#### **Question 1 : Assume that you are a data engineer for company ABC The company wanted to do cloud migration from their on-premises to Microsoft Azure cloud. You probably will use the Azure data factory for this purpose. You have created a pipeline that copies data of one table from on-premises to Azure cloud. What are the necessary steps you need to take to ensure this pipeline will get executed successfully?**

Ans. As a data engineer, here are the necessary steps you need to take to ensure the pipeline will get executed successfully:

Ensure that you have created a linked service for the on-premises database and another linked service for the Azure SQL database.

Create a dataset for the source table in the on-premises database, and another dataset for the destination table in the Azure SQL database.

Create an Azure Data Factory pipeline with a copy activity that uses the source and destination datasets and linked services that you created in the previous steps.

Make sure that you have granted the necessary permissions to the pipeline to access the on-premises database and the Azure SQL database.

Schedule the pipeline to run at a time when there is low traffic on the network to minimize the possibility of network congestion.

Monitor the pipeline as it runs to ensure that it is executing as expected and that there are no errors or issues.

Set up alerts and notifications to notify you if the pipeline fails or if there are any issues that require your attention.

Ensure that you have a backup plan in place in case of any unexpected errors or issues during the pipeline execution. This may include having a backup copy of the data or having a secondary pipeline that can be activated if the primary pipeline fails.

#### **Question 2: Assume that you are working for a company ABC as a data engineer. You have successfully created a pipeline needed for migration. This is working fine in your development environment. how would you deploy this pipeline in production without making any or very minimal changes?**

Ans: As a data engineer, here are the steps you can take to deploy the pipeline from your development environment to production with minimal changes:

Ensure that the development environment and the production environment have the same Azure Data Factory version and integration runtime.

Create the linked services, datasets, and pipeline in the production environment using the same names and configurations as in the development environment. This ensures that the pipeline references the correct resources in the production environment.

Export the pipeline JSON file from the development environment.

Modify the JSON file to reference the linked services and datasets in the production environment. This includes updating the connection strings and other environment-specific settings.

Import the modified pipeline JSON file into the production environment.

Verify that the pipeline runs successfully in the production environment by running a test job with a small data set.

If necessary, update the pipeline settings or configurations to optimize performance or address any issues that arise during testing.

Once the pipeline is running successfully in the production environment, set up appropriate monitoring and alerting to detect any issues or errors that may occur during runtime.

Make sure to document the deployment process and any configuration changes made in the production environment. This will help ensure consistency and facilitate future updates or modifications to the pipeline.

#### **Question 3: Assume that you have around 1 TB of data stored in Azure blob storage . This data is in multiple csv files. You are asked to do couple of transformations on this data as per business logic and needs, before moving this data into the staging container. How would you plan and architect the solution for this given scenario. Explain with the details.**

Ans: As a data engineer, here is how I would plan and architect a solution to perform transformations on 1 TB of data stored in Azure Blob Storage:

Create an Azure Data Factory pipeline that reads the CSV files from the source blob container and transforms the data according to the business logic.

Use Azure Databricks to perform the data transformation using Spark. Databricks provides an interactive workspace that allows you to write and run Spark jobs using Python or Scala.

Use the Azure Blob Storage connector in Databricks to read the CSV files from the source blob container.

Use the Apache Spark API to perform the necessary data transformations. Spark is a distributed computing framework that is highly scalable and can handle large volumes of data.

Write the transformed data to a new set of CSV files in a temporary blob container.

Use Azure Data Factory to move the transformed CSV files from the temporary blob container to the staging container.

To optimize performance, consider using Azure Data Factory's Parallel Copy feature, which enables parallel transfer of data between Blob Storage containers.

Configure the pipeline to run on a schedule or triggered by an event such as the availability of new CSV files in the source container.

Monitor the pipeline execution to ensure that it is running as expected and that there are no errors or issues.

Finally, consider implementing a data quality check to ensure that the transformed data meets the expected data quality standards. This could involve using Azure Data Factory's data validation feature or writing custom validation scripts using Databricks or Azure Functions.

#### **Question 4: Assume that you have an IoT device enabled on your vehicle. This device from the vehicle sends the data every hour and this is getting stored in a blob storage location in Microsoft Azure. You have to move this data from this storage location into the SQL database. How would design the solution explain with reason.**

Ans: As a data engineer, here's how I would design a solution to move data from a blob storage location to a SQL database:

Create an Azure Data Factory pipeline that reads the data from the blob storage location and moves it to the SQL database.

Use the Azure Blob Storage connector in Azure Data Factory to read the data from the blob storage location. This connector enables you to specify the file path, file format, and other settings needed to read the data from the blob storage location.

Use the Azure SQL Database connector in Azure Data Factory to write the data to the SQL database. This connector enables you to specify the table name, database name, credentials, and other settings needed to write the data to the SQL database.

Configure the pipeline to run on a schedule or triggered by an event such as the availability of new data in the blob storage location.

To optimize performance, consider using Azure Data Factory's Parallel Copy feature, which enables parallel transfer of data between Blob Storage and SQL Database.

Consider implementing data quality checks and data transformation, such as data cleansing, data formatting, or data aggregation, in the pipeline before writing data to the SQL database.

Monitor the pipeline execution to ensure that it is running as expected and that there are no errors or issues.

Ensure that proper authentication and authorization mechanisms are in place to restrict access to the SQL database and blob storage location.

Consider archiving the data in the blob storage location after it has been successfully transferred to the SQL database to minimize storage costs.

Finally, implement backup and disaster recovery plans to ensure the data's safety in case of any unexpected failures. This could involve regular backups of the SQL database, retention of data in the blob storage location, or replicating data across multiple regions or cloud providers.

#### **Question 5: Assume that you are doing some R&D over the data about the COVID across the world. This data is available by some of the public forum which is exposed as REST api. How would you plan the solution in this scenario?**

Ans: As a data engineer, here's how I would plan a solution to collect and analyze data about COVID across the world using a REST API:

Identify the REST API that provides the data about COVID across the world. Ensure that the API is reliable, provides real-time or near real-time data, and has the necessary data fields and attributes needed for analysis.

Determine the frequency of data collection and the amount of data to be collected. Based on this, choose an appropriate tool or programming language to query the API and store the data.

Use a data storage solution such as Azure Blob Storage or Azure Data Lake Storage to store the data collected from the REST API. Choose the appropriate data format such as JSON or CSV based on the data structure.

Set up a data pipeline using Azure Data Factory or Azure Functions to collect data from the REST API and store it in the data storage solution. Use a data ingestion framework such as Apache NiFi to transform the data into the desired format.

Consider implementing data quality checks and data transformation, such as data cleansing, data formatting, or data aggregation, in the pipeline before storing the data.

Use a data visualization tool such as Power BI or Tableau to create dashboards and reports that provide insights into the COVID data across the world.

Ensure that proper authentication and authorization mechanisms are in place to access the REST API and store the data in the data storage solution.

Monitor the data pipeline and storage solution to ensure that they are running as expected and that there are no errors or issues.

Consider implementing backup and disaster recovery plans to ensure the data's safety in case of any unexpected failures. This could involve regular backups of the data storage solution, replication of data across multiple regions or cloud providers, or setting up a hot standby for the pipeline.

Finally, ensure that the data collection and analysis comply with relevant regulations such as GDPR and CCPA and respect the privacy of individuals.