VISION BOARD

TIGER ANALYTICS DATA ENGINEER INTERVIEW QUESTIONS AND ANSWERS

DEVIKRISHNA R

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

SET 1

Python Questions

SQL Questions

Data Engineering Questions

PYTHON

1. Explain how decorators function in Python.

Answer:

Decorators in Python are functions that modify the behaviour of another function or method. They are applied using the @decorator_name syntax above the target function.
A decorator wraps the original function, often adding pre- or post-processing steps without changing the function's core logic.

Example:

```
def decorator(func):
    def wrapper():
        print("Before the function call")
        func()
        print("After the function call")
    return wrapper

@decorator
def say_hello():
    print("Hello!")

say_hello()
```

```
Before the function call
Hello!
After the function call
```

2. Reverse a string programmatically without using slicing.

Answer:

We can reverse a string using a loop or Python's built-in reversed() function.

Method 1: Using a loop

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

print(reverse_string("hello"))  # Output: "olleh"
```

```
olleh
```

Method 2: Using reversed()

```
def reverse_string(s):
    return ''.join(reversed(s))

print(reverse_string("hello"))  # Output: "olleh"
```

```
olleh
```

3. How do you manage missing values in a dataset using Python?

Answer:

Managing missing data involves various techniques depending on the context:

1. **Identify missing values:**
   Use isnull() or isna() from pandas.

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, None], 'B': [4, None, 6]})
print(df.isnull())  # Shows True for missing values
```

2. Remove missing values:

```
df.dropna(inplace=True)  # Removes rows with NaN
```

3. Fill missing values:

```
df.fillna(0, inplace=True)  # Replace NaN with 0
```

4. Interpolate missing values:

```
df.interpolate(method='linear', inplace=True)
```

4.Write a Python program to find the second largest number in a list.

Answer:

Method:

```
def second_largest(numbers):
    unique_numbers = list(set(numbers))  # Remove duplicates
    if len(unique_numbers) < 2:
        return None  # Not enough unique numbers
    unique_numbers.sort(reverse=True)  # Sort in descending order
    return unique_numbers[1]  # Return second largest

nums = [10, 20, 20, 4, 5, 99, 50]
print(second_largest(nums))
```

```
50
```

5. Generate an email ID from first and last names in Python.

Answer:

Here's a simple implementation:

```
def generate_email(first_name, last_name, domain="example.com"):
    first = first_name.lower().strip()
    last = last_name.lower().strip()
    return f"{first}.{last}@{domain}"

email = generate_email("John", "Doe")
print(email)
```

```
john.doe@example.com
```

SQL QUESTIONS:

1. How do you optimize SQL queries?

Answer:

Optimizing SQL queries improves performance and reduces execution time. Techniques include:

1. **Indexing:** Add indexes to columns used in WHERE, JOIN, and ORDER BY.
2. **Avoid SELECT *:** Query only the required columns.
3. **Use Joins Efficiently:** Prefer INNER JOIN over subqueries when possible.
4. **Analyze Query Plans:** Use EXPLAIN to identify bottlenecks.
5. **Limit Results:** Use LIMIT or TOP for pagination or reduced output.
6. **Avoid Wildcards:** Use specific conditions to narrow down results.
7. **Normalize Data:** Minimize redundancy but balance with denormalization if joins slow performance.
8. **Caching Results:** Use caching for frequently accessed data.

2. Write a SQL query to find the average salary of employees in each department.

Answer:

```sql
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id;
```

3. Explain the difference between a Primary Key, a Natural Key, and a Surrogate Key.

Answer:

1. **Primary Key:**
    - **Uniquely identifies each record in a table.**
    - **Enforces uniqueness and cannot contain NULL.**
2. **Natural Key:**
    - **A key derived from existing data attributes that naturally identify records.**
    - **Example: A social security number in a user table.**
3. **Surrogate Key:**
    - **A system-generated unique identifier, typically an integer or UUID.**
    - **Example: id column with auto-increment.**

4. Write a SQL query to delete the 10th highest salary from an employee table.

Answer:

```sql
DELETE FROM employees
WHERE salary = (
    SELECT DISTINCT salary
    FROM employees
    ORDER BY salary DESC
    LIMIT 1 OFFSET 9
);
```

5. Write a SQL query to retrieve unique records without using DISTINCT or GROUP BY.

Answer:

```sql
SELECT column1, column2
FROM employees e1
WHERE NOT EXISTS (
    SELECT 1
    FROM employees e2
    WHERE e1.column1 = e2.column1
    AND e1.column2 = e2.column2
    AND e1.id < e2.id
);
```

6. Given two tables, find the count of records using INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.

Answer:

Assuming two tables, table1 and table2:

- **INNER JOIN:**
  Matches records common in both tables.

```sql
SELECT COUNT(*)
FROM table1
INNER JOIN table2
ON table1.id = table2.id;
```

- **LEFT JOIN:**
  Returns all records from table1 and matches from table2.

```
SELECT COUNT(*)
FROM table1
LEFT JOIN table2
ON table1.id = table2.id;
```

- **RIGHT JOIN**:

  Returns all records from table2 and matches from table1.

```
SELECT COUNT(*)
FROM table1
RIGHT JOIN table2
ON table1.id = table2.id;
```

- **FULL OUTER JOIN:**
  Combines LEFT JOIN and RIGHT JOIN.

```
SELECT COUNT(*)
FROM table1
FULL OUTER JOIN table2
ON table1.id = table2.id;
```

7. Explain how to handle NULL values in JOIN operations in SQL.

Answer:

Handling NULL values in joins involves using specific strategies:

1. **Using Conditional Logic:**
   Use IS NULL or IS NOT NULL to include/exclude NULL values.

```sql
SELECT *
FROM table1
LEFT JOIN table2
ON table1.id = table2.id
WHERE table2.id IS NULL;
```

2. **Coalesce Values:**
   Replace NULL with default values using COALESCE().

```sql
SELECT COALESCE(table1.id, 0) AS id
FROM table1
LEFT JOIN table2
ON table1.id = table2.id;
```

3. **Ensure Proper Join Logic:**
   Avoid assumptions; explicitly handle NULL in columns that allow it.

8. How would you find the top 3 highest-paid employees in an organization?

```sql
SELECT *
FROM employees
ORDER BY salary DESC
LIMIT 3;
```

DATA ENGINEERING QUESTIONS:

1. What is a PCollection in the context of Data Engineering?

Answer:

A **PCollection** in Apache Beam is an immutable, distributed data set that represents data for parallel processing.

- **Bounded PCollection:** Finite size, typically used for batch processing.
- **Unbounded PCollection:** Infinite data, suited for streaming pipelines.
  It supports transformations like ParDo, GroupByKey, and Flatten for scalable processing.

2. What is the source of your data in a typical ETL pipeline?

Answer:

Common sources of data in an ETL pipeline include:

1. **Databases:** Relational (MySQL, PostgreSQL) or NoSQL (MongoDB, Cassandra).
2. **APIs:** REST or GraphQL endpoints.
3. **Files:** CSV, JSON, Parquet, or Avro from file systems like HDFS, AWS S3, or Azure Blob.
4. **Streaming Services:** Kafka, Kinesis, or Pub/Sub.
5. **Data Lakes/Warehouses:** Snowflake, BigQuery, or Redshift.

3. What do you understand by partitioning and clustering in a database context?

Answer:

- **Partitioning:** Splits data into smaller chunks for storage and query efficiency, based on a column or range (e.g., date or region).
- **Clustering:** Organizes data within partitions based on sorting keys, improving scan efficiency for queries.

4. What is the difference between ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform)?

Answer:

- **ETL:** Transformations occur before loading data into the target system. Suitable for traditional data warehouses.
- **ELT:** Data is loaded into the target system first, and transformations occur there. Ideal for cloud-based data lakes.

5. What is incremental loading in Delta Tables, and how would you implement it?

Answer:

Incremental loading updates only the new or changed records in Delta Tables.

Implementation:

1. Identify new or changed data using timestamps or a unique key.
2. Use a **merge** operation to upsert records:

```
MERGE INTO delta_table AS target
USING source_table AS source
ON target.id = source.id
WHEN MATCHED THEN UPDATE SET *
WHEN NOT MATCHED THEN INSERT *;
```

6. How do you handle data skew in distributed systems like Spark?

Answer:

1. **Key Salting:** Add random prefixes to keys to distribute data evenly.
2. **Increase Partitions:** Use .repartition() to balance partitions.
3. **Broadcast Joins:** Use for small datasets to avoid shuffles.
4. **Pre-Aggregation:** Reduce the volume of data before shuffles.

7. What is a Broadcast Join in Spark? Can you provide a scenario where it is beneficial?

Answer:

A **Broadcast Join** sends a small dataset to all nodes, allowing efficient joining with larger datasets.
Scenario: Joining a small lookup table (e.g., country codes) with a massive dataset (e.g., user logs).

8. How would you optimize Spark jobs and improve performance in a data pipeline?

Answer:

- Use **repartition** or **coalesce** to adjust partition size.
- Employ **broadcast joins** for small tables.
- Filter early to reduce data processed.
- Cache reusable datasets.
- Use **Parquet** or **ORC** for efficient storage.

9. What is the difference between a DataFrame and an RDD in Spark?

Answer:

- **DataFrame:** High-level abstraction with schema, optimized queries via Catalyst.
- **RDD:** Low-level abstraction, no schema, requires more manual optimization.

10. Can you explain the architecture of Spark and how it processes data in parallel?

Answer:

- **Driver Program:** Orchestrates job execution.
- **Cluster Manager:** Allocates resources (e.g., YARN, Kubernetes).
- **Executors:** Perform tasks on worker nodes.
  Spark splits jobs into stages and tasks, executing tasks in parallel across partitions.

11. What is the role of data vacuuming in Delta Lake?

Answer:

Vacuuming removes unused files (e.g., old versions, deleted data) to save storage and maintain performance.

VACUUM delta_table RETAIN 168 HOURS;

12. How would you implement pipeline error handling in Azure Data Factory (ADF)?

Answer:

1. **OnFailure Triggers:** Define actions for failed activities.
2. **Logging:** Log errors to a storage account or database.
3. **Alerts:** Use Azure Monitor or Logic Apps for notifications.
4. **Retry Policies:** Configure retries for transient issues.

13. How do you handle memory-related issues in Spark when working with large datasets?

Answer:

1. **Adjust Memory Allocation:** Increase executor and driver memory (spark.executor.memory).
2. **Use Disk-Based Storage:** Persist datasets with DISK_ONLY mode.
3. **Avoid Collecting Large Data:** Minimize use of .collect().
4. **Optimize Partitions:** Balance partition sizes.

14. What strategies would you use to debug a failed pipeline in Azure Data Factory?

Answer:

1. **Monitor Logs:** Review logs in the ADF Monitoring tab.
2. **Retry Activities:** Use re-run functionality to isolate issues.
3. **Check Linked Services:** Verify credentials and connectivity.
4. **Enable Debug Mode:** Test pipeline steps interactively.

SET2

1. Explain the differences between batch processing and real-time data processing.

Answer:

1. **Batch Processing:** Processes data in chunks, suited for historical analysis. Example: Generating a monthly sales report.
2. **Real-Time Processing:** Processes data continuously as it streams in, suitable for real-time applications like fraud detection.

---

2. Write a query to find the second-highest salary of an employee.

Answer:

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

3.What is the role of SparkContext in Spark?

Answer:

- SparkContext is the entry point for any Spark application.
- It communicates with the cluster manager and manages RDD creation, accumulators, and broadcast variables.

---

4. How do you monitor and debug data pipeline jobs in Spark?

Answer:

- **Spark UI:** Analyze job stages, tasks, and executor logs.
- **Driver Logs:** Debug errors and monitor the progress.
- **Third-party tools:** Use Ganglia or Prometheus for performance metrics.

---

5. What strategies do you use to handle skewed data in Spark jobs?

Answer:

1. **Key Salting:** Add random prefixes to keys.
2. **Broadcast Joins:** Avoid shuffles for small datasets.

3. **Pre-Aggregation:** Reduce data before shuffling.

---

6. Explain how Spark works internally when a job is submitted.

Answer:

1. The **Driver** converts the job into a Directed Acyclic Graph (DAG).
2. The **DAG Scheduler** divides the DAG into stages.
3. Tasks are assigned to executors, and computations are distributed across nodes.

---

7. What is the importance of partitioning in Spark, and how do you decide the partitioning strategy for a job?

Answer:

Partitioning ensures balanced workloads across nodes.
Strategy:

- Choose high-cardinality keys for even distribution.
- Adjust partition count (repartition or coalesce) based on cluster resources.

---

8. How do you handle missing or corrupted data in large datasets?

Answer:

- **Imputation:** Replace missing values with mean, median, or mode.
- **Deletion:** Drop rows or columns with significant missing data.
- **Custom Handling:** Use domain-specific rules for replacement.

---

9. Write a query to list customers who spent more than ₹1000 on their orders in the last month.

Answer:

```
SELECT customer_id, SUM(order_amount) AS total_spent
FROM orders
WHERE order_date >= DATEADD(MONTH, -1, GETDATE())
GROUP BY customer_id
HAVING SUM(order_amount) > 1000;
```

---

10. What is the difference between RANK(), DENSE_RANK(), and ROW_NUMBER()?

Answer:

- **RANK():** Assigns rank with gaps for ties.
- **DENSE_RANK():** Assigns consecutive ranks without gaps.
- **ROW_NUMBER():** Assigns unique numbers, even for ties.

---

11. Write a query to join two DataFrames with different schemas where the left table has more rows.

Answer:

```
df1.join(df2, df1['key'] == df2['key'], 'left')
```

Ensure you handle null values for mismatched rows.

---

12. How do you design and implement a data pipeline for incremental data processing?

Answer:

1. **Track Changes:** Use timestamps or version numbers.
2. **Extract Incremental Data:** Pull only new or modified records.
3. **Merge:** Use MERGE or upserts for updates.

---

13. Explain how you handled a specific challenge in your previous project, particularly related to large-scale data processing.

Answer:

- **Challenge:** Managing data skew during joins.
- **Solution:**
    1. Used key salting to distribute records evenly.
    2. Pre-aggregated data before joins to reduce shuffle size.

---

14. What are the key differences between Spark and Hadoop?

Answer:

- **Spark:** In-memory computation, faster for iterative tasks, user-friendly APIs.
- **Hadoop:** Disk-based, better for large-scale batch processing.

---

15. How do you ensure data quality and consistency in a distributed system?

Answer:

- Validate schemas, remove duplicates, enforce constraints, and implement lineage tracking.

---

16. Explain how Spark and Hadoop handle large datasets.

Answer:

- **Spark:** Uses in-memory storage and distributed processing for speed.
- **Hadoop:** Relies on HDFS for disk-based processing.

---

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

Answer:

- Adjust partition size, cache intermediate results, filter early, and use efficient formats like Parquet.

---

18. Write a query to find customers who spent the most on their orders in the last month.

Answer:

```sql
SELECT customer_id, SUM(order_amount) AS total_spent
FROM orders
WHERE order_date >= DATEADD(MONTH, -1, GETDATE())
GROUP BY customer_id
ORDER BY total_spent DESC
LIMIT 1;
```

---

19. Explain the use of window functions in SQL and how they were applied in your project.

Answer:

Window functions compute results across a set of rows.
Example in Projects: Ranking employees by performance using RANK().

---

20. How do you ensure the scalability of your data processing system?

Answer:

- Use distributed systems, optimize partitioning, and employ auto-scaling clusters.

---

21. Describe the process of data ingestion in a data pipeline using Spark.

Answer:

1. Load data from sources like Kafka, HDFS, or databases.
2. Apply transformations (filtering, joins).
3. Write the processed data to sinks (data lakes, warehouses).

---

22. What is the role of a DataFrame in Spark, and how is it different from an RDD?

Answer:

- **DataFrame:** Optimized, with schema, supports SQL-like operations.
- **RDD:** Low-level, no schema, requires manual optimizations.

---

23. How do you implement pipeline error handling in Spark?

Answer:

- Use exception handling, enable retries, log errors, and track metrics.

---

24. How do you implement a running total using a window function in Spark?

Answer:

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

window_spec = Window.partitionBy("group_column").orderBy("order_column")
df.withColumn("running_total", sum("value").over(window_spec))
```

---

25. Explain the differences between ETL and ELT.

Answer:

- **ETL:** Transformations occur before loading data.
- **ELT:** Load first, transform in the target system.

---

26. What is the difference between INNER JOIN, LEFT JOIN, and RIGHT JOIN?

Answer:

- **INNER JOIN:** Matches rows in both tables.
- **LEFT JOIN:** All rows from the left table, with nulls for unmatched right rows.
- **RIGHT JOIN:** All rows from the right table, with nulls for unmatched left rows.
