VISION BOARD

AZURE DATA ENGINEERING INTERVIEW Q&A TOPICWISE

DEVIKRISHNA R

LinkedIn: @Devikrishna R Email: visionboard044@gmail.com

TOPICS COVERED

- SQL
- PYTHON
- AZURE DATA FACTORY
- DATABRICKS
- PYSPARK
- DATA LAKE STORAGE (ADLS)
- DATA MODELING AND ARCHITECTURE
- OTHER QUESTIONS(AZURE DEVOPS, AZURE FUNCTIONS, LOGICAPPS AND GENERAL QUES)

QUESTIONS

SQL

1. Difference between TRUNCATE and DELETE

2. Difference between Stored Procedures and Functions

3. What are Views and CTEs (Common Table Expressions)?

4. How do aggregation functions differ from analytic functions in SQL?

5. Write SQL query to find top 3 earners in each department

6. Write SQL query to get the count of duplicates in a table

7. Use LAG and RANK functions in a query

8. Count occurrences of each word in a table column

9. How to find the second-highest salary in a table?

10. Write SQL query to convert row-level data to column-level using PIVOT

11. What is the difference between star schema, snowflake schema, and 3NF?

12. Explain ACID properties in SQL with example

13. How to create and use indexes (Clustered vs Non-Clustered)

14. Common Queries for Optimization

15. UNION vs UNION ALL: Differences in usage and performance

16. Concurrency issues in SQL and how to resolve them

17. Explain the role of Azure SQL Database in cloud-based data solutions

18. How do you set up and configure an Azure Data Factory pipeline for ETL?

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

20. How do you ensure high availability and disaster recovery for Azure SQL databases?

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

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

23. Explain the use of Azure Active Directory for SQL database authentication

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

25. Describe the process of using Azure SQL Data Sync for data replication

26. What are the best practices for database security in Azure SQL Database?

27. How do you integrate Azure SQL Database with other Azure services?

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

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

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

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

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

33. Describe the use of Azure Cosmos DB and its benefits over traditional SQL databases

34. How do you implement data partitioning in Azure SQL Data Warehouse (Synapse Dedicated SQL Pool)?

35. What are the common challenges in managing large-scale SQL databases in Azure?

36. How do you use Azure Data Lake Analytics with U-SQL for big data processing?

SQL IMPORTANT CODING QUESTIONS:

1. Find the second-highest salary in a table.

2. Write a SQL query to find the top 3 earners in each department.

3. Write a SQL query to find the nth highest salary in a table (e.g., 4th highest).

4. Write a SQL query to rank employees based on their performance score.

5. Write a SQL query to get the department-wise highest salary.

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

7. Write a SQL query to calculate the median salary of employees.

8. Write a SQL query to find the average salary of employees by department.

9. Count occurrences of each word in a table column.

10. Use LAG and RANK functions in a query.

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

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

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

14. Write a SQL query to create a new table with the results of a complex query.

15. Write a SQL query to join two tables and display specific columns.

16. Write SQL to find employees earning more than their manager.

17. Write a SQL query to find employees who do not have a manager.

18. Find records that exist in one table but not in another.

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

20. Get the count of duplicates in a table.

PYTHON

1. Explain the difference between lists, tuples, and sets in Python.

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

3. How do you use dictionaries in Python to store key-value pairs efficiently?

4. Describe the use of context managers in Python.

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

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

7. Generate Fibonacci numbers.

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

9. How do you handle missing or null values in a dataset using Python?

10. Identify duplicates in a list and count their occurrences.

11. Describe the process of data serialization and deserialization in Python.

12. How do you read and write data to and from a database using Python?

13. Explain the role of pandas library in data manipulation.

14. How do you integrate Python with big data tools like Apache Spark?

PYTHON CODING QUESTIONS

1. Check if a string is a palindrome.

2. Replace vowels in a string with spaces.

3. Count word occurrences in a string.

4. Generate prime numbers.

5. Find consecutive numbers in a list (e.g., print the number that appears consecutively 3 times).

AZURE DATA FACTORY

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

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

3. Explain the role of ADF in a hybrid data integration scenario.

4. How do full loads differ from incremental loads in ADF?

5. How to implement incremental load in ADF?

6. Explain the process of copying large files (e.g., 3TB) from on-

premises to Azure in ADF.

7. What is the binary copy method, and when is it used?

8. How to perform parallel copies in ADF using partitioning?

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

10. How to handle error handling in ADF using retry, try-catch blocks, and failover mechanisms?

11. How to implement error retry policies in ADF pipelines?

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

13. How do you handle schema drift in ADF?

14. How to implement data validation and data quality checks in ADF?

15. How to implement data masking in ADF for sensitive data?

16. Describe the steps to migrate SSIS packages to ADF.

17. How do you handle complex ADF pipelines? Explain challenges faced.

18. How to connect to Salesforce using ADF securely?

19. Where to store sensitive information like passwords in ADF (e.g., Azure Key Vault)?

20. How to pass parameters from ADF to Databricks notebooks?

21. Explain how to use ADF with Azure Databricks for complex transformations.

22. How do you optimize the performance of ADF pipelines?

23. Trigger types in ADF (Schedule, Tumbling Window, etc.).

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

DATABRICKS

1. Explain Databricks Lakehouse architecture (Bronze, Silver, Gold layers).
2. What is the difference between a job cluster and an interactive cluster?
3. What is Unity Catalog in Databricks?
4. What are Delta logs, and how to track data versioning in Delta tables?
5. Explain the difference between caching and persisting in Databricks.
6. Explain Databricks optimization techniques (e.g., file format, partitioning, caching).
7. How do autoscaling clusters work in Databricks?
8. How to copy files from Blob Storage to Databricks?
9. How to connect ADLS (Azure Data Lake Storage) to Databricks?
10. How to run one notebook from another notebook using %run or dbutils.notebook.run()?
11. How to create and deploy notebooks in Databricks?
12. How to manage and automate ETL workflows using Databricks Workflows?
13. How to use Databricks Delta Live Tables for continuous data

processing?

14. How to monitor Databricks jobs?

15. How to implement data validation and quality checks in Databricks?

16. How to implement SCD Type 1 and Type 2 using PySpark in Databricks?

17. How to give permission to specific notebooks or clusters to other users?

18. What are the best practices for securing data in Databricks?

19. How to integrate Databricks with Azure DevOps for CI/CD pipelines?

PYSPARK

1. Explain lazy evaluation in PySpark.

2. Difference between groupByKey and reduceByKey.

3. What is the difference between repartition and coalesce?

4. Explain broadcast join in PySpark.

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

6. What is the purpose of SparkContext and SparkSession?

7. How does caching work in PySpark?

8. Difference between wide and narrow transformations.

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

10. What is Z-ordering in Spark?

11. Explain Delta Lake and its benefits over Parquet.

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

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

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

15. How to persist and cache data in PySpark?

16. How do you design and implement data pipelines using Azure Data Factory (ADF)?

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

18. Describe the process of setting up and managing an Azure SQL Database.

19. How do you optimize data storage and retrieval in Azure Data Lake Storage (ADLS)?

20. Explain the use of Azure Synapse Analytics and how it integrates with other Azure services.

21. Difference between Spark SQL and PySpark DataFrame APIs

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

23. DataFrame API syntax to filter and aggregate data.

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

25. Describe the process of integrating Azure Databricks with Azure Data Lake Storage.

26. Write a PySpark code to: read a CSV, join two DataFrames, perform aggregation, and filter rows.

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

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

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

30. How do you ensure data consistency and reliability in distributed systems on Azure?

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

32. How do you manage and automate data workflows using Azure Logic Apps?

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

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

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

36. What are the different data partitioning strategies in Azure SQL Data Warehouse?

37. How do you use Azure Monitor to track and analyze performance metrics of your data infrastructure?

38. Describe the process of performing ETL operations using Azure Data Factory.

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

DATA LAKE STORAGE (ADLS)

1. How do you integrate ADLS with Azure Databricks for data processing?
2. Why is mounting preferred over using access keys to connect Databricks with ADLS?
3. What are the security features available in ADLS (e.g., access control lists, role-based access)?
4. How do you implement data encryption at rest and in transit in ADLS?
5. How do you handle schema evolution in ADLS?
6. Explain how to copy all files with different formats (CSV, Parquet, Excel) from a source to target.
7. How to track file names in the output table while performing copy operations in ADF?
8. How do you handle large-scale data ingestion into ADLS?
9. How do you monitor and troubleshoot issues in ADLS?
10. What are the differences between ADLS Gen1 and Gen2?
11. Explain the use of hierarchical namespaces in ADLS.
12. How do you manage data lifecycle policies in ADLS?
13. How do you implement versioning in ADLS?
14. How do you ensure data quality and validation in ADLS?
15. Describe the process of setting up and managing access control lists (ACLs) in ADLS.

16. How do you integrate ADLS with Azure Synapse Analytics?

17. Explain the process of auditing and logging access to data in ADLS.

18. How to optimize data retrieval from ADLS in Spark?

19. How do you manage and optimize storage costs in ADLS?

20. Explain the process of setting up disaster recovery for ADLS.

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

DATA MODELING AND ARCHITECTURE

1.Difference between a database, data warehouse, and data lake

2.What is data mart, and how does it differ from a data warehouse?

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

4.How do you design a star schema for a sales reporting system?

5.Explain the use of surrogate keys in dimensional modeling

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

7.How do you handle slowly changing dimensions (SCD) in data modeling?

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

9.Difference between ER modeling and dimensional modeling

10.What are the different types of data schemas, and how do you choose the right one for your data model?

11.Build a simple data warehouse for a grocery store

12.Describe the role of metadata in data modeling and data architecture

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

14.Explain the deployment architecture of a data pipeline involving ADF, Databricks, and ADLS

15.How do you ensure data consistency and integrity in a distributed data architecture?

16.How do you optimize data models for performance and scalability?

OTHER QUESTIONS

1. What is DAG in Spark, and how does it work?

2. Explain Spark architecture (Driver vs Worker).

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

4. How does fault tolerance work in ADF and Databricks?

5. What challenges have you faced in managing large datasets (e.g., 3TB+ files)?

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

7. How to improve the performance of a Spark job?

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

9. What is a Unity Catalog in Databricks?

10. Explain the role of integration runtimes in hybrid scenarios (on-prem to cloud).

11. How to design an end-to-end data pipeline architecture?

12. Snowflake-specific: How does it compare to Delta Lake or other data lakes?

13. When would you choose Snowflake over other platforms?

14. Difference between Azure Logic Apps, Azure Functions, and ADF.

15. Azure Synapse Analytics: How does it compare to Databricks?

16. How to use Azure DevOps for CI/CD pipelines

17. What are the key features of Azure DevOps?

18. How do you implement CI and CD in Azure DevOps?

19. Explain the role of pipelines in Azure DevOps

20. How do you manage dependencies and versioning in Azure DevOps?

21. What are Azure Functions, and how do they differ from traditional web services?

22. How do you deploy and manage Azure Functions?

23. Use cases for Azure Functions in data engineering.

24. How do you secure secrets and keys using Azure Key Vault?

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

26. What are the benefits of using Azure Synapse for big data processing?

27. How does Azure Synapse integrate with other Azure services?

28. Difference between dedicated and serverless SQL pool in Synapse.

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

30. Describe the process of data ingestion in Synapse

31. Best practices for data partitioning and distribution in Synapse.

32. How do you implement security and compliance in Synapse?

33. Concept of data integration and transformation in ADF.

34. Error handling and retry mechanisms in ADF.

35. Role of triggers and schedules in ADF.

36. How do you use Azure Monitor to analyze performance in Azure services?

ANSWERS

SQL

1. Difference between TRUNCATE and DELETE

Feature	`TRUNCATE`	`DELETE`
Logging	Minimal logging	Fully logged
WHERE Clause	Not supported	Supported
Triggers	Not fired	Triggers get fired
Rollback	Not always possible	Fully supported
Speed	Faster	Slower (row-by-row)

---

2. Difference between Stored Procedures and Functions

Feature	Stored Procedure	Function
Return Type	Zero or multiple	Must return one value
Usage in SELECT	Not allowed	Allowed
Transactions	Can contain	Cannot manage
Purpose	Complex logic	Reusable computations

---

3. What are Views and CTEs (Common Table Expressions)?

- **Views**:
  A virtual table created using a SELECT query. Data is not stored; it's fetched dynamically.

```sql
CREATE VIEW ActiveEmployees AS
SELECT * FROM Employees WHERE isActive = 1;
```

- **CTE (Common Table Expression)**:
  A temporary result set used within a SELECT, INSERT, UPDATE, or DELETE.

```sql
WITH TopSalaries AS (
  SELECT *, RANK() OVER (ORDER BY salary DESC) AS rnk
  FROM Employees
)
SELECT * FROM TopSalaries WHERE rnk <= 5;
```

---

4. How do aggregation functions differ from analytic functions in SQL?

Feature	Aggregation Functions	Analytic Functions
Output per group	One result per group	One result per row
Examples	`SUM()`, `AVG()`	`ROW_NUMBER()`, `LAG()`, `RANK()`
Use Case	Group totals	Row-wise comparisons, trends

---

5. Write SQL query to find top 3 earners in each department

```sql
SELECT * FROM (
  SELECT *, DENSE_RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS rnk
  FROM Employees
) ranked
WHERE rnk <= 3;
```

---

6. Write SQL query to get the count of duplicates in a table

```sql
SELECT column_name, COUNT(*) AS count
FROM TableName
GROUP BY column_name
HAVING COUNT(*) > 1;
```

---

7. Use LAG and RANK functions in a query

```sql
SELECT emp_id, salary,
       LAG(salary) OVER (ORDER BY salary) AS previous_salary,
       RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM Employees;
```

---

8. Count occurrences of each word in a table column

(SQL Server example using STRING_SPLIT)

```sql
WITH WordSplit AS (
  SELECT value AS word
  FROM TableName
  CROSS APPLY STRING_SPLIT(column_name, ' ')
)
SELECT word, COUNT(*) AS frequency
FROM WordSplit
GROUP BY word;
```

---

9. How to find the second-highest salary in a table?

Method 1: Using Subquery

```sql
SELECT MAX(salary)
FROM Employees
WHERE salary < (SELECT MAX(salary) FROM Employees);
```

Method 2: Using DENSE_RANK()

```sql
SELECT salary
FROM (
  SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS rnk
  FROM Employees
) ranked
WHERE rnk = 2;
```

Method 3: Using LIMIT OFFSET (MySQL)

```sql
SELECT DISTINCT salary FROM Employees ORDER BY salary DESC LIMIT 1 OFFSET 1;
```

---

10. Write SQL query to convert row-level data to column-level using PIVOT

```sql
SELECT *
FROM (
  SELECT employee_id, year, salary
  FROM SalaryTable
) src
PIVOT (
  MAX(salary) FOR year IN ([2022], [2023], [2024])
) AS pvt;
```

---

11. What is the difference between star schema, snowflake schema, and 3NF?

Feature	Star Schema	Snowflake Schema	3NF (Third Normal Form)
Structure	Denormalized	Normalized dimensions	Fully normalized
Query Performance	Fast	Slower joins	Moderate speed
Redundancy	High	Low	Minimal
Use Case	OLAP (Analytics)	OLAP (Detailed Analysis)	OLTP (Transactional DBs)

---

12. Explain ACID properties in SQL with example

- **A - Atomicity**: All operations in a transaction are done or none.
- **C - Consistency**: Data must be valid before and after transaction.
- **I - Isolation**: Transactions do not interfere with each other.
- **D - Durability**: Committed data is saved even if system crashes.

Example:

```
BEGIN TRANSACTION;
UPDATE Account SET balance = balance - 100 WHERE acc_id = 'A';
UPDATE Account SET balance = balance + 100 WHERE acc_id = 'B';
COMMIT;
```

---

13. How to create and use indexes (Clustered vs Non-Clustered)

- **Clustered Index**: Sorts and stores the data rows in the table.

```
CREATE CLUSTERED INDEX idx_emp_salary ON Employees(salary);
```

- **Non-Clustered Index**: Has a separate structure from data rows.

```
CREATE NONCLUSTERED INDEX idx_emp_name ON Employees(name);
```

---

14. Common Queries for Optimization

- **Find records that exist in one table but not in another**

```
SELECT * FROM TableA
WHERE NOT EXISTS (
  SELECT 1 FROM TableB WHERE TableA.id = TableB.id
);
```

- **Find employees earning more than their manager**

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

---

15. UNION vs UNION ALL: Differences in usage and performance

Feature	UNION	UNION ALL
Removes Duplicates	Yes	No
Performance	Slower (due to sorting)	Faster
Use Case	When unique records needed	When duplicates are okay

---

16. Concurrency issues in SQL and how to resolve them

- **Issues**: Dirty reads, lost updates, phantom reads.
- **Resolutions**:
  - Use appropriate **transaction isolation levels**
  - Implement **locking strategies**
  - Use **optimistic concurrency control** (e.g., timestamps)

---

17. Explain the role of Azure SQL Database in cloud-based data solutions

- PaaS (Platform as a Service) for relational databases
- Supports auto-scaling, geo-replication, backups
- Built-in monitoring and security
- Useful for cloud-native or hybrid data apps
- Integrates well with **Azure Data Factory**, **Power BI**, **Synapse**

---

18. How do you set up and configure an Azure Data Factory pipeline for ETL?

1. **Create Azure Data Factory** in Azure portal
2. Go to **Author & Monitor**
3. Create a **pipeline**
4. Add **source** (e.g., Blob Storage, SQL DB)
5. Add **transformation activities** (e.g., Data Flow, Mapping Data Flow)
6. Add **sink** (destination)
7. Debug and **validate pipeline**

8. Add **trigger** (Schedule or Manual)
9. Use **Monitor** tab to track runs

---

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

Feature	Azure SQL Database	Azure SQL Managed Instance
Use Case	Modern cloud apps	Lift-and-shift on-prem SQL apps
SQL Server Agent	Not supported	Supported
VNET Integration	Limited	Full
Feature Compatibility	Limited	Near 100% parity with SQL Server
Backups	Automated	Automated

---

20. How do you ensure high availability and disaster recovery for Azure SQL databases?

- **High Availability**:
  - **Built-in** with 99.99% SLA
  - Active geo-replication
  - Zone-redundant replicas
- **Disaster Recovery**:
  - Point-in-time restore
  - Auto-failover groups
  - Long-term retention policies

---

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

- Use **Query Store** to analyze and fix regressions
- Create proper **indexes** (clustered, non-clustered)
- **Avoid SELECT *** and reduce data scans
- Use **parameterized queries**
- Enable **automatic tuning**

---

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

- **Azure Monitor**: Insights into performance
- **Log Analytics & Alerts**: Custom monitoring
- **Query Performance Insight**: Pinpoints slow queries
- **Dynamic Management Views (DMVs)**: Deep diagnostics
- **SQL Auditing & Diagnostics Logs**

---

23. Explain the use of Azure Active Directory for SQL database authentication

- Enables **centralized identity** management
- Supports **Multi-Factor Authentication (MFA)**
- Prevents hard-coded SQL logins
- Users can connect using Azure AD credentials:

```
CONNECT WITH AZURE_AD_USER;
```

---

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

- **Encryption in Transit**: TLS/SSL
- **At Rest**: Transparent Data Encryption (TDE)
- **Column-Level Encryption (CLE)**: Specific sensitive columns
- **Always Encrypted**: Data remains encrypted in-use

---

25. Describe the process of using Azure SQL Data Sync for data replication

- Use **Azure Portal** to set up **Hub & Spoke** topology
- Define **Sync Group** and **conflict resolution policy**
- Connect databases and schedule sync
- Supports **bi-directional** sync between Azure and on-prem SQL

---

26. What are the best practices for database security in Azure SQL Database?

- Use **firewall rules** & **private endpoints**
- Enforce **TDE** and **Always Encrypted**
- Use **Azure AD Authentication**
- Enable **Auditing** and **Threat Detection**
- Apply **RBAC and least privilege principle**

---

27. How do you integrate Azure SQL Database with other Azure services?

- **Power BI**: For reporting & visualization
- **Azure Data Factory**: For ETL workflows

- **Azure Functions & Logic Apps**: For automation
- **Event Grid**: For event-driven processing
- **Azure Synapse**: For analytics

---

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

- Use **Azure Data Migration Assistant (DMA)** to assess compatibility
- Use **Azure Database Migration Service (DMS)** to transfer schema and data
- Choose **online or offline** migration
- Post-migration validation & optimization

---

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

- Choose between **SQL, NoSQL, and Lake** depending on use case
- Use **partitioning, sharding, and caching**
- Design for **fault tolerance and auto-scaling**
- Leverage **event-driven pipelines** with **Event Hub + Data Factory**
- Monitor with **Azure Monitor** & **Log Analytics**

---

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

- Create a **Logic App** using templates or designer
- Use triggers like **HTTP requests, schedule, or events**

- Add actions such as **“Execute SQL query”**
- Can integrate with **Outlook, SharePoint, Teams, etc.**

---

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

- Azure Data Lake: Massive storage for **structured and unstructured data**
- Stores data in **HDFS-compatible format**
- Integrated with **Synapse, Data Factory, Databricks**
- Query using **T-SQL via PolyBase** or **serverless pools**

---

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

- Create **Synapse workspace**
- Add **linked services** (Azure SQL, Data Lake, Blob, etc.)
- Use **Data Flows or Notebooks** for transformation
- Create **pipelines** for orchestration
- Monitor activities from the **Monitor tab**

---

33. Describe the use of Azure Cosmos DB and its benefits over traditional SQL databases

- Globally distributed, **multi-model NoSQL database**
- Offers **millisecond latency, auto-scaling**, and **99.999% availability**
- Supports **SQL API, MongoDB, Cassandra, Gremlin, Table APIs**

- Ideal for IoT, personalization engines, and global apps

---

34. How do you implement data partitioning in Azure SQL Data Warehouse (Synapse Dedicated SQL Pool)?

- Use **distribution methods**: Hash, Round Robin, Replicated
- Define **distribution column** in the table creation
- Example:

```
CREATE TABLE Sales (
  SaleID INT,
  Region NVARCHAR(50)
)
WITH (DISTRIBUTION = HASH(Region));
```

- Helps with **parallel processing** and **query performance**

---

35. What are the common challenges in managing large-scale SQL databases in Azure?

- **Query performance degradation**
- **Cost control** and usage monitoring
- **Data sync latency** in hybrid environments
- **Managing large volumes of telemetry and logs**
- **Scaling connections** under heavy loads

---

36. How do you use Azure Data Lake Analytics with U-SQL for big data processing?

- **U-SQL**: Hybrid of SQL + C#
- Used for processing large files in **Data Lake Storage Gen1**

- Example:

```
@data = EXTRACT Name string, Age int FROM "/data/people.csv"
USING Extractors.Csv();

OUTPUT @data TO "/output/people.txt" USING Outputters.Tsv();
```

SQL IMPORTANT CODING QUESTIONS:

1. Find the second-highest salary in a table.

Method 1: Using DISTINCT + ORDER BY + OFFSET

```
SELECT DISTINCT Salary
FROM Employees
ORDER BY Salary DESC
OFFSET 1 ROW FETCH NEXT 1 ROW ONLY;
```

Method 2: Using subquery

```
SELECT MAX(Salary)
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

---

2. Write a SQL query to find the top 3 earners in each department.

```
SELECT *
FROM (
    SELECT *,
           RANK() OVER (PARTITION BY DepartmentID ORDER BY Salary DESC) AS Rank
    FROM Employees
) AS Ranked
WHERE Rank <= 3;
```

3. Write a SQL query to find the nth highest salary in a table (e.g., 4th highest).

```sql
SELECT DISTINCT Salary
FROM Employees
ORDER BY Salary DESC
OFFSET 3 ROWS FETCH NEXT 1 ROW ONLY;
```

---

4. Write a SQL query to rank employees based on their performance score.

```sql
SELECT *, RANK() OVER (ORDER BY PerformanceScore DESC) AS Rank
FROM Employees;
```

---

5. Write a SQL query to get the department-wise highest salary.

```sql
SELECT DepartmentID, MAX(Salary) AS MaxSalary
FROM Employees
GROUP BY DepartmentID;
```

---

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

```sql
SELECT EmployeeID, Salary,
       SUM(Salary) OVER (ORDER BY EmployeeID) AS RunningTotal
FROM Employees;
```

---

7. Write a SQL query to calculate the median salary of employees.

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Salary) AS MedianSalary
FROM Employees;
```

---

8. Write a SQL query to find the average salary of employees by department.

```
SELECT DepartmentID, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY DepartmentID;
```

---

9. Count occurrences of each word in a table column.

Assume the column Feedback contains multiple words:

```
SELECT value AS Word, COUNT(*) AS Occurrences
FROM Employees
CROSS APPLY STRING_SPLIT(Feedback, ' ')
GROUP BY value;
```

---

10. Use LAG and RANK functions in a query.

```
SELECT EmployeeID, Name, Salary,
       LAG(Salary) OVER (ORDER BY Salary) AS PreviousSalary,
       RANK() OVER (ORDER BY Salary DESC) AS Rank
FROM Employees;
```

---

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

```sql
SELECT Number + 1 AS MissingFrom
FROM NumbersTable
WHERE Number + 1 NOT IN (SELECT Number FROM NumbersTable);
```

---

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

```sql
SELECT *
FROM (
    SELECT DepartmentID, Gender
    FROM Employees
) AS SourceTable
PIVOT (
    COUNT(Gender)
    FOR Gender IN ([Male], [Female])
) AS PivotTable;
```

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

```sql
WITH CTE AS (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY Name, Salary ORDER BY ID) AS rn
    FROM Employees
)
DELETE FROM CTE WHERE rn > 1;
```

---

14. Write a SQL query to create a new table with the results of a complex query.

```sql
SELECT DepartmentID, AVG(Salary) AS AvgSalary
INTO DepartmentAverageSalary
FROM Employees
GROUP BY DepartmentID;
```

---

15. Write a SQL query to join two tables and display specific columns.

```sql
SELECT e.EmployeeID, e.Name, d.DepartmentName
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID;
```

---

16. Write SQL to find employees earning more than their manager.

```sql
SELECT e.EmployeeID, e.Name
FROM Employees e
JOIN Employees m ON e.ManagerID = m.EmployeeID
WHERE e.Salary > m.Salary;
```

---

17. Write a SQL query to find employees who do not have a manager.

```sql
SELECT *
FROM Employees
WHERE ManagerID IS NULL;
```

---

18. Find records that exist in one table but not in another.

```sql
SELECT *
FROM TableA
WHERE ID NOT IN (SELECT ID FROM TableB);
```

Or using LEFT JOIN:

```sql
SELECT a.*
FROM TableA a
LEFT JOIN TableB b ON a.ID = b.ID
WHERE b.ID IS NULL;
```

---

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

```sql
SELECT *
FROM Employees
WHERE JoiningDate >= DATEADD(MONTH, -6, GETDATE());
```

---

20. Get the count of duplicates in a table.

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

PYTHON

1. Explain the difference between lists, tuples, and sets in Python.

- **List**: Ordered, mutable, allows duplicates.
  example = [1, 2, 2, 3]
- **Tuple**: Ordered, immutable, allows duplicates.
  example = (1, 2, 3)
- **Set**: Unordered, mutable, no duplicates.
  example = {1, 2, 3}

---

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

A concise way to create lists using a single line of code.

squares = [x**2 for x in range(5)]

---

3. How do you use dictionaries in Python to store key-value pairs efficiently?

Dictionaries use a **hash table** internally for fast lookup and retrieval.

```
student = {'name': 'Alice', 'age': 24}
print(student['name'])  # Output: Alice
```

---

4. Describe the use of context managers in Python.

They manage resources like files or database connections using with.

```
with open('file.txt', 'r') as f:
    data = f.read()
```

---

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

Use try, except, finally blocks to catch and manage errors.

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
finally:
    print("Done")
```

---

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

- Use built-in functions.
- Avoid redundant loops.
- Use set and dict for fast lookups.
- Use generators and list comprehensions.
- Use libraries like NumPy or Cython for computation-heavy tasks.

---

7. Generate Fibonacci numbers.

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b

fibonacci(10)
```

---

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

```
import pandas as pd

df = pd.DataFrame({'Name': ['John Doe', 'Jane Smith']})
df[['FirstName', 'LastName']] = df['Name'].str.split(' ', 1, expand=True)
```

---

9. How do you handle missing or null values in a dataset using Python?

Using pandas:

```
df.fillna(0)          # Replace with 0
df.dropna()           # Drop rows with missing values
df.isnull().sum()     # Check missing count per column
```

---

10. Identify duplicates in a list and count their occurrences.

```
from collections import Counter

lst = [1, 2, 2, 3, 4, 4, 4]
duplicates = Counter(lst)
print({k: v for k, v in duplicates.items() if v > 1})
```

---

11. Describe the process of data serialization and deserialization in Python.

Used for saving/loading data structures to/from files using pickle, json.

```python
import json

data = {'name': 'Alice'}
with open('data.json', 'w') as f:
    json.dump(data, f)  # Serialization

with open('data.json', 'r') as f:
    new_data = json.load(f)  # Deserialization
```

---

12. How do you read and write data to and from a database using Python?

Using sqlite3 or SQLAlchemy:

```python
import sqlite3

conn = sqlite3.connect('test.db')
cursor = conn.cursor()
cursor.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER, name TEXT)")
cursor.execute("INSERT INTO users VALUES (?, ?)", (1, 'Alice'))
conn.commit()
rows = cursor.execute("SELECT * FROM users").fetchall()
print(rows)
conn.close()
```

---

13. Explain the role of pandas library in data manipulation.

- Powerful for data cleaning, transformation, and analysis.
- Functions: read_csv, groupby, merge, pivot, dropna, etc.

```python
import pandas as pd

df = pd.read_csv('data.csv')
df.groupby('Category').mean()
```

---

14. How do you integrate Python with big data tools like Apache Spark?

Using **PySpark**, the Python API for Spark.

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()
df = spark.read.csv("data.csv", header=True, inferSchema=True)
df.show()
```

PYTHON CODING QUESTIONS

1. Check if a string is a palindrome.

A palindrome reads the same backward and forward.

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

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

---

2. Replace vowels in a string with spaces.

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

print(replace_vowels("Hello World"))  # "H ll  W rld"
```

3. Count word occurrences in a string.

```
from collections import Counter

def word_count(s):
    words = s.lower().split()
    return Counter(words)

print(word_count("Apple banana apple orange banana banana"))
# {'apple': 2, 'banana': 3, 'orange': 1}
```

---

4. Generate prime numbers.

Here's a function to generate all primes up to n.

```
def generate_primes(n):
    primes = []
    for num in range(2, n+1):
        for p in primes:
            if num % p == 0:
                break
        else:
            primes.append(num)
    return primes

print(generate_primes(20))
```

---

5. Find consecutive numbers in a list (e.g., print the number that appears consecutively 3 times).

```python
def find_consecutive_triplets(lst):
    for i in range(len(lst) - 2):
        if lst[i] == lst[i+1] == lst[i+2]:
            return lst[i]
    return None

print(find_consecutive_triplets([1, 2, 2, 2, 3, 4]))  # 2
```

AZURE DATA FACTORY (ADF)

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

Activities in ADF define tasks in a pipeline:

- **Copy Activity**: Moves data from source to sink.
- **Data Flow Activity**: Transforms data at scale.
- **Notebook Activity**: Triggers Azure Databricks notebooks.
- **Stored Procedure, Web, Lookup, and If Condition Activities**: Help with logic and integration.

---

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

- **Azure IR**: Handles cloud data movement and transformation.
- **Self-hosted IR**: Used for on-premises data movement.
- **Azure SSIS IR**: Executes SSIS packages in Azure.

---

3. Explain the role of ADF in a hybrid data integration scenario.

ADF enables integration between on-premises and cloud systems using **Self-hosted IR**, enabling secure data movement, transformation, and orchestration in hybrid environments.

---

4. How do full loads differ from incremental loads in ADF?

- **Full Load**: Moves all data every time.
- **Incremental Load**: Moves only new or changed data (using watermark columns, timestamps, etc.).

---

5. How to implement incremental load in ADF?

- Use **Lookup** or **Stored Procedure** to get last watermark value.
- Use it in **Source query filter** (WHERE ModifiedDate > lastWatermark).
- Update watermark post-load via Stored Procedure or Web Activity.

---

6. Explain the process of copying large files (e.g., 3TB) from on-premises to Azure in ADF.

- Use **Self-hosted IR** for secure access.
- Enable **staging** to temporarily load into Azure Blob/ADLS.
- Use **parallelism** and **compression** for optimization.
- Monitor with **Activity Run output**.

---

7. What is the binary copy method, and when is it used?

- It copies data **as-is** without parsing (e.g., images, PDFs, zipped files).
- Used when content doesn't need transformation.

---

8. How to perform parallel copies in ADF using partitioning?

- Use **Source partitioning** in Copy Activity.
- Methods: **Dynamic Range**, **Static Range**, or **Hash partitioning** based on a column (e.g., CustomerID).

---

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

- Create **control tables** with table names, columns, and queries.
- Use **Lookup + ForEach** to dynamically loop through and copy.
- Apply **parameterization** for dataset and linked service names.

---

10. How to handle error handling in ADF using retry, try-catch blocks, and failover mechanisms?

- Use **Retry policy** in activities.
- Create **Try block** using If Condition and **Catch block** using failure path.
- Log errors using **Stored Procedure** or **Append Blob**.
- Use **global parameters** for fallback/failover routes.

---

11. How to implement error retry policies in ADF pipelines?

- Each activity has a retry setting.
- Example:

```
"policy": {
  "retry": 3,
  "retryIntervalInSeconds": 30,
  "timeout": "7.00:00:00"
}
```

---

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

- Use **Monitor tab** in ADF UI.
- Enable **Log Analytics** or **Application Insights**.
- Set up **Alerts** via Azure Monitor.

---

13. How do you handle schema drift in ADF?

- Use **Mapping Data Flows** with Allow schema drift enabled.
- Enable **Auto Mapping**.
- Use **Derived Column** to adjust changing columns dynamically.

---

14. How to implement data validation and data quality checks in ADF?

- Create **validation queries** with Lookup or Stored Procedure.

- Use **Assert Activity** or custom logic via Data Flow or Notebook.
- Log results in **audit tables** or alerts.

---

15. How to implement data masking in ADF for sensitive data?

- Use **Derived Column** in Data Flows to mask data (e.g., REPLACE with asterisks).
- Optionally use **Azure Purview** for classification.

---

16. Describe the steps to migrate SSIS packages to ADF.

- Deploy SSIS packages to **Azure SSIS IR**.
- Use **ADF pipeline** to execute the SSIS package.
- Configure **SSISDB** and required components in Azure.

---

17. How do you handle complex ADF pipelines? Explain challenges faced.

- Use **modular pipelines** and **pipeline chaining** via Execute Pipeline.
- Use **global parameters**, **ARM templates**, and **config tables**.
- Challenges: managing dependencies, debugging nested activities, dynamic datasets.

---

18. How to connect to Salesforce using ADF securely?

- Use **Salesforce connector** with OAuth authentication.
- Store secrets like client secret/token in **Azure Key Vault**.

---

19. Where to store sensitive information like passwords in ADF (e.g., Azure Key Vault)?

- Store secrets in **Azure Key Vault**.
- Reference in **Linked Services** via @Microsoft.KeyVault secure string.

---

20. How to pass parameters from ADF to Databricks notebooks?

- Use **Notebook Activity**.
- Pass parameters as a JSON dictionary:

```
{
  "base_path": "/mnt/data",
  "file_name": "sales.csv"
}
```

---

21. Explain how to use ADF with Azure Databricks for complex transformations.

- Trigger **Notebook Activity** from ADF.
- Use **mounts** or direct blob access in Databricks.
- Return results to ADF using **dbutils.notebook.exit()**.

---

22. How do you optimize the performance of ADF pipelines?

- Use **partitioning**, **staging**, and **compression**.
- Minimize data transformation inside Copy Activities.
- Use **parallel activities**, **cache in Data Flows**, and **managed VNETs**.

---

23. Trigger types in ADF (Schedule, Tumbling Window, etc.).

- **Schedule Trigger**: Runs on defined time.
- **Tumbling Window**: Processes data in time-based slices.
- **Event Trigger**: Runs on blob/file event.
- **Manual Trigger**: On-demand.

---

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

- Use **Synapse as sink** in Copy Activity.
- Leverage **PolyBase** for large files.
- Use ADF to orchestrate **Spark notebooks** or **SQL scripts** in Synapse.

DATABRICKS

1. Explain Databricks Lakehouse architecture (Bronze, Silver, Gold layers).

- **Bronze**: Raw, ingested data from source systems.
- **Silver**: Cleaned, joined, and filtered data.
- **Gold**: Aggregated, business-level metrics for BI or analytics.

---

2. What is the difference between a job cluster and an interactive cluster?

- **Job Cluster**: Created for a specific job and terminates automatically. Used in production.

- **Interactive Cluster**: Persistent, used for development and exploration.

---

3. What is Unity Catalog in Databricks?

- A unified governance solution for all data and AI assets.
- Provides centralized access control, audit logging, and metadata across all workspaces.

---

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

- Delta logs are **transaction logs** stored in _delta_log/ folder.
- Versioning enables **time travel**:

```
SELECT * FROM table VERSION AS OF 5;
```

---

5. Explain the difference between caching and persisting in Databricks.

- **Cache**: Stores DataFrame in memory (default MEMORY_AND_DISK).
- **Persist**: Offers storage-level options like memory, disk, or both.

```
df.cache()          # or
df.persist(StorageLevel.DISK_ONLY)
```

---

6. Explain Databricks optimization techniques (e.g., file format, partitioning, caching).

- Use **Delta format** for faster reads/writes.
- Partition large datasets by date, region, etc.
- Enable **ZORDER** for file pruning:

```
OPTIMIZE table ZORDER BY (col);
```

7. How do autoscaling clusters work in Databricks?

- Automatically adjusts number of worker nodes based on workload.
- Saves cost and improves performance.
- Enabled via the cluster configuration UI.

---

8. How to copy files from Blob Storage to Databricks?

Mount the storage or use direct access:

```
dbutils.fs.mount(
  source = "wasbs://container@account.blob.core.windows.net/",
  mount_point = "/mnt/mydata",
  extra_configs = {"<conf-key>":dbutils.secrets.get(scope="myscope", key="storagekey")}
)
```

---

9. How to connect ADLS (Azure Data Lake Storage) to Databricks?

Use **service principal** or **managed identity** with OAuth configs:

```
spark.conf.set("fs.azure.account.auth.type", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type", "org.apache
    .hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
# ... set tenant ID, client ID, and secret
```

---

10. How to run one notebook from another notebook using %run or dbutils.notebook.run()?

- %run ./Utilities → runs and shares variables.
- dbutils.notebook.run("notebook_name", timeout, {"param": "value"}) → for modular jobs.

---

11. How to create and deploy notebooks in Databricks?

- Create from UI or import .dbc/.ipynb files.
- Use **Repos** for Git integration.
- Deploy via **Databricks Workflows**, Jobs UI, or CLI.

---

12. How to manage and automate ETL workflows using Databricks Workflows?

- Use **Jobs UI** or JSON config to chain notebooks/tasks.
- Support dependencies, retries, and triggers (daily, event-based).
- Monitor using job runs view.

---

13. How to use Databricks Delta Live Tables for continuous data processing?

- Define DLT pipelines using @dlt.table in Python.
- Use **LIVE mode** for continuous ingestion.
- Automatically handles schema inference, dependencies, and quality.

---

14. How to monitor Databricks jobs?

- Use the **Jobs UI** for logs, status, and run history.
- Integrate with **Azure Monitor** or **Slack/Teams alerts** for failures.

---

15. How to implement data validation and quality checks in Databricks?

- Use **Delta Expectations** with Delta Live Tables:

```
@dlt.expect("valid_age", "age > 0")
@dlt.table
def clean_data(): ...
```

- Manual assertions via PySpark or pandas profiling.

---

16. How to implement SCD Type 1 and Type 2 using PySpark in Databricks?

- **Type 1**: Overwrite existing rows.

```
df.write.format("delta").mode("overwrite").save("/mnt/target")
```

- **Type 2**: Maintain history using merge with effective dates and flags.

---

17. How to give permission to specific notebooks or clusters to other users?

- Go to notebook > Permissions > Add user/group.
- Use **Cluster Policies** to restrict cluster configurations.
- Manage access via **Unity Catalog** for table-level permissions.

---

18. What are the best practices for securing data in Databricks?

- Use **Unity Catalog** for fine-grained access.
- Store credentials in **Azure Key Vault**.
- Enable **private link**, audit logs, and secure workspace settings.

---

19. How to integrate Databricks with Azure DevOps for CI/CD pipelines?

- Use **Repos** for Git integration.
- Use **Databricks CLI** or REST APIs in Azure Pipelines.
- Automate notebook deployment, job creation, and testing.

PYSPARK

1. Explain lazy evaluation in PySpark.

- **Lazy Evaluation** means that PySpark doesn't execute transformations immediately. Instead, it builds a logical execution plan and waits until an action (like collect() or count()) is called to execute the plan. This approach optimizes the execution by allowing PySpark to run transformations more efficiently.

---

2. Difference between groupByKey and reduceByKey.

- **groupByKey**: Groups all values for each key across the cluster, which can lead to high data transfer costs.
- **reduceByKey**: Combines values for each key locally before shuffling, reducing data transfer and improving performance.

---

3. What is the difference between repartition and coalesce?

- **repartition**: Increases or decreases the number of partitions and performs a full shuffle, making it more expensive.
- **coalesce**: Only decreases the number of partitions without a full shuffle, making it more efficient for reducing partitions.

---

4. Explain broadcast join in PySpark.

- A **broadcast join** is used when joining a large DataFrame with a smaller one. PySpark broadcasts the smaller DataFrame to all nodes, allowing each node to perform the join locally without shuffling the large DataFrame. This significantly improves performance for such join operations.

---

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

- You can register a UDF in PySpark using the udf function. Here's an example:

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

def my_function(value):
    return value.upper()

my_udf = udf(my_function, StringType())
spark.udf.register("my_udf", my_udf)
```

This UDF can now be used in DataFrame operations or SQL queries.

---

6. What is the purpose of SparkContext and SparkSession?

- **SparkContext**: The entry point for low-level Spark functionality, allowing access to Resilient Distributed Datasets (RDDs) and cluster resources.
- **SparkSession**: Introduced in Spark 2.0, it encapsulates SparkContext and provides a unified entry point for DataFrame and Dataset APIs, simplifying the user experience.

---

7. How does caching work in PySpark?

- Caching in PySpark stores the data in memory to speed up subsequent actions on the same data. You can cache a DataFrame using:
- df.cache()

This avoids recomputation and speeds up data retrieval.

---

8. Difference between wide and narrow transformations.

- **Narrow Transformations**: Operations where each partition's data is used by at most one partition in the parent RDD. Examples include map, filter, and union.
- **Wide Transformations**: Operations where data from multiple partitions may be combined, leading to shuffling. Examples include reduceByKey and groupByKey.

---

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

- You can create a rank column using the rank() function with a window specification:

```
from pyspark.sql.window import Window
from pyspark.sql.functions import rank

window_spec = Window.partitionBy("department").orderBy("salary")
df = df.withColumn("rank", rank().over(window_spec))
```

This assigns ranks within each department based on salary.

---

10. What is Z-ordering in Spark?

- **Z-ordering** is a technique used to colocate related information in the same set of files. It maps multidimensional data to one dimension while preserving locality of the data points. This is particularly useful in optimizing queries that involve filtering on multiple columns.

---

11. Explain Delta Lake and its benefits over Parquet.

- **Delta Lake** is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads. Benefits over Parquet include:
    - **ACID Transactions**: Ensures data integrity with serializable isolation.
    - **Scalable Metadata Handling**: Handles large metadata efficiently.
    - **Time Travel**: Query previous versions of the data.

- **Schema Enforcement and Evolution**: Prevents bad data from causing data corruption.

---

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

- **Adaptive Query Execution (AQE)** is a feature that allows Spark to optimize and adjust query plans based on runtime statistics. It can optimize skew joins, coalesce shuffle partitions, and switch join strategies dynamically.

---

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

- Without timestamps, one approach is to use **Delta Lake's** versioning capabilities to track changes. Alternatively, maintain a separate control table to track processed records using unique identifiers.

---

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

- In Spark:
  - **Job**: Triggered by an action; corresponds to a Spark action like collect().
  - **Stage**: A job is divided into stages based on shuffle boundaries.
  - **Task**: Each stage is divided into tasks, with one task per partition.

---

15. How to persist and cache data in PySpark?

- **cache()**: Stores the data in memory.
- **persist(storage_level)**: Allows specifying storage levels (e.g., memory, disk). Example:

```
df.persist(StorageLevel.MEMORY_AND_DISK)
```

This stores the DataFrame in memory and spills to disk if necessary.

Absolutely! Here's your **reordered Q&A list** with **answers** for each topic related to **PySpark and Azure Data Engineering**:

---

16. How do you design and implement data pipelines using Azure Data Factory (ADF)?

In ADF, data pipelines are designed using a visual interface or JSON code. Steps:

- Define **Linked Services** to connect source/target.
- Use **Datasets** to represent input/output data.
- Orchestrate tasks using **Activities** (e.g., Copy, Data Flow, Execute Pipeline).
- Use **Triggers** (schedule/event/tumbling) to execute pipelines.
- Implement **parameterization**, **control flow**, and **monitoring** for dynamic and robust pipelines. ADF supports data movement, transformation, and control flow in hybrid cloud setups.

---

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

Schema evolution in ADLS (especially with Delta Lake) can be handled using:

- **Schema enforcement** to prevent bad writes.
- **Schema evolution** to allow new columns during write operations using:

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

- Use tools like **Azure Data Flow** or **Databricks Autoloader** for ingesting data with evolving schemas.
- Maintain metadata-driven frameworks to validate incoming schema changes before applying them.

---

18. Describe the process of setting up and managing an Azure SQL Database.

Steps include:

1. **Create Azure SQL Server** and define credentials.
2. **Create a database** inside the server (choose DTU or vCore model).
3. **Configure firewall rules** to allow access.
4. Use **Azure Data Studio**, **SSMS**, or **ADF** to manage and query data.
5. Implement **monitoring** (via Azure Monitor), **backup**, **auditing**, and **auto-tuning**.
6. Apply **RBAC** and **Key Vault** for secure connection strings.

---

19. How do you optimize data storage and retrieval in Azure Data Lake Storage (ADLS)?

To optimize ADLS:

- Store files in **columnar formats** like **Parquet/Delta**.
- Use **partitioning** and **Z-ordering** to accelerate query performance.
- Implement **data lifecycle policies** to manage old/stale data.
- Optimize file sizes (not too many small files).
- Enable **Hierarchical Namespace** for file/folder operations.
- Use **caching** and **Databricks** for efficient retrieval.

---

20. Explain the use of Azure Synapse Analytics and how it integrates with other Azure services.

Azure Synapse is an analytics service that unifies:

- **Data warehousing (Dedicated SQL pools)**
- **Big data processing (Spark, Serverless SQL pools)** It integrates with:
- **Azure Data Lake (ADLS)** for storage
- **Azure Data Factory** for pipelines
- **Power BI** for visualization
- **Azure Purview** for governance
- Supports **on-demand querying**, **Spark notebooks**, and **Synapse pipelines**.

---

21. Difference between Spark SQL and PySpark DataFrame APIs.

- **Spark SQL**: Uses SQL queries over DataFrames. Good for SQL developers. Example:

```sql
SELECT department, SUM(salary) FROM employee GROUP BY department;
```

- **PySpark DataFrame API**: Python-based functional API. Example:

```python
df.groupBy("department").agg(sum("salary").alias("total_salary"))
```

- Both are **interoperable**, but DataFrame API is preferred for type safety and better debugging.

---

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

- **Drop nulls**:

```python
df.dropna()  # drops rows with any nulls
df.dropna(subset=["column1"])  # drops rows with null in specific column
```

- **Fill nulls**:

```python
df.fillna({"column1": "default", "column2": 0})
```

- **Replace nulls with mean/median** using pandas_udf or approxQuantile.

---

23. DataFrame API syntax to filter and aggregate data.

To filter and aggregate using PySpark:

```
from pyspark.sql.functions import sum as _sum

df.filter(df.department == "Sales") \
  .groupBy("department") \
  .agg(_sum("salary").alias("sumofsalary"))
```

This filters rows by department and then groups by department to calculate total salary.

---

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

Answer: Azure Stream Analytics allows real-time data ingestion and processing from sources like IoT Hub, Event Hubs, or Blob Storage. You write SQL-like queries to process streaming data and send the results to various outputs such as Azure SQL DB, Data Lake, or Power BI. It's ideal for real-time dashboards, anomaly detection, and telemetry processing.

---

25. Describe the process of integrating Azure Databricks with Azure Data Lake Storage.

Answer: Integration involves:

- Mounting ADLS Gen2 to Databricks using OAuth and a service principal.
- Configuring permissions via Azure RBAC or ACLs.

- Using dbutils.fs.mount() or ABFS path to read/write data.

---

26. Write a PySpark code to: read a CSV, join two DataFrames, perform aggregation, and filter rows.

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

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

df1 = spark.read.csv("path/file1.csv", header=True, inferSchema=True)
df2 = spark.read.csv("path/file2.csv", header=True, inferSchema=True)

# Join
joined_df = df1.join(df2, df1["id"] == df2["emp_id"])

# Aggregation
agg_df = joined_df.groupBy("department").agg(sum("salary").alias("total_salary"))

# Filter
filtered_df = agg_df.filter(agg_df["total_salary"] > 50000)
filtered_df.show()
```

---

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

Answer: Azure Key Vault securely stores secrets, keys, and certificates. It integrates with services like ADF, Synapse, and Databricks to manage access using Managed Identity. This prevents hardcoding credentials in code.

---

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

Answer: Managed Identity provides Azure services with an automatically managed identity in Azure AD to access other

resources securely. For instance, ADF can access Azure SQL or Data Lake using its system-assigned identity.

---

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

Answer:

- **Data at rest:** Use encryption via Azure Storage Service Encryption, TDE for SQL DB.
- **Data in transit:** Enforce HTTPS, use VPNs, and TLS for secure connections.

---

30. How do you ensure data consistency and reliability in distributed systems on Azure?

Answer: Strategies include:

- Idempotent operations
- Retry policies
- Exactly-once semantics in Stream Analytics
- Using Delta Lake with Databricks for ACID guarantees
- Monitoring with Azure Monitor

---

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

Answer: Use ADF's **Monitoring Tab** for pipeline run history, alerts, and logs. You can enable diagnostic logging to Azure Monitor or Log Analytics for deeper troubleshooting and performance metrics.

---

32. How do you manage and automate data workflows using Azure Logic Apps?

Answer: Logic Apps provide visual, code-free workflows to integrate services (e.g., trigger on new file in Blob, send email, start ADF pipeline). Logic Apps can call REST APIs or trigger Databricks jobs.

---

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

Answer: Azure HDInsight is a managed Apache big data platform (Hadoop, Spark, Hive). You use it to run large-scale processing using distributed compute, ideal for ETL jobs, machine learning, and log processing.

---

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

Answer:

- Use Geo-redundant storage (GRS)
- Automated backups for Azure SQL
- Synapse Geo-backups
- Replication and Data Sync
- ADF failover pipelines

---

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

Answer: PolyBase allows querying external data in Blob Storage, ADLS, or Hadoop directly from SQL queries in Synapse (SQL DW) without importing data. Ideal for ELT and big data processing.

---

36. What are the different data partitioning strategies in Azure SQL Data Warehouse?

Answer: Partitioning includes:

- **Hash partitioning** (common for distribution)
- **Range partitioning** (used in partitioned tables for date ranges)
  It helps in improving query performance and parallelism.

---

37. How do you use Azure Monitor to track and analyze performance metrics of your data infrastructure?

Answer: Azure Monitor collects metrics/logs from ADF, Databricks, SQL DB, etc. You can set up alerts, visualize metrics in dashboards, and analyze performance trends or failures.

---

38. Describe the process of performing ETL operations using Azure Data Factory.

Answer:

1. Use Linked Services to connect sources and sinks.
2. Create Datasets.
3. Build a pipeline with activities like Copy, Data Flow, and Stored Procedure.
4. Schedule or trigger pipelines.
5. Monitor execution and performance.

---

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

Answer:

- Use Azure Blob lifecycle policies to auto-tier data.
- Archive infrequently used data to cool/archive tiers.
- Retain logs using Azure Monitor settings.
- Define clear retention policies for compliance.

DATA LAKE STORAGE (ADLS)

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

- You can integrate ADLS Gen2 with Azure Databricks by mounting the storage using OAuth2 with a service principal or managed identity. Alternatively, use direct ABFS paths in your code.
- Example using ABFS:

```
df = spark.read.format("parquet").load("abfss://<container>@<account
    >.dfs.core.windows.net/<path>")
```

2. Why is mounting preferred over using access keys to connect Databricks with ADLS?

- Mounting provides a more secure and manageable method by using credentials stored in Azure Key Vault.
- Access keys are less secure and harder to rotate.

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

- Azure AD-based RBAC

- POSIX-style Access Control Lists (ACLs)
- Firewall and virtual network rules
- Private endpoint support
- Encryption at rest and in transit

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

- **At rest**: Microsoft-managed keys or customer-managed keys (CMK) via Key Vault.
- **In transit**: TLS/SSL encryption for secure communication.

5. How do you handle schema evolution in ADLS?

- Use Delta Lake for automatic schema evolution.
- Enable schema evolution options in Spark reads/writes.
- Maintain metadata via Azure Purview or Synapse.

6. Explain how to copy all files with different formats (CSV, Parquet, Excel) from a source to target.

- Use ADF's Copy Activity with wildcard paths.
- Create a parameterized dataset or use data flows for format-specific logic.
- Use metadata-driven pipeline for dynamic behavior.

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

- Enable additionalColumns in the Copy Activity source to include file names.
- Use dynamic content: @dataset().SourceFileName

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

- Use tools like Azure Data Factory, Dataflows, or Azure Databricks.
- Enable partitioning and parallelism.
- Use Event Grid for event-driven ingestion.

9. How do you monitor and troubleshoot issues in ADLS?

- Use Azure Monitor and Log Analytics.
- Enable diagnostic settings to track operations.
- Integrate with Azure Purview for lineage.

10. What are the differences between ADLS Gen1 and Gen2?

Feature	ADLS Gen1	ADLS Gen2
Storage backend	Proprietary	Azure Blob Storage
Hierarchical namespace	Yes	Yes
Security model	ACL only	ACL + RBAC
Integration	Limited	Extensive (Synapse, Databricks, etc.)

11. Explain the use of hierarchical namespaces in ADLS.

- Enables folder and file-level access control.
- Supports atomic rename and move operations.
- Improves performance with directory structure.

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

- Use blob lifecycle management rules to transition data between tiers (hot, cool, archive).

- Define rules for auto-delete or move based on creation or last modified dates.

13. How do you implement versioning in ADLS?

- Enable blob versioning at the storage account level.
- Access previous versions using APIs or Azure Storage Explorer.

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

- Use ADF Data Flows or Databricks to implement validation logic.
- Maintain metadata rules and run quality checks post-load.

15. Describe the process of setting up and managing access control lists (ACLs) in ADLS.

- Use Azure Storage Explorer or CLI:

```
az storage fs access set --acl "user::rwx,group::r--,other::---" \
    --path /mydir --filesystem myfs --account-name mystorage
```

16. How do you integrate ADLS with Azure Synapse Analytics?

- Use Linked Services to connect Synapse to ADLS.
- Query data using serverless SQL pool (OPENROWSET) or dedicated pool with external tables.

17. Explain the process of auditing and logging access to data in ADLS.

- Enable diagnostic logs for storage account.
- Log categories: read, write, delete operations.
- Export logs to Log Analytics or Event Hub.

18. How to optimize data retrieval from ADLS in Spark?

- Use columnar formats (Parquet, Delta).
- Apply partition pruning and predicate pushdown.

- Avoid small files.
- Cache frequently accessed data.

19. How do you manage and optimize storage costs in ADLS?

- Store infrequently accessed data in cool/archive tiers.
- Use lifecycle policies for auto-deletion.
- Monitor usage via Azure Cost Management.

20. Explain the process of setting up disaster recovery for ADLS.

- Use geo-redundant storage (GRS) or read-access GRS.
- Set up failover regions and data replication.
- Implement backup and recovery strategy via Data Factory or third-party tools.

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

Feature	Blob Storage	ADLS Gen2
Hierarchical namespace	No	Yes
Analytics optimized	No	Yes
ACL support	Limited	Full support
File system semantics	No	Yes

DATA MODELING AND ARCHITECTURE

1. **Difference between a database, data warehouse, and data lake**

- **Database**: Stores structured data for day-to-day transactions (OLTP).
- **Data Warehouse**: Stores structured, historical data for analytical processing (OLAP).
- **Data Lake**: Stores raw, structured, semi-structured, and unstructured data at scale.

2. **What is data mart, and how does it differ from a data warehouse?**

- **Data Mart**: A subset of a data warehouse tailored for a specific department or business unit.
- **Data Warehouse**: A central repository for all organizational data from various domains.

3. **What are fact and dimension tables in data modeling?**

- **Fact Table**: Contains measurable, quantitative data (e.g., sales amount).
- **Dimension Table**: Contains descriptive attributes (e.g., customer, product).

4. **How do you design a star schema for a sales reporting system?**

- **Star Schema**: Central fact table (e.g., Sales) linked to surrounding dimension tables (e.g., Customer, Product, Time, Store).

5. **Explain the use of surrogate keys in dimensional modeling**

- **Surrogate Keys**: Unique identifiers (often integers) used in dimension tables instead of natural business keys for consistency and tracking changes.

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

- **Denormalization**: Combining normalized tables to reduce joins and improve query performance; used in OLAP systems.

7. **How do you handle slowly changing dimensions (SCD) in data modeling?**

- **SCD Type 1**: Overwrite old data.
- **SCD Type 2**: Add new records with versioning.
- **SCD Type 3**: Track limited historical data in same record.

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

- **Normalization**: Organizing data to reduce redundancy and improve integrity. Forms include 1NF, 2NF, 3NF, etc.

9. **Difference between ER modeling and dimensional modeling**

- **ER Modeling**: Used for transactional systems, emphasizes entities and relationships.
- **Dimensional Modeling**: Used for analytical systems, emphasizes fact and dimension tables.

10. **What are the different types of data schemas, and how do you choose the right one for your data model?**

- **Star Schema**: Simple, used for quick reporting.
- **Snowflake Schema**: Normalized dimensions, better for complex hierarchies.
- **Galaxy Schema**: Multiple fact tables, used for complex data marts.

- **Choice Depends On**: Query complexity, performance needs, business requirements.

11. **Build a simple data warehouse for a grocery store**

- **Fact Table**: Sales (SaleID, ProductID, CustomerID, StoreID, DateID, Quantity, TotalAmount)
- **Dimension Tables**: Product, Customer, Store, Date

12. **Describe the role of metadata in data modeling and data architecture**

- **Metadata**: Data about data. Helps in governance, lineage, auditing, cataloging, and easier maintenance.

13. **How do you implement a data governance framework in a data lake environment?**

- Use **Azure Purview** for cataloging and lineage.
- Define **data ownership and stewardship roles**.
- Enforce **access control**, **policies**, and **auditing**.

14. **Explain the deployment architecture of a data pipeline involving ADF, Databricks, and ADLS**

- **ADF**: Orchestrates pipelines, triggers Databricks notebooks.
- **Databricks**: Performs data transformation using PySpark.
- **ADLS**: Serves as storage for raw, curated, and processed data.
- Flow: Source > ADF > Databricks > ADLS > Synapse (Optional for reporting)

15. **How do you ensure data consistency and integrity in a distributed data architecture?**

- Use **transactional support** (e.g., Delta Lake ACID).
- Implement **data quality checks** and **validation rules**.

- Leverage **versioning and checkpoints**.

16. **How do you optimize data models for performance and scalability?**

- Use **partitioning**, **indexing**, and **denormalization**.
- Monitor and **analyze query plans**.
- Design for **read-heavy or write-heavy** loads depending on usage.

OTHER QUESTIONS

1. What is DAG in Spark, and how does it work?

A Directed Acyclic Graph (DAG) in Spark represents a sequence of computations performed on data. Each node is an RDD, and the edges represent operations. Spark builds the DAG during job submission and uses it for scheduling tasks across the cluster.

2. Explain Spark architecture (Driver vs Worker).

Spark has a master-slave architecture. The Driver coordinates the execution of tasks, maintains the DAG, and manages metadata. Workers (executors) run the actual computations and store data.

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

- Narrow: Data required to compute the records in a single partition resides in the same partition (e.g., map, filter).
- Wide: Requires shuffling data across partitions (e.g., reduceByKey, groupByKey).

4. How does fault tolerance work in ADF and Databricks?

- ADF: Retry policies, fail activity options, and try-catch logic.
- Databricks: DAG lineage and Delta Lake's ACID support allow recovery from failures.

5. What challenges have you faced in managing large datasets (e.g., 3TB+ files)?

- Long processing times
- Shuffling issues
- Optimizing storage format (Parquet/Delta)
- Managing memory and partitioning

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

- Use Azure DevOps pipelines
- For ADF: ARM templates and Git integration
- For Databricks: Repos, notebooks as code, and workspace CLI or REST API

7. How to improve the performance of a Spark job?

- Broadcast joins
- Caching/persisting intermediate results
- Partitioning strategies
- Avoiding wide transformations when possible

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

Delta Lake is a storage layer that brings ACID compliance to Spark by maintaining transaction logs, supporting schema enforcement, and allowing time travel.

9. What is a Unity Catalog in Databricks?

A unified governance layer for all data assets in Databricks. It provides centralized access control, auditing, and data lineage.

10. Explain the role of integration runtimes in hybrid scenarios (on-prem to cloud).

Integration Runtimes (IR) in ADF enable secure data movement. Self-hosted IR is used for on-prem sources, while Azure IR handles cloud-native movement.

11. How to design an end-to-end data pipeline architecture?

Use ADF for orchestration, Databricks for transformation, ADLS for storage, and Azure SQL/Synapse for serving. Ensure modular, scalable, and fault-tolerant design.

12. Snowflake-specific: How does it compare to Delta Lake or other data lakes?

- Snowflake: Fully managed, supports SQL, automatic scaling, and time travel.
- Delta Lake: Open-source, tightly integrated with Spark, ACID transactions.

13. When would you choose Snowflake over other platforms?

When ease of use, scaling, and seamless SQL experience with minimal ops overhead is prioritized.

14. Difference between Azure Logic Apps, Azure Functions, and ADF.

- Logic Apps: Workflow automation
- Functions: Serverless code execution
- ADF: Data integration and ETL

15. Azure Synapse Analytics: How does it compare to Databricks?

- Synapse: SQL-centric, good for BI and reporting
- Databricks: ML/AI-focused, Spark-native, highly performant

16. How to use Azure DevOps for CI/CD pipelines.

Use pipelines (YAML/classic) to automate build, test, and deploy. Integrate Git repos, ARM templates, and Databricks CLI.

17. What are the key features of Azure DevOps?

- Boards
- Repos
- Pipelines
- Test Plans
- Artifacts

18. How do you implement CI and CD in Azure DevOps?

- CI: Automate code builds and testing
- CD: Deploy automatically to target environments

19. Explain the role of pipelines in Azure DevOps.

Pipelines automate build and deployment. They support tasks, agents, environments, and gates.

20. How do you manage dependencies and versioning in Azure DevOps?

Use artifacts, semantic versioning, and build variables to manage dependencies.

21. What are Azure Functions, and how do they differ from traditional web services?

Azure Functions are serverless, event-driven compute services. Unlike web services, they scale automatically and only run when triggered.

22. How do you deploy and manage Azure Functions?

Deploy using Visual Studio, Azure CLI, GitHub Actions, or DevOps. Manage through the Azure portal or CLI.

23. Use cases for Azure Functions in data engineering.

- Triggering ADF pipelines
- Processing queue messages
- Lightweight transformations

24. How do you secure secrets and keys using Azure Key Vault?
Store secrets in Key Vault and reference them in ADF, Functions, and pipelines using managed identities.

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

Use linked services or configurations to fetch secrets during runtime.

26. What are the benefits of using Azure Synapse for big data processing?

- Serverless and dedicated SQL pools
- Deep integration with Power BI and Spark
- Data lake integration

27. How does Azure Synapse integrate with other Azure services?

- With ADLS, ADF, Power BI, and Azure ML through linked services and Synapse Studio.

28. Difference between dedicated and serverless SQL pool in Synapse.

- Dedicated: Fixed resources, predictable performance
- Serverless: Pay-per-query, flexible

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

Use SQL insights, query plans, and workload management.

30. Describe the process of data ingestion in Synapse.

Use COPY statement, ADF pipelines, or Synapse pipelines to ingest data.

31. Best practices for data partitioning and distribution in Synapse.

- Use hash-distribution for large joins
- Replicated tables for small lookup tables

32. How do you implement security and compliance in Synapse?

- Role-based access control
- Auditing
- Data masking and encryption

33. Concept of data integration and transformation in ADF.

Use data flows, mapping activities, and custom transformations in pipelines.

34. Error handling and retry mechanisms in ADF.

- Use error rows, custom logging, retry policies, and fail activity handlers.

35. Role of triggers and schedules in ADF.

- Schedule: Time-based triggers
- Tumbling/Custom event: For windowed or data-based triggering

36. How do you use Azure Monitor to analyze performance in Azure services?

Azure Monitor collects logs and metrics. Integrate with Log Analytics and Application Insights for visualization and alerts.