SQL INTERVIEW QUESTIONS

Q finding the second-highest salary

Solution 1: Using a Subquery with `LIMIT` and `OFFSET`

```
SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1;
```

```
SELECT DISTINCT saleamount
FROM sales
ORDER BY saleamount DESC
OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;
```

```
SELECT DISTINCT(Breakup_Amount) FROM #temp1
ORDER BY Breakup_Amount DESC
OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;
```

Solution 2: Using `ROW_NUMBER()` Window Function

SELECT salary

FROM (

  SELECT salary, ROW_NUMBER() OVER (ORDER BY salary DESC) as row_num

  FROM employees

) AS ranked_salaries

WHERE row_num = 2;

```
SELECT SaleAmount ,SaleID
FROM (
    SELECT
    SaleID,
        SaleAmount,
        ROW_NUMBER() OVER (ORDER BY SaleAmount DESC) AS RowNum
    FROM Sales
) AS ranked_salaries
WHERE RowNum = 4;
```

```
WITH rankedSaleAmount AS (

          SELECT
   SaleAmount,
        dense_rank() OVER (ORDER BY SaleAmount DESC) AS RowNum
    FROM Sales
```

```
)

SELECT SaleAmount
FROM rankedSaleAmount
WHERE RowNum = 2;
```

Solution 3: Using `DENSE_RANK()` Window Function

```
SELECT salary
FROM (
   SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) as rank
   FROM employees
) AS ranked_salaries
WHERE rank = 2;
```

Solution 4: Using a Subquery with `MAX()`

```
SELECT MAX(salary)
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

```
DELETE FROM sales
WHERE saleID NOT IN (
   SELECT MIN(saleID)
   FROM sales
   GROUP BY salesperson
);
==========================================
--Find duplicates (example assumes duplicate `OrderAmount` values, not the full scenario)
WITH CTE_Duplicates AS (
   SELECT OrderID, OrderAmount,
      ROW_NUMBER() OVER (PARTITION BY OrderAmount ORDER BY OrderID) AS RowNum
   FROM Orders
```

)

DELETE FROM CTE_Duplicates

WHERE RowNum > 1;

```sql
-- Calculate running totals
SELECT
    SaleID,
    SaleDate,
    SalesPerson,
    SaleAmount,
    SUM(SaleAmount) OVER (ORDER BY SaleDate) AS RunningTotal
FROM Sales2
ORDER BY SaleDate;
```

SERIES

```sql
WITH Numbers AS (
    SELECT 1 AS MYNumber
    UNION ALL
    SELECT MYNumber + 1
    FROM Numbers
    WHERE MYNumber <10
)
SELECT MYNumber
FROM Numbers
```

Star Schema

1. **Structure**:
    - In a star schema, the **fact table** (which contains the measures or metrics of the business) is at the center, and it is connected directly to multiple **dimension tables**.
    - Each dimension table is denormalized, meaning it contains all the necessary attributes without separating them into multiple related tables.
2. **Characteristics**:
    - Dimension tables are not normalized (i.e., there is redundancy in data).
    - Simpler design, easier to understand and navigate.
    - Faster query performance due to fewer joins.
3. **When to Use**:
    - When the primary goal is to optimize read operations and improve query performance.
    - Suitable for data marts where speed and simplicity are more critical than storage efficiency.

Snowflake Schema

1. **Structure**:
    - The snowflake schema is an extension of the star schema where the dimension tables are further normalized into multiple related tables. Each dimension can have multiple related tables, making the schema resemble a snowflake.
2. **Characteristics**:
    - Dimension tables are normalized (data is split into related tables to eliminate redundancy).
    - More complex design with more tables and more joins.
    - Reduced data redundancy and better data integrity but might lead to slower query performance due to additional joins.
3. **When to Use**:
    - When storage optimization is more important, and there is a need for better data integrity and normalization.
    - Suitable for large and complex data warehouses where data storage efficiency is crucial.

==========

When to Use Each Schema

- **Star Schema**:
    - **Use Case**: Ideal for simpler, smaller data marts where read performance is more important than storage efficiency.
    - **Advantages**: Simple design, faster read operations, easier to understand and implement.
- **Snowflake Schema**:
    - **Use Case**: Suitable for larger, more complex data warehouses where data storage optimization, data integrity, and maintenance are more important.
    - **Advantages**: Reduced data redundancy, more structured data, better data integrity, but may require more complex queries.

1. Requirement Analysis and Planning
2. Data Modeling
3. ETL (Extract, Transform, Load) Process Design
4. Build and Populate Dimension Tables
5. Build and Populate Fact Tables
6. Data Validation and Quality Assurance
7. Create Indexes and Optimize Performance
8. Build Aggregations and Views
9. Implement Security and Access Controls
10. Develop Reports, Dashboards, and Analytics
11. Testing and Validation
12. Deployment and Maintenance

In **SQL Server Integration Services (SSIS)**, **containers** are logical groupings that help manage and execute tasks in an SSIS package. They provide structure to the SSIS package and allow for better control over the flow of execution, especially in complex ETL (Extract, Transform, Load) processes.

Types of Containers in SSIS

1. **Sequence Container**
2. **For Loop Container**
3. **Foreach Loop Container**
4. **Task Host Container**

A **Fact Table** is a central table in a star schema or snowflake schema of a data warehouse. It stores quantitative data (measurable facts) for analysis and contains foreign keys to dimension tables

A **Factless Fact Table** is a type of fact table that does not have any measurable facts or numeric data. Instead, it is used to capture the occurrence of events or describe many-to-many relationships between dimension tables.

Yes, a **fact table** can contain `NULL` values, but it is generally not recommended or common practice in data warehousing. `NULL` values may exist in a fact table under certain circumstances, but they are often avoided to ensure data quality and consistency.

In simple terms, SSIS (SQL Server Integration Services) has **four main parts**:

1. **Control Flow:** This is the brain of SSIS, deciding what tasks to run and in what order (like a to-do list).
2. **Data Flow:** This is where the real work happens—it moves data from one place to another, cleaning and changing it along the way.
3. **Event Handlers:** These are like safety nets; they catch problems or special events and let you take action when something goes wrong.
4. **Package Explorer:** Think of it as a map; it shows everything in your SSIS package and how it's all connected.

In **SQL Server Integration Services (SSIS)**, the concepts of **Control Flow** and **Data Flow** are core components used to build ETL (Extract, Transform, Load) processes. Understanding these two components is crucial for creating effective and efficient SSIS packages.

1. Control Flow

Control Flow is the backbone of an SSIS package. It defines the **workflow** of the package and controls the **order of execution** of tasks. It deals with the flow of operations and determines **what** to do and **when** to do it. It includes tasks like executing SQL statements, sending emails, processing files, running scripts, and even managing error handling.

Key Elements of Control Flow

- **Tasks**: The individual operations performed in the Control Flow. Examples include the Execute SQL Task, Data Flow Task, Script Task, etc.
- **Precedence Constraints**: The lines connecting tasks that determine the order in which tasks are executed. They can be based on success, failure, or completion of the preceding task.
- **Containers**: Logical groupings of tasks (e.g., Sequence Container, For Loop Container) that provide structure to the workflow and help manage task execution.

Example of Control Flow

Imagine you need to automate the following process:

1. **Download a file** from an FTP server.
2. **Extract data** from the downloaded file.
3. **Load data** into a SQL Server database.
4. **Send an email** notification upon completion.

In SSIS, the **Control Flow** would look like this:

- Use an **FTP Task** to download the file.
- Use a **Data Flow Task** to handle the data extraction and loading.
- Use a **Send Mail Task** to send the notification.
- Use **Precedence Constraints** to connect these tasks in the required order.

The Control Flow would manage the entire process and ensure tasks are executed in the right sequence.

2. Data Flow

Data Flow in SSIS deals specifically with the **extraction, transformation, and loading (ETL)** of data. While the Control Flow handles the overall process logic, the Data Flow focuses on the **movement and transformation of data** from source to destination.

Key Elements of Data Flow

- **Sources**: Define where the data is coming from (e.g., SQL Server, Excel, Flat File, Oracle).

- **Transformations**: Define how the data is manipulated or transformed (e.g., Data Conversion, Conditional Split, Aggregation).
- **Destinations**: Define where the data is going (e.g., SQL Server, Excel, Flat File, Data Warehouse).

Example of Data Flow

Imagine you need to perform the following ETL operations:

1. **Extract data** from a flat file (CSV).
2. **Transform data** by converting data types and filtering rows.
3. **Load data** into a SQL Server table.

In SSIS, the **Data Flow** would look like this:

- Use a **Flat File Source** to read data from the CSV file.
- Use a **Data Conversion Transformation** to convert data types as needed.
- Use a **Conditional Split Transformation** to filter rows based on specific conditions.
- Use an **OLE DB Destination** to load the transformed data into a SQL Server table.

The Data Flow would manage the movement and transformation of data within this specific ETL process.