AZURE DATA ENGINEERING INTERVIEW Q & A - COMPANYWISE

(This document contains interview questions and answers which I experienced personally in all my 20+ interviews)

List of Companies:

- **Tiger Analytics**
- **Tredence**
- **Fractal**
- **LatentView**
- **Deloitte**
- **KPMG**
- **PWC**
- **Infosys**
- **TCS**
- **Cognizant**
- **EY**
- **Persistent**
- **Ecolabs**
- **FedEx**

Tiger Analytics

1. Explain lazy evaluation in PySpark.
2. How does caching work in PySpark?
3. What is the difference between wide and narrow transformations?
4. How do you optimize query performance in Azure SQL Database?
5. Describe the process of integrating ADF with Azure Synapse Analytics.
6. How do you handle schema evolution in Azure Data Lake?
7. Write a SQL query to find the nth highest salary in a table.
8. How do you implement CI/CD pipelines for deploying ADF and Databricks solutions?
9. Write PySpark code to calculate the total sales for each product category.
10. Explain how broadcast joins improve performance in PySpark.
11. Describe the role of the driver and executor in Spark architecture.
12. How do you manage and monitor ADF pipeline performance?
13. Write a SQL query to find employees with salaries greater than the department average.
14. Explain the concept of Delta Lake and its advantages.
15. How do you implement schema drift handling in ADF?
16. Write Python code to check if a number is a palindrome.
17. What is the significance of Z-ordering in Delta tables?
18. How do you handle incremental data load in Databricks?
19. Explain Adaptive Query Execution (AQE) in Spark.
20. How do you optimize data partitioning in ADLS?
21. Describe the process of creating a data pipeline for real-time analytics.
22. Write PySpark code to perform a left join between two DataFrames.
23. What are the security best practices for Azure Data Lake?
24. Explain the use of Integration Runtime (IR) in ADF.
25. How do you design a fault-tolerant architecture for big data processing?

Tredence

1. Difference between groupByKey and reduceByKey in PySpark.
2. How to register a User Defined Function (UDF) in PySpark?
3. What are Delta logs, and how to track data versioning in Delta tables?
4. How do you monitor and troubleshoot ADF pipeline failures?
5. What is the use of Delta Lake, and how does it support ACID transactions?
6. Explain the concept of Managed Identity in Azure and its use in data engineering.
7. Write a SQL query to find employees earning more than their manager.
8. Describe the process of migrating on-premises databases to Azure SQL Database.
9. Write PySpark code to filter records based on a condition.
10. How do you implement error handling in ADF pipelines?
11. Explain the role of SparkSession in PySpark.
12. How do you optimize storage costs in ADLS?
13. Write a SQL query to find duplicate records in a table.
14. What is the purpose of caching in PySpark, and how is it implemented?
15. Describe the process of integrating ADF with Databricks for ETL workflows.
16. Write Python code to count the frequency of words in a string.
17. How do you handle schema evolution in Delta Lake?
18. Explain the difference between streaming and batch processing in Spark.
19. How do you secure data pipelines in Azure?
20. What are the best practices for managing large datasets in Databricks?

Deloitte

1. What is Z-ordering in Spark?
2. Explain the difference between Spark SQL and PySpark DataFrame APIs.
3. How to implement incremental load in ADF?
4. How do you handle large-scale data ingestion into ADLS?
5. Write Python code to split a name column into firstname and lastname.
6. What are fact and dimension tables in data modeling?
7. How do you design and implement data pipelines using Azure Data Factory?
8. Explain the concept of PolyBase in Azure SQL Data Warehouse.
9. Write a SQL query to calculate the cumulative sum of a column.
10. How do you manage partitioning in PySpark?
11. Explain the use of Delta Lake for data versioning.
12. How do you monitor and troubleshoot Spark jobs?
13. Write a SQL query to find employees with the highest salary in each department.
14. How do you optimize joins in PySpark for large datasets?
15. Describe the process of setting up CI/CD for Azure Data Factory.
16. Write Python code to reverse a string.
17. What are the key features of Databricks notebooks?
18. How do you handle late-arriving data in ADF?
19. Explain the concept of Data Lakehouse.
20. How do you implement disaster recovery for ADLS?

Fractal

1. How do autoscaling clusters work in Databricks?
2. What are the types of Integration Runtimes (IR) in ADF?
3. Difference between Blob Storage and Azure Data Lake Storage (ADLS).
4. How do you integrate Databricks with Azure DevOps for CI/CD pipelines?
5. Write a SQL query to convert row-level data to column-level data using pivot.
6. How do you ensure data quality and validation in ADLS?
7. Explain the use of hierarchical namespaces in ADLS.
8. Describe the process of setting up and managing an Azure Synapse Analytics workspace.
9. Write PySpark code to calculate the average salary by department.
10. How do you implement streaming pipelines in Databricks?
11. Explain the purpose of Delta Lake checkpoints.
12. How do you handle data encryption in ADLS?
13. Write a SQL query to find the top 3 customers by sales.
14. How do you optimize Spark jobs for better performance?
15. Describe the role of triggers in ADF pipelines.
16. Write Python code to find the largest number in a list.
17. How do you implement parallel processing in PySpark?
18. Explain the concept of lineage in data pipelines.
19. How do you manage access control in Azure Data Lake?
20. What are the challenges in integrating on-premises data with Azure services?

Infosys

1. What is the difference between a job cluster and an interactive cluster in Databricks?
2. How to copy all tables from one source to the target using metadata-driven pipelines in ADF?
3. How do you implement data encryption in Azure SQL Database?
4. Write Python code to generate Fibonacci numbers.
5. What are the best practices for managing and optimizing storage costs in ADLS?
6. How do you implement security measures for data in transit and at rest in Azure?
7. Describe the role of triggers and schedules in Azure Data Factory.
8. How do you optimize data storage and retrieval in Azure Data Lake Storage?
9. Write a SQL query to find employees with no manager assigned.
10. How do you implement data deduplication in PySpark?
11. Explain the concept of Delta Lake compaction.
12. How do you monitor ADF pipeline performance?
13. Write a SQL query to find the second-highest salary in a table.
14. How do you implement incremental load in Databricks?
15. Describe the role of Azure Key Vault in securing sensitive data.
16. Write Python code to sort a list of dictionaries by a key.
17. How do you handle schema evolution in ADF?
18. Explain the concept of shufling in Spark.
19. How do you manage metadata in Azure Data Lake?
20. What are the key considerations for designing scalable pipelines in ADF?

EY

1. How to handle null values in PySpark (drop/fill)?
2. What is AQE (Adaptive Query Execution) in Databricks?
3. How do you handle error handling in ADF using retry, try-catch blocks, and failover mechanisms?
4. How to track file names in the output table while performing copy operations in ADF?
5. Write a SQL query to display the cumulative sum of a column.
6. Explain the role of Azure Key Vault in securing sensitive data.
7. How do you manage and automate ETL workflows using Databricks Workflows?
8. Describe the process of setting up disaster recovery for ADLS.
9. Explain the difference between narrow and wide transformations in PySpark.
10. How do you optimize PySpark jobs for large datasets?
11. Write a SQL query to find the average salary for each department.
12. What is the role of Delta Lake in modern data architectures?
13. How do you monitor and debug ADF pipelines?
14. Write PySpark code to perform an inner join between two DataFrames.
15. How do you manage schema drift in ADF?
16. Describe the concept of fault tolerance in Spark.
17. Write Python code to calculate the factorial of a number.
18. How do you handle incremental data loads in ADLS?
19. What are the security features in Azure Synapse Analytics?
20. Explain the concept of partitioning in PySpark.
21. How do you implement real-time data processing in Databricks?
22. Write a SQL query to find duplicate records in a table.
23. How do you integrate Azure Key Vault with ADF pipelines?
24. What are the best practices for optimizing storage costs in ADLS?
25. How do you implement CI/CD for Azure Synapse Analytics?
26. Explain the role of Integration Runtime in ADF.
27. How do you secure sensitive data in Azure?
28. Describe the process of creating a data pipeline for real-time analytics.

Persistent

1. Explain broadcast join in PySpark.
2. How to create a rank column using the Window function in PySpark?
3. What is the binary copy method in ADF, and when is it used?
4. How do you monitor and optimize performance in Azure Synapse?
5. Write Python code to identify duplicates in a list and count their occurrences.
6. What are the key features of Azure DevOps?
7. How do you handle schema drift in ADF?
8. Explain the concept of denormalization and when it should be used.

Cognizant

1. Difference between repartition and coalesce in PySpark.
2. How to persist and cache data in PySpark?
3. How do you use Azure Logic Apps to automate data workflows in SQL databases?
4. What are the differences between ADLS Gen1 and Gen2?
5. Write a SQL query to find gaps in a sequence of numbers.
6. How do you ensure high availability and disaster recovery for Azure SQL databases?
7. Explain the role of pipelines in Azure DevOps.
8. How do you implement data masking in ADF for sensitive data?

TCS

1. How many jobs, stages, and tasks are created during a Spark job execution?
2. What are the activities in ADF (e.g., Copy Activity, Notebook Activity)?
3. How do you integrate ADLS with Azure Databricks for data processing?
4. Write Python code to check if a string is a palindrome.
5. How do you implement data governance in a data lake environment?
6. Explain the differences between Azure SQL Database and Azure SQL Managed Instance.
7. How do you monitor and troubleshoot issues in Azure SQL Database?
8. Describe the process of data ingestion in Azure Synapse.

LatentView

1. Explain the purpose of SparkContext and SparkSession.
2. How to handle incremental load in PySpark when the table lacks a last_modified column?
3. How do you use Azure Stream Analytics for real-time data processing?
4. What are the security features available in ADLS (e.g., access control lists, role-based access)?
5. Write a SQL query to remove duplicate rows from a table.
6. How do you manage data lifecycle policies in ADLS?
7. What are the key considerations for designing a scalable data architecture in Azure?
8. How do you integrate Azure Key Vault with other Azure services?

EXL

1. Difference between TRUNCATE and DELETE in SQL.
2. How do you handle big data processing using Azure HDInsight?
3. How to implement parallel copies in ADF using partitioning?
4. Write Python code to replace vowels in a string with spaces.
5. How do you implement data encryption at rest and in transit in ADLS?
6. Describe the use of Azure Synapse Analytics and how it integrates with other Azure services.
7. How do you implement continuous integration and continuous deployment (CI/CD) in Azure DevOps?
8. Explain the role of metadata in data modeling and data architecture.

KPMG

1. How to create and deploy notebooks in Databricks?
2. What are the best practices for data archiving and retention in Azure?
3. How do you connect ADLS (Azure Data Lake Storage) to Databricks?
4. Write a SQL query to list all employees who joined in the last 6 months.
5. How do you implement data validation and quality checks in ADF?
6. Explain the concept of Azure Data Lake and its integration with SQL-based systems.
7. How do you handle exceptions and errors in Python?
8. What is the process of normalization, and why is it required?

PwC

1. Write a PySpark code to join two DataFrames and perform aggregation.
2. What is the difference between wide and narrow transformations in Spark?
3. How do you integrate Azure Synapse Analytics with other Azure services?
4. How do you monitor and troubleshoot data pipelines in Azure Data Factory?
5. Write Python code to generate prime numbers.
6. How do you optimize Python code for better performance?
7. Explain the concept of list comprehensions and provide an example.
8. How do you implement disaster recovery and backup strategies for data in Azure?

Tiger Analytics Q & A

---

1. Explain lazy evaluation in PySpark.

Lazy evaluation means transformations (like map, filter) are not executed immediately. Instead, they're only evaluated when an action (like collect, count) is triggered. This approach optimizes execution by minimizing data passes and enabling the Spark engine to build an efficient execution plan.

2. How does caching work in PySpark?

When you cache a DataFrame using .cache() or .persist(), Spark stores it in memory (or disk if needed) so repeated actions can reuse the same data instead of recomputing. It's useful when you use the same dataset multiple times.

3. What is the difference between wide and narrow transformations?

- **Narrow transformations** (e.g., map, filter) operate on a single partition, no shuffling required.
- **Wide transformations** (e.g., reduceByKey, join) involve data shuffling across nodes and are more expensive.

4. How do you optimize query performance in Azure SQL Database?

- Create proper **indexes** (clustered, non-clustered)
- Use **query hints** and **execution plans**
- Avoid SELECT *
- Optimize with **partitioning** and **statistics updates**
- Monitor via **Query Performance Insight** and **DMVs**

---

5. Describe the process of integrating ADF with Azure Synapse Analytics.

- Use **Linked Services** in ADF to connect to Synapse
- Create **pipelines** to move or transform data
- You can run **stored procedures**, execute **Spark notebooks**, or use **copy activity** to load data
- Monitor with ADF's **Monitor tab**

---

6. How do you handle schema evolution in Azure Data Lake?

Use formats like **Delta Lake** or **Parquet** that support schema evolution. In ingestion, tools like **ADF** or **Databricks Auto Loader** can be configured to merge schema changes (mergeSchema option).

---

7. Write a SQL query to find the nth highest salary in a table.

```
SELECT DISTINCT salary
FROM employees e1
WHERE N-1 = (
  SELECT COUNT(DISTINCT salary)
  FROM employees e2
  WHERE e2.salary > e1.salary
);
```

---

8. How do you implement CI/CD pipelines for deploying ADF and Databricks solutions?

- Use **Azure DevOps/GitHub** for source control
- Integrate ADF with **Git repository**
- Use ARM templates for ADF deployment
- Use **Databricks Repos**, **notebook export/import**, and **databricks-cli**
- Use **release pipelines** for deployment automation

---

9. Write PySpark code to calculate the total sales for each product category.

```
df.groupBy("category").agg(sum("sales").alias("total_sales")).show()
```

---

10. Explain how broadcast joins improve performance in PySpark.

Broadcast joins send a small dataset to all worker nodes, avoiding costly shuffles. Use broadcast() when one of the tables is small enough to fit in memory:

```
from pyspark.sql.functions import broadcast
df.join(broadcast(small_df), "id")
```

---

11. Describe the role of the driver and executor in Spark architecture.

- **Driver**: Coordinates the Spark application; maintains metadata and DAG
- **Executors**: Run tasks on worker nodes, perform computations, and return results

---

12. How do you manage and monitor ADF pipeline performance?

- Use **Monitor tab** for activity runs, trigger runs
- Enable **logging via Log Analytics**
- Use **Activity Duration and Output metrics**
- Implement **retry policies**, **alerts**, and **timeouts**

---

13. Write a SQL query to find employees with salaries greater than the department average.

```
SELECT e.*
FROM employees e
JOIN (
  SELECT department_id, AVG(salary) AS avg_salary
  FROM employees
  GROUP BY department_id
) d ON e.department_id = d.department_id
WHERE e.salary > d.avg_salary;
```

---

14. Explain the concept of Delta Lake and its advantages.

Delta Lake is an open-source storage layer that brings **ACID transactions**, **time travel**, **schema evolution**, and **concurrent writes** to data lakes using formats like Parquet.

---

15. How do you implement schema drift handling in ADF?

Enable **"Auto Mapping"** and check **"Allow schema drift"** in Copy Activity. Use dynamic column mapping when the schema can vary over time.

---

16. Write Python code to check if a number is a palindrome.

```
def is_palindrome(n):
   return str(n) == str(n)[::-1]
```

```
print(is_palindrome(121))  # True
```

---

17. What is the significance of Z-ordering in Delta tables?

Z-ordering organizes data to improve query performance by **clustering related data together**. It reduces I/O during filtering and improves **data skipping**.

---

18. How do you handle incremental data load in Databricks?

- Use **watermarking** or **timestamp columns**
- Filter records where last_updated > last_processed
- Use **merge (upsert)** logic with Delta Lake

---

19. Explain Adaptive Query Execution (AQE) in Spark.

AQE dynamically optimizes query plans based on runtime stats, enabling:

- Dynamically switching join strategies
- Re-optimizing skewed partitions
- Coalescing shuffle partitions

---

20. How do you optimize data partitioning in ADLS?

- Partition by frequently queried columns (e.g., date, region)
- Avoid too many small files (“file size tuning”)
- Use tools like **Azure Data Explorer**, **Databricks**, or **Partition Discovery**

---

21. Describe the process of creating a data pipeline for real-time analytics.

- Use **Event Hubs / IoT Hub** for ingestion
- Use **Stream Analytics** or **Structured Streaming (Databricks)**
- Process and write to **Delta Lake / Cosmos DB / Synapse**
- Visualize using **Power BI**

---

22. Write PySpark code to perform a left join between two DataFrames.

```
df1.join(df2, on="id", how="left").show()
```

---

23. What are the security best practices for Azure Data Lake?

- Use **RBAC + ACLs**
- Enable **Data Encryption (at rest and in transit)**
- Use **Managed Identities** for secure access
- Monitor with **Azure Defender** and **Log Analytics**

---

24. Explain the use of Integration Runtime (IR) in ADF.

Integration Runtime (IR) is the compute infrastructure in ADF. It handles:

- Data movement
- Data transformation
- Supports cloud, self-hosted, and SSIS runtimes

---

25. How do you design a fault-tolerant architecture for big data processing?

- Use **retry logic** and **checkpointing**
- Design for **idempotency**
- Use **Delta Lake** for ACID compliance
- Implement **monitoring**, **alerting**, and **disaster recovery** strategies

---

Tredence Interview Preparation – Q & A

1. Difference between groupByKey and reduceByKey in PySpark.

In PySpark, `groupByKey` groups all values with the same key into a single collection, which can be memory-intensive and cause data shuffling. It's less efficient and should be used when you truly need to group all values.
On the other hand, `reduceByKey` merges values for each key using an associative reduce function, performing the merge locally before shuffling data. This reduces data movement and is generally more efficient for aggregations like sum, count, etc.

2. How to register a User Defined Function (UDF) in PySpark?

You can define a regular Python function and register it as a UDF using `pyspark.sql.functions.udf`.
For example:
```python
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

def square(x):
   return x * x

square_udf = udf(square, IntegerType())
df.withColumn("squared", square_udf(df["value"]))
```

3. What are Delta logs, and how to track data versioning in Delta tables?

Delta logs are stored in the `_delta_log` directory inside a Delta Lake table folder. These logs track every change (add, remove, update) in JSON and parquet files.
You can use `DESCRIBE HISTORY table_name` in Databricks or Spark SQL to view the full version history of a Delta table.

4. How do you monitor and troubleshoot ADF pipeline failures?

You can monitor pipelines in the Azure Data Factory Monitoring tab. It shows activity runs, duration, errors, and status. You can also set up alerts via Azure Monitor, and use log analytics or custom logging to capture detailed error info.

5. What is the use of Delta Lake, and how does it support ACID transactions?

Delta Lake adds ACID transaction capabilities to data lakes. It ensures consistency by using transaction logs and locking mechanisms. So even in distributed environments, reads and writes remain reliable. It also supports time travel, schema enforcement, and rollback.

6. Explain the concept of Managed Identity in Azure and its use in data engineering.

Managed Identity provides an automatically managed identity in Azure Active Directory. It allows ADF, Databricks, or Azure Functions to authenticate to Azure services like ADLS or Key Vault securely without needing credentials in code.

7. Write a SQL query to find employees earning more than their manager.

```sql
SELECT e.name
FROM Employees e
JOIN Employees m ON e.manager_id = m.id
WHERE e.salary > m.salary;
```

8. Describe the process of migrating on-premises databases to Azure SQL Database.

You typically use the Data Migration Assistant (DMA) or Azure Database Migration Service (DMS). First, assess compatibility using DMA, then provision your Azure SQL DB, create the schema, and migrate data using DMS with minimal downtime.

9. Write PySpark code to filter records based on a condition.

```python
filtered_df = df.filter(df['age'] > 30)
filtered_df.show()
```

10. How do you implement error handling in ADF pipelines?

You can use 'If Condition', 'Until', and 'Try-Catch'-like logic using 'Failure' dependencies and 'custom activity' with parameters to log failures. ADF also supports sending failure alerts via Logic Apps or Azure Monitor.

11. Explain the role of SparkSession in PySpark.

SparkSession is the entry point to use Spark functionality. It replaces older contexts like SQLContext and HiveContext. You use it to read/write data, execute SQL queries, and configure settings.

12. How do you optimize storage costs in ADLS?

Use lifecycle management rules to move older data to cooler storage tiers. You can also compress files (e.g., parquet, snappy), partition intelligently, and avoid small files by using batching or merge strategies.

13. Write a SQL query to find duplicate records in a table.

```sql
SELECT name, COUNT(*)
FROM Employees
GROUP BY name
HAVING COUNT(*) > 1;
```

14. What is the purpose of caching in PySpark, and how is it implemented?

Caching helps speed up repeated access to data. You can use `.cache()` or `.persist()` to keep data in memory or on disk.
```python
df.cache()
```

```
df.count()  # Materializes the cache
```

15. Describe the process of integrating ADF with Databricks for ETL workflows.

In ADF, use the 'Azure Databricks' activity to run notebooks or jobs. Pass parameters if needed. You can also link to a Databricks cluster and orchestrate complex workflows combining multiple activities (e.g., copying, transforming, loading).

16. Write Python code to count the frequency of words in a string.

```python
from collections import Counter

text = "hello world hello"
word_freq = Counter(text.split())
print(word_freq)
```

17. How do you handle schema evolution in Delta Lake?

Delta Lake supports schema evolution using the `mergeSchema` option during write operations. This lets you add new columns without rewriting the full dataset.

```python
df.write.option("mergeSchema", "true").format("delta").mode("append").save(path)
```

18. Explain the difference between streaming and batch processing in Spark.

Batch processing handles fixed-size data at intervals, while streaming ingests data in real-time. Spark Structured Streaming provides a micro-batch model, making stream processing feel like continuous batch execution.

19. How do you secure data pipelines in Azure?

Use Managed Identities, RBAC, firewall rules, and private endpoints. Secure data in transit (HTTPS) and at rest (encryption). Also, monitor access via Azure Monitor and audit logs.

20. What are the best practices for managing large datasets in Databricks?

Partition data wisely, avoid small files, cache interim results, use Delta format, prune columns/rows, and leverage Z-ordering and indexing where possible. Monitor with Spark UI and optimize jobs.

Deloitte interview questions & Answer

---

1. What is Z-ordering in Spark?

Answer:
Z-ordering is a technique used in Delta Lake (on Databricks) to **optimize the layout of data** on disk. It helps improve the performance of queries, especially when you're filtering on multiple columns. Imagine you're organizing your bookshelf so that you can find a book faster — that's what Z-ordering does for your data.

It works by co-locating related information close together, so Spark doesn't have to scan the whole dataset. You typically use it during OPTIMIZE with ZORDER BY like this:

OPTIMIZE my_table ZORDER BY (customer_id, order_date)

---

2. Explain the difference between Spark SQL and PySpark DataFrame APIs.

Answer:
Both are used to work with structured data in Spark, but they offer different interfaces:

- **Spark SQL:** Lets you write SQL queries as strings. It's helpful if you're familiar with SQL and want to run traditional queries.
- spark.sql("SELECT * FROM orders WHERE total > 1000")
- **PySpark DataFrame API:** Uses Python methods to manipulate data, which is more programmatic and integrates better with Python code.
- orders.filter(orders.total > 1000).show()

Under the hood, they both use the same execution engine — so performance-wise, they're similar.

---

3. How to implement incremental load in ADF?

Answer:
Incremental load means only loading **new or changed data** instead of reloading everything.

You can implement it in ADF using:

- **Watermark columns** like LastModifiedDate
- Use the **Lookup** or **Stored Procedure** activity to get the last load time.
- In your source dataset, use a **filter** with dynamic content:
- SELECT * FROM source_table WHERE LastModifiedDate > @pipeline().parameters.lastLoadTime

- Then update the watermark value after the load is done using a Stored Procedure or sink logic.

---

4. How do you handle large-scale data ingestion into ADLS?

Answer:
For massive data loads into ADLS, best practices include:

- Use **parallelism** in ADF — set up partitioning or multiple concurrent source queries.
- Use **compression formats** like Parquet or Avro for better performance and cost.
- Ingest data using **ADF Copy Activity**, **Dataflows**, or even **Azure Databricks** for heavy lifting.
- For real-time ingestion, use **Azure Event Hub or IoT Hub** with **Stream Analytics**.

---

5. Write Python code to split a name column into firstname and lastname.

Answer:
Here's a simple snippet using Python and Pandas:

```
import pandas as pd


df = pd.DataFrame({'name': ['John Smith', 'Alice Johnson']})

df[['first_name', 'last_name']] = df['name'].str.split(' ', 1, expand=True)

print(df)
```

Output:

```
          name first_name last_name
0    John Smith      John     Smith
1  Alice Johnson     Alice   Johnson
```

Perfect! Let's continue with Deloitte questions **6 to 10**, explained clearly and in a conversational tone.

---

6. What are fact and dimension tables in data modeling?

Answer:
Think of **fact** and **dimension** tables as the heart of a data warehouse design:

- **Fact Table:**
  Stores **measurable data** — like sales, revenue, quantity, etc. It usually has **foreign keys** pointing to dimension tables. Example: Sales_Fact with fields like product_id, store_id,

sales_amount.

- **Dimension Table:**
  Stores **descriptive data** — like product name, store location, customer details. They help give context to the facts. Example: Product_Dim with product_id, product_name, category.

 Think of it like this: Fact = "What happened?" and Dimension = "Who/What/Where?"

---

7. How do you design and implement data pipelines using Azure Data Factory?

Answer:
Designing a data pipeline in ADF involves a few key steps:

1. **Source Dataset** – Define where your data is coming from (e.g., SQL, Blob, API).
2. **Activities** – Use Copy activity, Data Flow, Stored Proc, etc.
3. **Transformations** – Apply mapping, filtering, joins in **Mapping Data Flows** if needed.
4. **Sink Dataset** – Where your data lands (Azure SQL, ADLS, Synapse, etc.).
5. **Triggers** – To run the pipeline on schedule or event-based.
6. **Monitoring** – Track pipeline runs using the Monitor tab in ADF.

🔧 Bonus: Use **parameters**, **variables**, and **metadata-driven pipelines** to make it dynamic and reusable!

---

8. Explain the concept of PolyBase in Azure SQL Data Warehouse.

Answer:
PolyBase lets you run SQL queries on **external data** stored in files like CSV or Parquet on **Azure Blob or ADLS**, as if it were in a table.

📦 Example use case: Query a huge CSV file stored in ADLS directly from Azure Synapse using T-SQL, without importing it first:

SELECT * FROM ExternalTable

This is super useful when:

- Ingesting external data into Synapse
- Running ELT workloads
- Avoiding extra copy operations

---

9. Write a SQL query to calculate the cumulative sum of a column.

Answer:
Here's a SQL example to calculate cumulative (running) total of salaries:

```
SELECT
  employee_id,
  salary,
  SUM(salary) OVER (ORDER BY employee_id) AS cumulative_salary
FROM
  employees;
```

You can also partition it (e.g., by department):

```
SUM(salary) OVER (PARTITION BY department_id ORDER BY employee_id)
```

---

10. How do you manage partitioning in PySpark?

Answer:
Partitioning in PySpark helps Spark **distribute data** and **parallelize processing** efficiently.

How to manage it:

- **Check current partitions**:
- df.rdd.getNumPartitions()
- **Repartitioning** (increases or balances partitions):
- df = df.repartition(8)
- **Coalescing** (reduces partitions, more efficient than repartition):
- df = df.coalesce(4)

📌 Tip: Use repartition() when increasing partitions for parallelism, and coalesce() when writing to storage and reducing file count.

Here we go with **Deloitte questions 11–15**, answered in a crisp, easy-to-understand style:

---

11. Explain the use of Delta Lake for data versioning.

Answer:
Delta Lake brings **ACID transactions** to data lakes. One of its coolest features? **Time travel and versioning**.

🔄 Every time you write data (append, overwrite, merge), Delta logs the change — allowing you to:

- Query old versions like:
- SELECT * FROM table_name VERSION AS OF 5

- Or use a timestamp:
- SELECT * FROM table_name TIMESTAMP AS OF '2024-04-01T00:00:00'

✅ Great for rollback, audits, debugging, and historical analysis.

---

12. How do you monitor and troubleshoot Spark jobs?

Answer:
You can monitor Spark jobs using:

- **Spark UI**
  Accessed via Databricks or Spark History Server. It shows:
    - Stages and tasks
    - Shuffle reads/writes
    - Execution DAGs
- **Cluster Metrics**
  Check CPU, memory usage, and executor health.
- **Logs**
  Application logs can be viewed in the Spark UI or exported to Azure Log Analytics.

👀 Common issues to watch for:

- Skewed joins
- Out-of-memory errors
- Long GC times

Pro tip: Enable **Adaptive Query Execution (AQE)** for dynamic optimizations.

---

13. Write a SQL query to find employees with the highest salary in each department.

Answer:
Using **window functions**:

```
SELECT *

FROM (

  SELECT *,

      RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS rank

  FROM employees

) AS ranked

WHERE rank = 1;
```

This handles **ties** too (e.g., if two employees share the top salary in a department).

---

14. How do you optimize joins in PySpark for large datasets?

Answer:
You've got a few smart options:

1. **Broadcast Join**
   Broadcast the smaller dataset:
2. from pyspark.sql.functions import broadcast
3. df.join(broadcast(small_df), "id")
4. **Partitioning**
   Ensure both DataFrames are partitioned properly on join keys.
5. **Skew Handling**
   Use salting techniques if one key has a lot more rows.
6. **Join Type**
   Choose wisely — avoid cross joins unless necessary.
7. **Caching**
   Cache reused datasets if memory allows.

💡 Rule of thumb: Avoid shuffles where possible!

---

15. Describe the process of setting up CI/CD for Azure Data Factory.

Answer:
CI/CD in ADF typically uses **Azure DevOps** (or GitHub Actions):

1. **Connect ADF to Git repo** (Azure Repos or GitHub).
2. **Develop in Git mode** (not live).
3. **Publish** from collaboration branch (e.g., feature/dev) to 'adf_publish' branch.
4. **CI Pipeline** – Triggers on commit to 'adf_publish', creates ARM template artifacts.
5. **CD Pipeline** – Deploys those templates to QA/Prod using az deployment.

🚀 Bonus: Use parameterized Linked Services and datasets for environment flexibility.

Yes, let's keep the momentum going! Here are **Deloitte interview questions 16–20** answered clearly and conversationally:

---

16. Write Python code to reverse a string.

Answer: You can reverse a string in a single line using Python slicing:

```
text = "Hello Deloitte"
reversed_text = text[::-1]
print(reversed_text)  # Output: etioletoD olleH
```

Or, using a loop (for interviews that ask for it manually):

```
def reverse_string(s):
    result = ""
    for char in s:
        result = char + result
    return result
```

---

17. What are the key features of Databricks notebooks?

Answer:

Databricks notebooks are like your coding cockpit — and they come packed with features:

- 💻 **Multi-language support** (%python, %sql, %scala, %bash)
- 📊 **Rich visualizations**: Line charts, bar charts, maps, etc.
- 🔁 **Job scheduling** and **parameterization**
- 👯 **Collaboration**: Multiple users can edit the same notebook simultaneously
- □ **MLflow integration** for tracking machine learning models
- □ **Widgets**: Create input boxes and dropdowns for dynamic notebook interaction
- 🔒 **Role-based access control (RBAC)** for security

Super helpful for prototyping, debugging, and presenting work!

---

18. How do you handle late-arriving data in ADF?

Answer:

Late-arriving data = data that arrives after the scheduled pipeline run. Here's how you can handle it in **Azure Data Factory**:

- **Watermarking**: Maintain a watermark (e.g., max modified date) to track what's already loaded.

- **Reprocessing window**: Load data for a range like last 7 days to catch any laggards.
- **Trigger re-runs**: Use tumbling window triggers with "retry" and "dependency" settings.
- **Delta loads**: Make sure your sink supports upserts (like Delta Lake or SQL with MERGE).
- **Logging and alerts**: Track and notify when data doesn't arrive as expected.

---

19. Explain the concept of Data Lakehouse.

Answer:

The **Data Lakehouse** combines the flexibility of a **Data Lake** with the reliability of a **Data Warehouse**.

🚀 Key traits:

- Uses **open file formats** (like Parquet, Delta)
- Supports **ACID transactions** (thanks to Delta Lake or Apache Iceberg)
- Enables **BI and ML** workloads on the same data
- Reduces **data duplication** and movement
- Supports **schema enforcement and governance**

In Databricks, Delta Lake powers the Lakehouse model — enabling fast, reliable analytics directly on raw data.

---

20. How do you implement disaster recovery for ADLS?

Answer:

Disaster recovery for **Azure Data Lake Storage (Gen2)** focuses on resilience and data availability:

1. **Geo-Redundant Storage (GRS)**
   Stores copies of your data in a secondary region automatically.
2. **Snapshots**
   Capture point-in-time versions of files/folders for recovery.
3. **Versioning (for hierarchical namespaces)**
   Keeps historical versions of files in case of accidental deletion or overwrite.
4. **Soft delete**
   Enables you to recover deleted blobs within a retention period.
5. **Replication strategy**
   For critical workloads, use **Azure Data Factory** to mirror data across regions.
6. **Automated backup** via **Azure Backup or third-party tools** like Veeam or Rubrik.

✔ And of course, **test your recovery plan regularly!**

Sure! Here are the answers for the **Fractal Analytics** interview questions (1–20), written in a simple and conversational format for easy understanding:

---

1. How do autoscaling clusters work in Databricks?

Autoscaling in Databricks automatically adjusts the number of worker nodes in a cluster based on workload. When the load increases (e.g., more tasks or larger data), it adds nodes; when the load decreases, it removes nodes. This helps save costs and ensures performance.

---

2. What are the types of Integration Runtimes (IR) in ADF?

There are three types:

- **Azure IR** – Used for data movement between cloud sources.
- **Self-hosted IR** – Installed on-premises to connect on-prem data sources.
- **Azure-SSIS IR** – For running SSIS packages in Azure.

---

3. Difference between Blob Storage and Azure Data Lake Storage (ADLS).

- **Blob Storage** is general-purpose storage for unstructured data.
- **ADLS Gen2** is built on Blob but optimized for analytics. It supports hierarchical namespace, ACLs, and better performance for big data processing.

---

4. How do you integrate Databricks with Azure DevOps for CI/CD pipelines?

- Use the **Databricks Repos** feature to sync code with Azure Repos or GitHub.
- Use Azure DevOps pipelines to automate deployment using **Databricks CLI** or REST APIs.
- Scripts can include notebook deployment, cluster setup, and job scheduling.

---

5. Write a SQL query to convert row-level data to column-level using pivot.

```
SELECT department,
    MAX(CASE WHEN gender = 'Male' THEN salary END) AS Male_Salary,
    MAX(CASE WHEN gender = 'Female' THEN salary END) AS Female_Salary
FROM employee
```

GROUP BY department;

---

6. How do you ensure data quality and validation in ADLS?

- Use **ADF Data Flows** or **Databricks** for validations.
- Implement checks like null values, range validation, data types.
- Create logs or alerts if data fails rules.
- Store validation results separately.

---

7. Explain the use of hierarchical namespaces in ADLS.

Hierarchical namespaces allow directories and subdirectories, like a traditional file system. This makes operations like move, delete, and rename more efficient and enables file-level ACLs.

---

8. Describe the process of setting up and managing an Azure Synapse Analytics workspace.

- Create Synapse workspace and link it to ADLS.
- Create pools (SQL and Spark).
- Ingest data using pipelines or linked services.
- Use Studio to run queries, manage notebooks, monitor jobs, and secure access via RBAC.

---

9. Write PySpark code to calculate the average salary by department.

df.groupBy("department").agg(avg("salary").alias("avg_salary")).show()

---

10. How do you implement streaming pipelines in Databricks?

- Use readStream to read from sources like Kafka, Event Hub, etc.
- Apply transformations.
- Write to sinks using writeStream with checkpointing enabled.
  Example:

df = spark.readStream.format("delta").load("input_path")

df.writeStream.format("delta").option("checkpointLocation", "chkpt_path").start("output_path")

---

11. Explain the purpose of Delta Lake checkpoints.

Checkpoints store the current state of the Delta table to speed up the recovery process. They are created every 10 commits and help avoid reading all log files when querying a table.

---

12. How do you handle data encryption in ADLS?

- Data is encrypted at rest using Microsoft-managed or customer-managed keys (CMK).
- For data in transit, HTTPS is enforced.
- You can use Azure Key Vault to manage CMKs securely.

---

13. Write a SQL query to find the top 3 customers by sales.

```
SELECT customer_id, SUM(sales) as total_sales
FROM orders
GROUP BY customer_id
ORDER BY total_sales DESC
LIMIT 3;
```

---

14. How do you optimize Spark jobs for better performance?

- Use **cache/persist** when reusing data.
- Use **broadcast joins** for small tables.
- Tune **partitioning** and **shuffle operations**.
- Enable **Adaptive Query Execution**.
- Avoid wide transformations when possible.

---

15. Describe the role of triggers in ADF pipelines.

Triggers control **when** a pipeline runs. Types include:

- **Schedule Trigger** – Based on time.
- **Tumbling Window** – For periodic runs with dependency tracking.
- **Event Trigger** – Runs when a blob is created/modified in storage.

---

16. Write Python code to find the largest number in a list.

```
numbers = [4, 9, 1, 23, 6]
```

```
print(max(numbers))
```

---

17. How do you implement parallel processing in PySpark?

Spark automatically parallelizes tasks across nodes.
To implement it manually, you can:

- Use repartition() or coalesce() to control number of partitions.
- Use transformations like mapPartitions() for parallel execution.

---

18. Explain the concept of lineage in data pipelines.

Lineage tracks **where data came from**, **how it changed**, and **where it goes**. It helps with debugging, auditing, and compliance. Tools like Purview or ADF monitoring show lineage visually.

---

19. How do you manage access control in Azure Data Lake?

- Use **Access Control Lists (ACLs)** at file/folder level.
- Use **RBAC** to control access at resource level.
- Integrate with **Azure AD** for authentication.
- Combine both for fine-grained control.

---

20. What are the challenges in integrating on-premises data with Azure services?

- **Latency** and **bandwidth** issues.
- **Firewall and VPN** configurations.
- **Authentication** between on-prem and cloud.
- Keeping **data in sync** during migration.
- Choosing the right **Integration Runtime** in ADF.

Infosys interview Questions Answers

---

1. What is the difference between a job cluster and an interactive cluster in Databricks?

- **Job Cluster**: Created temporarily for the duration of a job and terminated afterward. It is ideal for production workloads where resources are not needed all the time.

- **Interactive Cluster**: Stays alive until manually terminated. Used for development and exploration where you need repeated access for testing and debugging.

---

2. How to copy all tables from one source to the target using metadata-driven pipelines in ADF?

Use a **metadata-driven approach**:

- Store metadata (source and destination table names, schema, etc.) in a control table or config file.
- Loop through the metadata using a **ForEach** activity in ADF.
- Use **Lookup** to fetch metadata, then a **Copy Activity** to perform the actual data movement using parameters from the metadata.
- Dynamic datasets and parameterized linked services enable flexibility for source and sink.

---

3. How do you implement data encryption in Azure SQL Database?

- **At rest**: Azure SQL provides **Transparent Data Encryption (TDE)** by default to protect data stored on disk.
- **In transit**: Uses **TLS (Transport Layer Security)** for securing data being transferred over the network.
- You can manage keys using **Azure Key Vault** to rotate and manage encryption keys securely.

---

4. Write Python code to generate Fibonacci numbers.

```
def generate_fibonacci(n):
    fib = [0, 1]
    for i in range(2, n):
        fib.append(fib[i-1] + fib[i-2])
    return fib[:n]


print(generate_fibonacci(10))  # Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

---

5. What are the best practices for managing and optimizing storage costs in ADLS?

- Use **lifecycle policies** to automatically move infrequently used data to cool/archive tiers.
- **Partition data** efficiently to reduce scanning costs.
- Store **compressed formats** like Parquet or Avro.

- Delete obsolete or temporary files regularly.
- Monitor and audit usage to detect cost anomalies early.

---

6. How do you implement security measures for data in transit and at rest in Azure?

At Rest:

- Use **Azure Storage encryption** (enabled by default) with Microsoft or customer-managed keys.
- For SQL databases, use **Transparent Data Encryption (TDE)**.
- Store secrets (like connection strings, passwords) in **Azure Key Vault**.

In Transit:

- Always enable **HTTPS** for secure communication.
- Use **Private Endpoints** or **VPN/ExpressRoute** for secure access to Azure services.
- For services like ADF, enable **managed identity** and secure connections between services using role-based access control (RBAC).

---

7. Describe the role of triggers and schedules in Azure Data Factory.

- **Triggers** are used to **start pipelines** based on events or schedules.
    - **Schedule Trigger**: Executes pipelines on a time-based schedule (e.g., every night at 1 AM).
    - **Event Trigger**: Starts pipeline when a file arrives in blob storage.
    - **Manual Trigger**: Triggered on-demand or from a REST API.
- Triggers help **automate data pipelines**, reducing the need for manual intervention.

---

8. How do you optimize data storage and retrieval in Azure Data Lake Storage?

- Store data in **columnar formats** like **Parquet** or **Delta** for efficient querying.
- **Partition data** by frequently queried columns (e.g., date, region) to speed up read operations.
- Use **Z-ordering** (in Delta Lake) for better data skipping.
- Enable **hierarchical namespace** to organize data logically.
- Compress data to reduce storage size and I/O costs.

---

9. Write a SQL query to find employees with no manager assigned.

Assuming the manager_id column refers to the employee's manager:

```
SELECT *
FROM employees
WHERE manager_id IS NULL;
```

This query pulls all employees who don't have a manager—possibly the top-level execs!

---

10. How do you implement data deduplication in PySpark?

You can use the dropDuplicates() method:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Deduplication").getOrCreate()

df = spark.read.csv("path/to/data.csv", header=True, inferSchema=True)

# Drop duplicate rows based on all columns
dedup_df = df.dropDuplicates()

# Or drop duplicates based on specific columns
dedup_df = df.dropDuplicates(["employee_id"])

dedup_df.show()
```

This helps you eliminate repeated records efficiently before further transformations.

---

11. Explain the concept of Delta Lake compaction.

Delta Lake compaction is the process of **optimizing the number and size of small files** in your Delta

table.

- In streaming or frequent batch writes, Delta tables can accumulate **lots of small files**, hurting performance.
- Compaction helps by **combining many small files into larger ones**, improving:
    - **Read performance**
    - **Query planning efficiency**
    - **Data skipping**

How to perform compaction:

```
from delta.tables import DeltaTable


deltaTable = DeltaTable.forPath(spark, "/mnt/delta/sales")

deltaTable.optimize().executeCompaction()
```

Or using SQL in Databricks:

```
OPTIMIZE delta.`/mnt/delta/sales`
```

---

12. How do you monitor ADF pipeline performance?

You can monitor performance using:

- **Monitor tab in ADF Studio**: Check pipeline, activity run status, execution time, and errors.
- **Azure Monitor + Log Analytics**: Enable diagnostic logging to push logs to a Log Analytics workspace.
- **Alerts**: Set up alerts based on failure, duration, or custom metrics.
- **Activity run metrics**: Track time taken for data movement, transformation, or lookups.

This helps you identify bottlenecks, long-running activities, and troubleshoot issues faster.

---

13. Write a SQL query to find the second-highest salary in a table.

```
SELECT MAX(salary) AS SecondHighestSalary

FROM employees

WHERE salary < (

   SELECT MAX(salary) FROM employees

);
```

This works by getting the **max salary** that is **less than the highest one**.

Alternate (using DENSE_RANK):

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

---

14. How do you implement incremental load in Databricks?

Incremental load means **only processing new or updated records**.

Common approaches:

1. **Using Watermark (timestamp column):**
2. df = spark.read.format("delta").load("/mnt/delta/source")
3. new_data = df.filter("last_updated > '2024-01-01 00:00:00'")
4. **Using merge for upserts:**
5. from delta.tables import DeltaTable
6.
7. delta_table = DeltaTable.forPath(spark, "/mnt/delta/target")
8.
9. delta_table.alias("tgt").merge(
10. new_data.alias("src"),
11. "tgt.id = src.id"
12.).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()

This keeps your target table in sync without reprocessing all records.

---

15. Describe the role of Azure Key Vault in securing sensitive data.

Azure Key Vault is a **centralized cloud service to manage secrets**, keys, and certificates.

In data engineering, it's used to:

- **Store connection strings, passwords, API keys**, etc.
- Integrate with **ADF, Databricks, Synapse** securely (via linked services).
- Support **access control** via Azure RBAC and **Managed Identity**.
- Enable **audit logs** for secret access.

Using Key Vault removes hardcoded secrets and improves overall security and compliance.

---

16. Write Python code to sort a list of dictionaries by a key.

Let's say you have a list of dictionaries with employee info, and you want to sort them by salary:

```
employees = [
  {'name': 'Alice', 'salary': 70000},
  {'name': 'Bob', 'salary': 50000},
  {'name': 'Charlie', 'salary': 60000}
]

# Sort by salary in ascending order
sorted_employees = sorted(employees, key=lambda x: x['salary'])

for emp in sorted_employees:
  print(emp)
```

To sort in **descending order**, just add reverse=True.

---

17. How do you handle schema evolution in ADF?

Schema evolution in ADF refers to handling changes in source data structure like **added columns**, **changed datatypes**, etc.

Ways to handle it:

1. **Auto Mapping in Copy Data Activity**:
    - Use *"Auto Mapping"* in the mapping tab.
    - It adapts to new columns if the source changes.
2. **Flexible Schema Handling in Data Flows**:
    - Use *"Allow schema drift"* in data flows.

- This lets ADF handle columns not explicitly defined in your transformation.

3. **Parameterization & Metadata-driven Pipelines**:
    - Dynamically read schema info using metadata (from SQL or config files).
    - Loop through columns and build dynamic transformations.
4. **Using select * cautiously**:
    - Helps in quickly adopting new columns but can break downstream if not handled with care.

---

18. Explain the concept of shuffling in Spark.

Shuffling is the process where Spark redistributes data across partitions, typically triggered by operations like:

- groupByKey()
- join()
- repartition()
- distinct()

It's **expensive** because:

- It involves **disk I/O**, **network transfer**, and **serialization**.
- It can lead to **performance bottlenecks** and **OOM errors** if not managed properly.

Optimization Tips:

- Prefer reduceByKey() over groupByKey().
- Use **broadcast joins** when joining a small dataset with a large one.
- Minimize wide transformations.

---

19. How do you manage metadata in Azure Data Lake?

Managing metadata in ADLS ensures **discoverability**, **governance**, and **data lineage**.

Ways to manage metadata:

1. **Azure Purview (Microsoft Purview)**:
    - Automatically scans ADLS, builds **data catalog**, tracks schema, and provides **lineage**.
2. **Directory Naming Conventions**:
    - Use folder structures to store partitioned metadata like /year=2024/month=04/.

3. **Metadata Tables**:
    - Store metadata (file paths, schema info, modified timestamps) in SQL/Delta tables.
4. **Schema registry** (e.g., Azure Schema Registry or Confluent if Kafka used):
    - Maintains schema versions for streaming data.
5. **Tagging and Classification**:
    - Classify sensitive data (e.g., PII, financial) and add custom tags.

---

20. What are the key considerations for designing scalable pipelines in ADF?

To build scalable and efficient pipelines in Azure Data Factory:

✅ **Parameterization**

- Use parameters in datasets and linked services to reuse pipelines.

✅ **Use of Mapping Data Flows**

- Design transformations within ADF's scalable Spark-based data flow engine.

✅ **Optimize parallelism**

- Enable *"Degree of copy parallelism"*.
- Use *ForEach activities with batch count*.

✅ **Fault Tolerance**

- Add **Retry policies**, **Timeouts**, and **Failure paths** (via If Condition or Until).

✅ **Scalable Linked Services**

- Use **Azure Integration Runtime** for cloud-to-cloud, and **Self-hosted IR** for on-prem connectivity.

✅ **Incremental Loads**

- Load only the delta (new/changed records) to reduce load and speed up pipelines.

✅ **Monitoring and Alerting**

- Integrate with Azure Monitor for performance and failure alerts.

EY Interview Questions & Answers

1. How to handle null values in PySpark (drop/fill)?

In PySpark, you can handle nulls using dropna() and fillna() methods:

```
# Drop rows with null values

df_cleaned = df.dropna()


# Fill null values

df_filled = df.fillna({'column1': 'default', 'column2': 0})
```

You can specify subsets of columns and threshold of non-null values as needed.

---

2. What is AQE (Adaptive Query Execution) in Databricks?

AQE is a Spark optimization that adjusts query execution plans at runtime. It helps with:

- Dynamically switching join strategies (e.g., shuffle to broadcast)
- Optimizing skewed joins
- Coalescing shuffle partitions

AQE improves performance for unpredictable data patterns.

---

3. How do you handle error handling in ADF using retry, try-catch blocks, and failover mechanisms?

- **Retry**: Configure retry policies in activity settings (number of retries and intervals).
- **Try-Catch**: Use **If Condition**, **Switch**, or **Until** activities with custom logic.
- **Failover**: Use global parameters or alternative execution paths to redirect processing when failure occurs.

Logging and alerts via Log Analytics or Azure Monitor are also key.

---

4. How to track file names in the output table while performing copy operations in ADF?

Use the @dataset().path or @item().name expressions in a Copy Data activity's sink mapping. You can also use the **Get Metadata** activity to fetch file names beforehand and pass them through a pipeline variable into the sink (e.g., SQL column).

---

5. Write a SQL query to display the cumulative sum of a column.

```
SELECT
```

```
    employee_id,
    department,
    salary,
    SUM(salary) OVER (PARTITION BY department ORDER BY employee_id) AS cumulative_salary
FROM employees;
```

This gives a running total of salary per department.

---

6. Explain the role of Azure Key Vault in securing sensitive data.

Azure Key Vault:

- Manages secrets (like passwords, keys, connection strings)
- Provides secure access through managed identities
- Ensures encryption at rest and in transit
- Integrates with ADF, Databricks, and other Azure services to eliminate hardcoding secrets in code

---

7. How do you manage and automate ETL workflows using Databricks Workflows?

- Use **Databricks Workflows** to define job clusters, tasks, and dependencies.
- Chain tasks using notebooks, Python scripts, or SQL commands.
- Schedule jobs using cron expressions or triggers.
- Monitor runs with built-in logging and alerts.
- Integrate with CI/CD pipelines via REST APIs or GitHub actions.

---

8. Describe the process of setting up disaster recovery for ADLS.

- **Geo-redundant storage (GRS)** ensures data is replicated across regions.
- **Soft delete** and **versioning** help recover from accidental deletes.
- Implement automated **backups** via Data Factory or third-party tools.
- Monitor with **Azure Monitor** and ensure access control via RBAC/ACLs.

---

9. Explain the difference between narrow and wide transformations in PySpark.

- **Narrow**: Each partition depends on a single partition (e.g., map, filter)

- **Wide**: Involves shuffling data between partitions (e.g., groupByKey, join, reduceByKey)
- Wide transformations are more expensive and often trigger shuffles.

---

10. How do you optimize PySpark jobs for large datasets?

- Use **cache/persist** smartly
- Avoid **shuffles** when possible
- Use **broadcast joins** for small lookup tables
- Tune **partition size** and **memory** usage
- Leverage **DataFrame API** over RDD
- Enable **AQE** for dynamic optimizations

---

11. Write a SQL query to find the average salary for each department.

SELECT department_id, AVG(salary) AS avg_salary

FROM employees

GROUP BY department_id;

This will return the average salary grouped by each department.

---

12. What is the role of Delta Lake in modern data architectures?

Delta Lake brings **ACID transactions**, **schema enforcement**, **time travel**, and **unified batch & streaming** processing to data lakes. It bridges the gap between data lakes and data warehouses, forming the foundation for the **Lakehouse** architecture.

Key benefits:

- Reliable data pipelines
- Easier data governance
- Simplified ETL and analytics
- Scalable with open format (Parquet-based)

---

13. How do you monitor and debug ADF pipelines?

You can monitor and debug using:

- **Monitor tab**: Check activity run status, duration, errors

- **Activity output logs**: Inspect input/output/error messages
- **Azure Monitor and Log Analytics**: Track performance and trigger alerts
- **Integration Runtime metrics**: View resource utilization
- Enable **diagnostic settings** to log to storage, Event Hubs, or Log Analytics for centralized monitoring

---

14. Write PySpark code to perform an inner join between two DataFrames.

```
df_joined = df1.join(df2, df1.id == df2.id, "inner")

df_joined.show()
```

This joins df1 and df2 where the id columns match, keeping only matched rows.

---

15. How do you manage schema drift in ADF?

Schema drift refers to changes in the source schema (like new columns). To handle it in ADF:

- In **Copy Data** activity, enable **"Auto Mapping"** to allow dynamic schema mapping.
- Use **dataset parameters** for dynamic datasets.
- Set **"Allow schema drift"** in the source/sink settings.
- Use **data flows** to dynamically map columns and use expressions like byName().

---

16. Describe the concept of fault tolerance in Spark.

Spark achieves fault tolerance by:

- Using **RDD lineage** to recompute lost partitions
- Storing intermediate data in memory/disk
- Writing checkpoints for long lineage chains
- Using **speculative execution** to run slow tasks on alternate nodes
- Ensuring executors can recover from node failures

---

17. Write Python code to calculate the factorial of a number.

```
def factorial(n):
   if n == 0 or n == 1:
      return 1
```

```
    return n * factorial(n - 1)


print(factorial(5))  # Output: 120
```

Or iteratively:

```
def factorial(n):
    result = 1
    for i in range(2, n+1):
        result *= i
    return result
```

---

18. How do you handle incremental data loads in ADLS?

Approaches:

- Use **watermarking** or **lastModifiedDate** filters in source queries
- Implement **delta loads** with Copy Activity and a metadata table
- Track changes using **Change Data Capture (CDC)** or **file naming conventions**
- Maintain **audit columns** (e.g., updated_at) for identifying new/changed records

---

19. What are the security features in Azure Synapse Analytics?

Key features:

- **Managed Identity** for secure resource access
- **Azure Active Directory (AAD)** integration
- **Role-Based Access Control (RBAC)**
- **Network isolation** with Private Endpoints and Firewalls
- **Data encryption** (in transit and at rest)
- Integration with **Azure Key Vault**
- **Auditing** and **threat detection** via Defender for Cloud

---

20. Explain the concept of partitioning in PySpark.

Partitioning in PySpark refers to how data is distributed across worker nodes.

Why it's important:

- Impacts parallelism and performance
- Helps reduce shuffles

Use:

- repartition(n) → reshuffles data for equal-sized partitions
- coalesce(n) → reduces number of partitions (faster, no shuffle)
- Partitioning columns are critical for operations like joins or aggregations

---

21. How do you implement real-time data processing in Databricks?

You can implement real-time processing using **Structured Streaming** in Databricks:

- Read streaming data from sources like **Kafka**, **Event Hubs**, or **Azure IoT Hub**
- Use **structured streaming APIs** in PySpark
- Write results to sinks like **Delta Lake**, **Synapse**, or **ADLS**
- Use **trigger options** for micro-batching or continuous processing

Example:

```
df = spark.readStream.format("kafka") \
  .option("kafka.bootstrap.servers", "<broker>") \
  .option("subscribe", "topic") \
  .load()

processed = df.selectExpr("CAST(value AS STRING)")

query = processed.writeStream \
  .format("delta") \
  .option("checkpointLocation", "/checkpoints/") \
  .start("/output/path/")
```

---

22. Write a SQL query to find duplicate records in a table.

```
SELECT column1, column2, COUNT(*)
FROM table_name
GROUP BY column1, column2
```

HAVING COUNT(*) > 1;

This finds duplicates based on combinations of columns.

---

23. How do you integrate Azure Key Vault with ADF pipelines?

To use Key Vault secrets in ADF:

1. **Create a Linked Service** for Key Vault.
2. **Access secrets** in linked services or parameters using @Microsoft.KeyVault(...).
3. Set **Managed Identity** for ADF and give it Key Vault access with **"Get" secret permissions**.
4. Example in JSON or expression:
5. "@Microsoft.KeyVault(SecretName='sqlPassword')"

---

24. What are the best practices for optimizing storage costs in ADLS?

- **Lifecycle policies** to auto-delete or move old/unused data
- Use **Hierarchical Namespace (HNS)** for directory-level management
- Choose appropriate **storage tiers** (Hot, Cool, Archive)
- **Compress files** (e.g., Parquet, Avro) to reduce size
- Avoid storing small files; consolidate using **compaction**
- Monitor with **Azure Cost Management**

---

25. How do you implement CI/CD for Azure Synapse Analytics?

Steps:

1. **Source control**: Integrate Synapse workspace with Git (Azure DevOps or GitHub).
2. **Develop in Git mode**: Code notebooks, pipelines, SQL scripts, etc.
3. Use **ARM templates** or **Synapse deployment scripts** for infrastructure as code.
4. Set up a **CI/CD pipeline** using Azure DevOps:
    - Validate templates
    - Use tasks like "Azure Resource Group Deployment"
    - Deploy notebooks and artifacts using Synapse REST API or Synapse CLI

---

26. Explain the role of Integration Runtime in ADF.

Integration Runtime (IR) is the compute infrastructure used by ADF for:

- **Data movement** (copy data between sources)
- **Activity dispatching** (e.g., data flow, stored proc)
- **SSIS execution** (for lifted SSIS packages)

Types:

- **Azure IR**: For cloud-native data movement
- **Self-hosted IR**: For on-prem or VNet-restricted sources
- **Azure-SSIS IR**: For running SSIS packages in ADF

---

27. How do you secure sensitive data in Azure?

- Use **Azure Key Vault** for managing secrets, keys, and certificates
- Apply **encryption at rest** (default) and **in transit** (TLS/HTTPS)
- Leverage **RBAC** and **AAD integration** for access control
- Configure **Private Endpoints** to isolate traffic
- Use **network security groups**, firewalls, and VNet rules
- Enable **auditing, logging, and threat detection**

---

28. Describe the process of creating a data pipeline for real-time analytics.

Typical steps:

1. **Ingest** data using Kafka/Event Hubs/IoT Hub into Databricks
2. Process it using **Structured Streaming** with low-latency logic
3. Write to a **Delta Lake** table (streaming sink)
4. Query the table with **Power BI** or **Synapse Serverless SQL**
5. Use **monitoring and alerting** for latency and freshness
6. Ensure **checkpointing**, **idempotency**, and **scalability**

Persistent Systems Interview Questions & Answers

---

1. Explain broadcast join in PySpark.

A **broadcast join** is used when one of the DataFrames is small enough to fit in memory. Spark broadcasts the smaller DataFrame to all executors, avoiding a full shuffle operation.

Benefits:

- Reduces data movement
- Faster than shuffle joins
- Ideal when joining a large dataset with a small lookup table

Example:

```
from pyspark.sql.functions import broadcast


result = large_df.join(broadcast(small_df), "id")
```

---

2. How to create a rank column using the Window function in PySpark?

Use Window and rank() or dense_rank() functions to create a rank column.

Example:

```
from pyspark.sql.window import Window

from pyspark.sql.functions import rank


windowSpec = Window.partitionBy("department").orderBy("salary")

df.withColumn("rank", rank().over(windowSpec)).show()
```

---

3. What is the binary copy method in ADF, and when is it used?

Binary copy in ADF transfers files **as-is**, without parsing or transformation. It's useful when:

- File formats are unknown or unsupported
- You want to move images, zip files, videos, etc.
- You need fast point-to-point transfer between file-based systems

You can enable binary copy by setting **"binary copy"** in the Copy activity.

---

4. How do you monitor and optimize performance in Azure Synapse?

Monitoring tools:

- **Monitor Hub** in Synapse Studio
- **DMVs** (Dynamic Management Views) for query insights
- **SQL Activity logs**

Optimization tips:

- Use **result set caching**
- Choose proper **distribution methods** (hash, round-robin)
- Use **materialized views**
- Avoid excessive shuffling
- Partition large tables appropriately

---

5. Write Python code to identify duplicates in a list and count their occurrences.

```
from collections import Counter

data = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple']
counter = Counter(data)

duplicates = {item: count for item, count in counter.items() if count > 1}
print(duplicates)
```

Output:

```
{'apple': 3, 'banana': 2}
```

---

6. What are the key features of Azure DevOps?

- **Version control** with Git
- **Pipelines** for CI/CD
- **Boards** for Agile project tracking
- **Artifacts** for package management
- **Test Plans** for automated/manual testing
- **Integration** with tools like GitHub, Slack, VS Code

---

7. How do you handle schema drift in ADF?

Schema drift = changes in source schema (e.g., added/removed columns).

How to handle:

- Enable **"Allow schema drift"** in mapping data flows
- Use **auto-mapping** or dynamic column handling
- Use **parameterized datasets** to pass schema info
- Combine with **schema projection** for flexibility

---

8. Explain the concept of denormalization and when it should be used.

Denormalization is the process of combining normalized tables into fewer tables to improve read performance.

When to use:

- In **OLAP** systems or data warehouses
- When query speed is more important than storage efficiency
- For simplifying complex joins

Pros: Faster queries, easier reporting
Cons: Data redundancy, maintenance overhead

Cognizant Interview Questions & Answers

---

1. Difference between repartition() and coalesce() in PySpark

- **repartition(n)**: Increases or decreases partitions by **reshuffling** the data. More expensive due to full shuffle.
- **coalesce(n)**: Reduces the number of partitions by **merging** existing ones. More efficient for decreasing partitions since it avoids full shuffle.

Use Cases:

- Use repartition() when increasing partitions or for better data distribution.
- Use coalesce() when decreasing partitions, especially before writing large data.

---

2. How to persist and cache data in PySpark?

- **cache()**: Stores the DataFrame in **memory** only.

- **persist()**: Gives you control over **storage level** (e.g., memory, disk, or both).

Example:

```
df.cache()  # Stores in memory


from pyspark.storagelevel import StorageLevel

df.persist(StorageLevel.MEMORY_AND_DISK)
```

Use unpersist() to free memory when done.

---

3. How do you use Azure Logic Apps to automate data workflows in SQL databases?

Azure Logic Apps let you create **serverless workflows** using connectors for SQL Server, Azure SQL, and more.

Example Workflow:

1. Trigger: Schedule or HTTP request
2. Action 1: Run stored procedure or SQL query
3. Action 2: Send an email or push results to Power BI

Use Logic Apps for:

- Periodic data sync
- Alerts based on SQL conditions
- Event-driven SQL workflows

---

4. Differences between ADLS Gen1 and Gen2

Feature	**ADLS Gen1**	**ADLS Gen2**
Based on	HDFS	Azure Blob Storage
Cost	Higher	Lower (pay-as-you-go blob pricing)
Performance	Moderate	Better performance
Security	ACL support	ACL + RBAC
Integration	Limited	Broad integration with Azure tools
Hierarchical namespace	Yes	Optional (but recommended)

ADLS Gen2 is the recommended and modern option for big data storage in Azure.

---

5. Write a SQL query to find gaps in a sequence of numbers

```
SELECT curr.id + 1 AS missing_id
FROM your_table curr
LEFT JOIN your_table next ON curr.id + 1 = next.id
WHERE next.id IS NULL;
```

This finds missing numbers **between existing sequence values**.

---

6. How do you ensure high availability and disaster recovery for Azure SQL Databases?

- **High Availability:**
  - Use **Premium** or **Business Critical** tier with zone-redundant availability
  - Auto-failover groups for seamless failover between regions
- **Disaster Recovery:**
  - **Geo-replication** (readable secondary in different region)
  - **Backups**: Point-in-time restore, long-term retention

---

7. Explain the role of pipelines in Azure DevOps

- Pipelines are used to **automate CI/CD** processes.
- **CI pipeline**: Builds, tests, and validates code automatically.
- **CD pipeline**: Deploys artifacts to environments like ADF, Databricks, Azure SQL, etc.

Key Components:

- YAML or Classic pipelines
- Build agents
- Stages, jobs, and tasks
- Integration with Git repos and artifacts

---

8. How do you implement data masking in ADF for sensitive data?

Options for data masking in ADF:

- Use **Derived Column** transformation in Mapping Data Flows to mask or replace values:
- deriveColumn("masked_ssn", expr("substring(ssn, 1, 3) + '****'"))

- **Dynamic Data Masking** at the Azure SQL level
- Leverage **Azure Key Vault** and parameterization to avoid exposing secrets
- Apply **row-level or column-level filters** for sensitive datasets

---

TCS Data Engineering Interview Questions & Answers

---

1. How many jobs, stages, and tasks are created during a Spark job execution?

In Spark:

- **Job**: Triggered by an action (e.g., collect, count, save, etc.).
- **Stage**: A job is divided into **stages** based on wide transformations (shuffle boundaries).
- **Task**: A stage is divided into **tasks**, with each task processing a partition of data.

Example: If a Spark job reads data, performs a groupBy, and writes it:

- It could be 1 job,
- Split into 2 stages (before and after the shuffle),
- With N tasks (equal to number of partitions in each stage).

---

2. What are the activities in ADF (e.g., Copy Activity, Notebook Activity)?

Common **ADF Activities** include:

- **Copy Activity**: Transfers data between source and sink.
- **Data Flow Activity**: Transforms data using Mapping Data Flows (visual ETL).
- **Notebook Activity**: Executes Databricks notebooks.
- **Stored Procedure Activity**: Runs stored procedures in SQL DB.
- **Web Activity**: Calls a REST endpoint.
- **Lookup/Get Metadata/Set Variable**: Used for control flow.

Each activity serves a unique role in orchestrating end-to-end pipelines.

---

3. How do you integrate ADLS with Azure Databricks for data processing?

Integration steps:

1. **Mount ADLS to Databricks** using:
    - SAS token
    - Service principal (recommended with Azure Key Vault)

```
configs = {
  "fs.azure.account.auth.type": "OAuth",
  "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
  "fs.azure.account.oauth2.client.id": "<client-id>",
  "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="kv-scope", key="client-secret"),
  "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<tenant-
id>/oauth2/token"
}

dbutils.fs.mount(
  source = "abfss://<container-name>@<storage-account>.dfs.core.windows.net/",
  mount_point = "/mnt/datalake",
  extra_configs = configs)
```

2. Read/write from/to /mnt/datalake/... like any file system.

---

4. Write Python code to check if a string is a palindrome.

```
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]

print(is_palindrome("Madam"))  # True
```

---

5. How do you implement data governance in a data lake environment?

Key components of **Data Governance** in a data lake:

- **Data Cataloging**: Use Azure Purview or Microsoft Purview for metadata management.
- **Access Control**: Apply RBAC + ACLs on ADLS Gen2.

- **Data Classification**: Tag sensitive data for security.
- **Data Lineage**: Track data transformations using Purview or Databricks lineage.
- **Auditing**: Enable logging for all access and changes.
- **Schema Enforcement**: Use Delta Lake to maintain schema.

---

6. Explain the differences between Azure SQL Database and Azure SQL Managed Instance.

Feature	Azure SQL Database	Azure SQL Managed Instance
Managed by Azure	Yes	Yes
SQL Server compatibility	Partial	Full (including SQL Agent, SSIS)
VNET support	Limited	Full VNET integration
Use cases	Cloud-first, modern apps	Lift-and-shift legacy apps

Managed Instance is closer to full SQL Server, great for migration.

---

7. How do you monitor and troubleshoot issues in Azure SQL Database?

- **Monitoring Tools**:
    - **Query Performance Insight**: Shows resource-consuming queries.
    - **SQL Auditing**: Captures DB activities.
    - **Azure Monitor / Log Analytics**: Tracks performance and issues.
    - **DMVs (Dynamic Management Views)**: For real-time diagnostics.
- **Troubleshooting Tips**:
    - Use sys.dm_exec_requests, sys.dm_exec_query_stats
    - Analyze deadlocks, long-running queries
    - Set alerts on CPU/memory utilization

---

8. Describe the process of data ingestion in Azure Synapse.

Ingestion in Synapse includes:

- **Using COPY INTO** from external sources like ADLS, Blob, or external tables.
- **Synapse Pipelines** (ADF-like) to orchestrate batch loads.
- **Linked Services** for connection.

- **Integration with PolyBase** for fast bulk loading.
- **Streaming ingestion** using Event Hubs or Kafka with Spark pools.

Ingested data lands in **dedicated SQL pools**, **serverless SQL**, or **Spark tables**, depending on the architecture.

LatentView Data Engineering Interview Questions & Answers

---

1. Explain the purpose of SparkContext and SparkSession.

- **SparkContext**:
    - The entry point to Spark Core functionality. It sets up internal services and establishes a connection to a Spark execution environment.
    - Used in older Spark versions (<2.0).
    - Example: sc = SparkContext(appName="MyApp")
- **SparkSession**:
    - Introduced in Spark 2.0 as the new entry point for all Spark functionality, including Spark SQL and DataFrame APIs.
    - Internally, it creates a SparkContext.
    - Recommended over SparkContext for all modern use.
    - Example: spark = SparkSession.builder.appName("MyApp").getOrCreate()

---

2. How to handle incremental load in PySpark when the table lacks a last_modified column?

If there's no last_modified or created_at column:

- **Option 1**: Use **checksum/hash comparison**:
    - Generate a hash of relevant columns and compare with the previously stored hash snapshot.
- **Option 2**: Use **change tracking tables** or **CDC (Change Data Capture)** from the source if supported.
- **Option 3**: If the table is small, do full load and deduplicate on the destination using surrogate keys or primary keys.
- **Option 4**: Use **record versioning** if available via audit logs.

---

3. How do you use Azure Stream Analytics for real-time data processing?

- Azure Stream Analytics (ASA) is a real-time event processing engine.
- **Steps**:
    1. Define **input**: IoT Hub, Event Hub, or Blob Storage.
    2. Define **query**: Use SQL-like syntax to process data (aggregations, joins, filters).
    3. Define **output**: Azure SQL, Power BI, ADLS, Blob, etc.
- **Example query**:

```
SELECT deviceId, AVG(temperature) AS avgTemp
INTO outputAlias
FROM inputAlias
GROUP BY deviceId, TumblingWindow(minute, 1)
```

---

4. What are the security features available in ADLS (e.g., access control lists, role-based access)?

Security features in Azure Data Lake Storage Gen2:

- **Role-Based Access Control (RBAC)**:
    - Assigns permissions at subscription/resource/container level via Azure AD roles.
- **Access Control Lists (ACLs)**:
    - Fine-grained permissions at folder and file level.
    - Supports POSIX-style permissioning.
- **Network Security**:
    - Virtual Network (VNet) service endpoints, firewalls, and private endpoints.
- **Encryption**:
    - Data is encrypted at rest with Microsoft-managed or customer-managed keys (CMK).
    - Supports HTTPS for encryption in transit.

---

5. Write a SQL query to remove duplicate rows from a table.

```
DELETE FROM my_table
WHERE id NOT IN (
  SELECT MIN(id)
  FROM my_table
```

```
  GROUP BY column1, column2, column3
);
```

This assumes id is a unique identifier. Adjust columns based on what defines a duplicate.

---

6. How do you manage data lifecycle policies in ADLS?

Use **Azure Blob Lifecycle Management policies**:

- Define **rules** for automatic tiering or deletion based on blob age or last modified date.

Examples:

- Move blobs to **cool tier** after 30 days.
- Delete blobs **older than 180 days**.

Steps:

1. Go to storage account > Data Management > Lifecycle Management.
2. Add rule: Choose conditions and actions.
3. Apply to selected containers or blob prefixes.

---

7. What are the key considerations for designing a scalable data architecture in Azure?

- **Modular design**: Decouple ingestion, processing, and serving layers.
- **Use of scalable services**: Databricks, Synapse, ADF, ADLS Gen2.
- **Partitioning**: Effective partitioning for parallelism.
- **Security & governance**: Use Purview, Key Vault, RBAC/ACLs.
- **Monitoring**: Implement telemetry/logging with Azure Monitor, Log Analytics.
- **Automation**: Use CI/CD and infrastructure as code (ARM/Bicep/Terraform).
- **Cost optimization**: Use proper storage tiers, autoscaling, and data lifecycle rules.

---

8. How do you integrate Azure Key Vault with other Azure services?

- **Azure Key Vault** is used to store secrets, keys, and certificates securely.

Integration Examples:

- **ADF**: Create a linked service and reference secrets using @Microsoft.KeyVault() syntax.
- **Databricks**: Use secret scopes backed by Key Vault (dbutils.secrets.get()).
- **App Services / Functions**: Reference secrets in environment variables using

@Microsoft.KeyVault in configuration.

- **Synapse**: Secure credentials for Linked Services.

Steps:

1. Assign **Managed Identity** to the service.
2. Grant **access policy** in Key Vault for the identity.
3. Reference the secret in service configuration.

EXL Data Engineering Interview Questions & Answers

---

1. Difference between TRUNCATE and DELETE in SQL

Feature	DELETE	TRUNCATE
Operation Type	DML (Data Manipulation Language)	DDL (Data Definition Language)
WHERE Clause	Can use WHERE to delete specific rows	Cannot filter rows, deletes all
Logging	Fully logged	Minimal logging
Rollback	Can be rolled back	Can be rolled back (in most RDBMS)
Triggers	Activates triggers	Does not activate triggers
Identity Reset	Doesn't reset identity column	Resets identity column (in some DBs)
Speed	Slower for large data	Faster for large data

---

2. How do you handle big data processing using Azure HDInsight?

Azure HDInsight is a cloud-based service for open-source analytics frameworks like Hadoop, Spark, Hive, etc.

Steps for handling big data:

- **Cluster Selection**: Choose Spark/Hadoop cluster type depending on workload.
- **Data Storage**: Use ADLS or Blob Storage for input/output.
- **Data Ingestion**: Use tools like Apache Sqoop, Kafka, or ADF.
- **Processing**:
    - Use Spark jobs for in-memory distributed processing.
    - Use Hive or Pig for SQL-based ETL.

- **Optimization**: Use partitioning, caching, compression, and YARN tuning.
- **Monitoring**: Use Ambari for cluster monitoring, logs, and performance tuning.

---

3. How to implement parallel copies in ADF using partitioning?

You can implement parallel copy using **Source partitioning** in Copy Activity.

Steps:

1. In ADF Copy Activity:
    - Go to **Source** tab → Enable **"Enable partitioning"**.
2. Set partition option:
    - **Dynamic Range**: Provide column and range values (e.g., Date, ID).
    - **Static Range**: Predefine ranges.
3. Set **degree of parallelism** (default is 4).

This breaks the data into slices and copies in parallel, improving performance.

---

4. Write Python code to replace vowels in a string with spaces.

```
def replace_vowels_with_space(s):
   vowels = 'aeiouAEIOU'
   return ''.join(' ' if char in vowels else char for char in s)


# Example
print(replace_vowels_with_space("Data Engineer"))
# Output: "D t  Eng n  r"
```

---

5. How do you implement data encryption at rest and in transit in ADLS?

Encryption at Rest:

- Enabled by default in ADLS using Azure Storage Service Encryption (SSE).
- You can use:
    - **Microsoft-managed keys** (default).
    - **Customer-managed keys (CMK)** stored in Azure Key Vault.

Encryption in Transit:

- Enforced using **HTTPS**.
- Private Endpoints and VPNs help avoid public internet exposure.

Advanced security: Enable **Secure Transfer Required**, use **firewalls** and **VNet integration**.

---

6. Describe the use of Azure Synapse Analytics and how it integrates with other Azure services.

Azure Synapse is an integrated analytics platform combining big data and data warehousing.

Key Uses:

- Data ingestion, preparation, management, and visualization.
- Run T-SQL queries on both structured and unstructured data.
- Real-time analytics with Spark & SQL engines.

Integration:

- **Azure Data Lake**: Store raw and curated data.
- **ADF**: Pipelines for data orchestration.
- **Power BI**: For dashboarding directly from Synapse.
- **Azure ML**: For machine learning model training.
- **Azure DevOps**: For CI/CD automation.

---

7. How do you implement continuous integration and continuous deployment (CI/CD) in Azure DevOps?

Steps:

1. **Source Code Management**:
    - Store ADF/Synapse code in Azure Repos or GitHub.
2. **CI Pipeline**:
    - Trigger on code push.
    - Run validation tests or linting.
    - Package ARM templates / JSONs for deployment.
3. **CD Pipeline**:
    - Deploy artifacts to target environment (dev/test/prod).
    - Use az CLI, PowerShell, or built-in deployment tasks.
4. **Approvals**:

- Add stage approvals for production.
- Use environment-specific variables.

---

8. Explain the role of metadata in data modeling and data architecture.

Metadata is data about data.

In Data Modeling:

- Describes data structure (columns, data types, constraints).
- Defines relationships between tables (PK, FK).
- Helps tools understand schema and validation rules.

In Data Architecture:

- Tracks **data lineage** and **provenance**.
- Facilitates **governance**, **cataloging**, and **compliance**.
- Used in tools like **Azure Purview**, **Databricks Unity Catalog**.

It acts as a **blueprint** that improves discoverability, quality, and trust in data systems.

KPMG Data Engineering Interview Questions & Answers

---

1. How to create and deploy notebooks in Databricks?

Creating a notebook:

1. Log in to Azure Databricks.
2. Click on **Workspace > Users > your email**.
3. Click **Create > Notebook**.
4. Name the notebook, choose a default language (Python, SQL, Scala, etc.), and attach a cluster.

Deploying notebooks:

- **Manual execution**: Run the notebook interactively.
- **Scheduled job**: Convert the notebook into a job and set a schedule.
- **CI/CD deployment**: Use Git (e.g., Azure DevOps) to store notebooks and deploy using Databricks CLI or REST API in pipelines.

---

2. What are the best practices for data archiving and retention in Azure?

- **Use Azure Data Lake/Blob** for long-term storage.
- **Apply Lifecycle Management Policies**:
    - Move old data to cool/archive tiers based on age.
    - Automatically delete data after retention period ends.
- **Tag data** with metadata for classification (e.g., creation date).
- **Encrypt and secure** archived data (use CMK if needed).
- **Monitor access** and compliance using Azure Monitor/Azure Purview.
- **Document** and automate retention policies in your data governance strategy.

---

3. How do you connect ADLS (Azure Data Lake Storage) to Databricks?

You can connect using **OAuth (Service Principal)** or **Access Key**:

Mounting ADLS Gen2 to Databricks:

```
configs = {

  "fs.azure.account.auth.type": "OAuth",

  "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

  "fs.azure.account.oauth2.client.id": "<client-id>",

  "fs.azure.account.oauth2.client.secret": "<client-secret>",

  "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<tenant-
id>/oauth2/token"

}


dbutils.fs.mount(

  source = "abfss://<container-name>@<storage-account>.dfs.core.windows.net/",

  mount_point = "/mnt/mydata",

  extra_configs = configs)
```

---

4. Write a SQL query to list all employees who joined in the last 6 months.

```
SELECT *
FROM employees
WHERE join_date >= DATEADD(MONTH, -6, GETDATE());
```

Adjust function names (GETDATE() or CURRENT_DATE) depending on SQL dialect (SQL Server, MySQL, etc.)

---

5. How do you implement data validation and quality checks in ADF?

- Use **Data Flow** or **Stored Procedure** activities.
- Perform:
    - **Null checks**, **Data type checks**, **Range checks**, etc.
- Create **Validation activities** with expressions (e.g., row count > 0).
- Add **If Condition** or **Until Loop** for conditional logic.
- Log validation results to a control table or send alerts.
- Use **Custom Logging Framework** in ADF for monitoring.

---

6. Explain the concept of Azure Data Lake and its integration with SQL-based systems.

Azure Data Lake (ADLS) is a scalable, secure storage for big data.

Integration with SQL systems:

- Use **PolyBase** or **OPENROWSET** in Azure Synapse to query ADLS files.
- **External Tables** in Synapse map to files in ADLS.
- Use **ADF** to move/transform data between ADLS and SQL DBs.
- **Databricks** and **Azure SQL DB** can integrate for both ETL and analytics workloads.

This hybrid integration enables flexible, scalable, and cost-effective data architecture.

---

7. How do you handle exceptions and errors in Python?

Use **try-except-finally** blocks:

```
try:
    # risky code
    x = 10 / 0
except ZeroDivisionError as e:
```

```
    print(f"Error occurred: {e}")
except Exception as e:
    print(f"Unexpected error: {e}")
finally:
    print("Cleanup actions if needed.")
```

- Use logging for error logging instead of print.
- You can also raise custom exceptions using raise.

---

8. What is the process of normalization, and why is it required?

Normalization is organizing data to reduce redundancy and improve integrity.

Steps (Normal Forms):

- **1NF**: Remove repeating groups.
- **2NF**: Remove partial dependency.
- **3NF**: Remove transitive dependency.

Why it's required:

- Reduces data redundancy.
- Improves consistency and integrity.
- Makes data easier to maintain.

However, in analytics workloads, **denormalization** is preferred for performance.

PwC Data Engineering Interview Questions & Answers

---

1. Write a PySpark code to join two DataFrames and perform aggregation.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import sum

spark = SparkSession.builder.appName("JoinAgg").getOrCreate()

# Sample data
```

```
df1 = spark.createDataFrame([
    (1, "Sales", 1000),
    (2, "Marketing", 1500),
    (3, "Sales", 2000)
], ["id", "department", "salary"])

df2 = spark.createDataFrame([
    (1, "John"),
    (2, "Alice"),
    (3, "Bob")
], ["id", "name"])

# Join and aggregate
joined_df = df1.join(df2, "id")
agg_df = joined_df.groupBy("department").agg(sum("salary").alias("total_salary"))
agg_df.show()
```

---

2. What is the difference between wide and narrow transformations in Spark?

- **Narrow Transformation**:
    - Each input partition contributes to only one output partition.
    - Examples: map(), filter(), union()
    - No shuffle, faster.
- **Wide Transformation**:
    - Requires data to be shuffled across partitions.
    - Examples: groupBy(), join(), reduceByKey()
    - Involves a **shuffle**, more expensive in terms of performance.

---

3. How do you integrate Azure Synapse Analytics with other Azure services?

Azure Synapse integrates with:

- **ADLS Gen2**: Store and access data via linked services and workspaces.

- **Azure Data Factory**: Use ADF pipelines to load data into Synapse.
- **Power BI**: For real-time reporting and dashboards directly on Synapse data.
- **Azure Key Vault**: To manage credentials securely.
- **Azure ML**: To run ML models on Synapse using Spark pools or SQL analytics.

---

4. How do you monitor and troubleshoot data pipelines in Azure Data Factory?

- Use **Monitoring** tab in ADF UI for real-time run history.
- Check **activity run details** (duration, errors, input/output).
- Enable **diagnostic logging** to log analytics.
- Set up **Alerts and Metrics** in Azure Monitor.
- Implement custom **logging in pipelines** (store status/errors in log tables).
- Use **integration runtime logs** to analyze data movement issues.

---

5. Write Python code to generate prime numbers.

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True


# Generate first 10 primes
primes = []
num = 2
while len(primes) < 10:
    if is_prime(num):
        primes.append(num)
    num += 1
```

```
print(primes)
```

---

6. How do you optimize Python code for better performance?

- Use **built-in functions** and **list comprehensions**.
- Avoid unnecessary loops; prefer **vectorized operations** using libraries like NumPy or Pandas.
- Use **generators** instead of lists for memory efficiency.
- Profile code using cProfile or line_profiler.
- Use **multi-threading** or **multi-processing** for I/O-bound or CPU-bound tasks.
- Use efficient data structures (set, dict over list when appropriate).

---

7. Explain the concept of list comprehensions and provide an example.

List Comprehension is a concise way to create lists.

```
# Example: Create a list of squares from 1 to 5
squares = [x**2 for x in range(1, 6)]
print(squares)  # Output: [1, 4, 9, 16, 25]
```

It's equivalent to:

```
squares = []
for x in range(1, 6):
    squares.append(x**2)
```

List comprehensions are **faster** and more **readable**.

---

8. How do you implement disaster recovery and backup strategies for data in Azure?

- Use **Geo-redundant storage (GRS)** for automatic replication across regions.
- Enable **Azure Backup** and **Recovery Services Vault**.
- Schedule **regular backups** for Azure SQL, Synapse, and Blob/ADLS.
- Use **Soft Delete** and **Point-in-Time Restore** features.
- Implement **RA-GRS** (Read-Access Geo-Redundant Storage) for high availability.
- Document and test a **DR plan** using **Azure Site Recovery**.

Gopi Rayavarapu!

www.linkedin.com/in/gopi-rayavarapu-5b560020a