the other hand, is the context in which calculations are evaluated based on the current filters applied to the data. Filter context is dynamic and can change based on interactions with visuals, slicers, or explicit filters applied within DAX formulas. When you create measures or calculated columns, DAX evaluates them within the filter context.

**Example:** Suppose you have a measure **[TotalSales]** defined as **SUM(Sales[SalesAmount])**. When you place this measure on a visual, such as a bar chart, and apply a filter for a specific year, the filter context restricts the calculation of **[TotalSales]** to only include sales transactions from that year.

1. What is the difference in behaviour between the SUM function and the SUMX function when working with related tables in Power BI?

Solution- In Power BI, both the **SUM** and **SUMX** functions are used for summation

1. **SUM Function**:
   * The **SUM** function is an aggregator function that calculates the sum of values in a column within the current filter context.
   * It works well when you want to aggregate values from a single column in the current table or within the current filter context.
   * When used with related tables, **SUM** ignores relationships and considers only the current table's context. It won't follow relationships to other tables.
2. **SUMX Function**:
   * The **SUMX** function is an iterator function that iterates over each row of a table, evaluates an expression for each row, and sums the results.
   * It's useful for performing calculations that involve multiple tables or require row-level context, such as calculations across related tables.
   * When used with related tables, **SUMX** follows relationships and evaluates the expression within the context of each related row. It can aggregate values from related tables based on the current filter context
3. What are some considerations you need to keep in mind to ensure query folding occurs during data loading in Power Query?

Solution- Query folding in Power Query is a feature that optimizes query performance by pushing certain data transformation operations back to the data source

**Supported Operations**

**Query Dependencies**

**Native Query Language**

**Data Source Capabilities**

**Custom Functions and Expressions**