Q1. The entry point in Spark-Core application is:

Options:

1. Driver

2. SparkContext

3. Executor

4. Spark

Answer: SparkContext

Explanation: SparkContext is the main entry point for any Spark-Core application. It allows the application to access the cluster and perform computations. Without it, Spark jobs cannot be initiated.

---

Q2. SparkSQL translates commands into codes. These codes are processed by:

Options:

1. Driver Nodes

2. Executor Nodes

3. Cluster Manager

4. Spark Master

Answer: Executor Nodes

Explanation: The codes generated by SparkSQL commands are processed by Executor Nodes. Executors handle the execution of tasks on the cluster and return the results back to the Driver.

---

Q3. If using a textFile() function of a Spark Context, the number of partitions are equal to:

Options:

1. Number of files on HDFS

2. Number of executors

3. Number of blocks of the file on HDFS

4. None of the above

Answer: Number of blocks of the file on HDFS

Explanation: By default, the textFile() function partitions the data based on the number of HDFS blocks. Each HDFS block is treated as a partition.

---

Q4. Launching Spark application on YARN cluster with --deploy-mode as cluster will launch the driver…

Options:

1. On worker nodes on YARN cluster side

2. Locally as an external client

3. Locally as an embedded client

4. None of the above

Answer: On worker nodes on YARN cluster side

Explanation: In cluster mode, the driver is launched on the worker nodes within the YARN cluster itself, rather than locally on the client machine.

---

Q5. In Spark, which file format do you prefer to achieve predicate pushdown?

Options:

1. CSV

2. Parquet

3. Avro

4. Text

Answer: Parquet

Explanation: Parquet supports columnar storage and predicate pushdown, enabling faster filtering of data by reducing the number of rows read during query processing.

---

Q6. Which of the following is good for low-level transformations and actions?

Options:

1. RDD

2. DataFrame

3. Dataset

4. SparkSQL

Answer: RDD

Explanation: RDD (Resilient Distributed Dataset) provides low-level APIs for transformations and actions, making it ideal for users requiring fine-grained control over their computations.

---

Q7. Which of the following is NOT a benefit of using Spark for handling large volumes of data?

Options:

1. Spark offers high performance

2. Spark can only handle thousands of records

3. Spark allows orderly data flows

4. Spark can use strictly defined schemas while ingesting data

Answer: Spark can only handle thousands of records

Explanation: Spark is designed to handle massive volumes of data, far beyond thousands of records. This makes option 2 incorrect.

---

Q8. Why would Spark use immutable data frames to begin with?

Options:

1. To add complexity to your Spark tasks

2. To efficiently handle data throughout the cluster

3. To easily modify variable values as needed

4. To conserve storage space

Answer: To efficiently handle data throughout the cluster

Explanation: Immutability ensures consistency, fault tolerance, and reliable operations in distributed environments. It simplifies data handling across Spark clusters.

---

Q9. Consider a DataFrame called users\_df with columns as ID, Name, Age, and State. If we wanted to return only the Name and State fields for any ID greater than 3000, which code snippet meets these requirements?

Options:

1. users\_df.filter('ID > 3000').select("Name", "State")

2. users\_df.select("Name", "State").filter('ID > 3000')

3. users\_df.select("Name", "State").where('ID > 3000')

4. None of the above

Answer: users\_df.filter('ID > 3000').select("Name", "State")

Explanation: Both filter and where can be used to filter data, but the question specifically checks for a code snippet using select and where.

—

Q10. When creating a new user-defined function, which is not a possible value for the second argument?

Options:

1. ArrayType(IntegerType())

2. IntegerType()

3. LongType()

4. udf()

Answer: udf()

Explanation: The second argument of a user-defined function (UDF) in Spark defines the return type. udf() is not a valid data type, whereas ArrayType(IntegerType()), IntegerType(), and LongType() are valid return types.

---

Q11. State True or False: In broadcast join, a smaller DataFrame gets shared across all worker nodes.

Options:

1. TRUE

2. FALSE

Answer: TRUE

Explanation: A broadcast join is used to broadcast a smaller DataFrame to all worker nodes, ensuring efficient join operations with larger DataFrames.

---

Q12. Consider the following Data Frame called users\_df, If you wanted to return only the entries without nulls, which of following options would not work? ID Name Age State

140 George L 47 lowa

3260 Mary R 34 Vermont

18502 null 68 Ohio

999 Rick W 23 California

Options:

1. users\_df = users\_df.filter(users\_df.Name.isNotNull())

2. users\_df = users\_df.where(users\_df.ID == 18502)

3. users\_df = users\_df.where(~ (users\_df.ID == 18502))

4. users\_df = users\_df.filter(~ col('Name').isNull())

Answer: users\_df = users\_df.where(users\_df.ID == 18502)

Explanation: This code only filters rows where the ID is equal to 18502, but it does not handle null values. The other options effectively filter out null entries.

---

Q13. Which of the following is an advantage of Spark DataFrames over RDDs?

Options:

1. Operations using DataFrames are automatically optimized

2. They are smaller

3. They can perform more kinds of operations

4. They can hold more kinds of data

Answer: Operations using DataFrames are automatically optimized

Explanation: Spark DataFrames benefit from Catalyst Optimizer, which automatically optimizes queries, making DataFrames more efficient compared to RDDs.

---

Q14. Which of the following queries returns a table of tail numbers and destinations for flights (this table flights has 16 columns) that lasted more than 10 hours? In the following queries, the column air\_time contains the duration of the flight in minutes.

Options:

1. SELECT dest, tail\_num FROM flights WHERE air\_time > 10;

2. SELECT dest, tail\_num FROM flights WHERE air\_time > 600;

3. SELECT \* FROM flights WHERE air\_time > 600;

4. None of the above

Answer: SELECT dest, tail\_num FROM flights WHERE air\_time > 600;

Explanation: 10 hours equals 600 minutes. The correct query filters flights with air\_time > 600 and selects only the dest and tail\_num columns.

---

Q15. Which of the following steps are involved in creating a Spark Structured Streaming application?

Options:

1. Read

2. Transform

3. Sink

4. All of the above

Answer: All of the above

Explanation: Creating a Spark Structured Streaming application involves reading the data stream, transforming it as required, and sinking (writing) the final output to a target destination.

---

Q16. In Spark Streaming, which output mode will you use when you know that you are not going to update any previous output, and each batch will write new records only?

Options:

1. Append

2. Update

3. Complete

4. None of the above

Answer: Append

Explanation:

Append Mode: This mode is designed specifically for scenarios where each batch adds new data to the existing output. It's the most efficient choice when there are no updates or deletions to previous records.

Update Mode: This mode is used when you need to update existing records based on the current batch's data. It's more complex and resource-intensive than Append mode.

Complete Mode: This mode is used when you want to completely replace the previous output with the results of the current batch. It's typically used for aggregations or other operations that produce a single, final result.

---

Q17. State True or False: Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks.

Options:

1. TRUE

2. FALSE

Answer: TRUE

Explanation: Broadcast variables are used in Spark to cache a read-only variable on all machines, reducing the overhead of data transfer during execution.

---

Q18. FlatMap transforms an RDD of length N into another RDD of length M. Which of the following is true for N and M?

Options:

1. N > M

2. N < M

3. N ≤ M

4. None of the above

Answer: N ≤ M

Explanation: FlatMap can produce zero or more elements for each input element. Therefore, the length of the resulting RDD (M) can be equal to or greater than the original RDD (N).

---

Q19. A team member of your project described the following error scenario:

After giving the command:

spark-submit --master yarn --deploy-mode cluster --driver-memory 2g --num-executors 2 --executor-memory 1g data\_pipeline.py

I am getting the error message: "module ingest not found". The script data\_pipeline.py depends on modules ingest.py, transform.py, and persist.py.

Options:

1. Dependent modules and the main Python script (data\_pipeline.py) are not in the same directory.

2. The --py-files option is missing in the command to list all dependent modules.

3. Need to submit the job in client mode only.

4. Need to submit the job in local mode only.

Answer: The --py-files option is missing in the command to list all dependent modules.

Explanation: The --py-files option allows the user to specify additional Python files required by the main script. Omitting this option causes dependency-related errors.

---

Q20. State True or False: In Kafka, a topic can be created without a partition.

Options:

1. TRUE

2. FALSE

Answer: FALSE

Explanation: A Kafka topic must have at least one partition. Partitions are fundamental to Kafka's scalability and parallelism.

---

Q21. Kafka fulfills the promise of fault tolerance by:

Options:

1. Providing multiple brokers

2. Increasing the number of partitions

3. The replication factor for the topic partitions

4. None of the above

Answer: The replication factor for the topic partitions

Explanation: Kafka achieves fault tolerance by replicating partitions across multiple brokers. The replication factor ensures that if one broker fails, the data remains available on other brokers.

---

Q22. Which file format will you use in Spark if you want to achieve predicate pushdown at the time of query processing?

Options:

1. Text

2. Avro

3. Parquet

4. None of the above

Answer: Parquet

Explanation: Parquet supports columnar storage and allows predicate pushdown, which improves query performance by filtering data at the storage level.

---

Q23. Given the following exo\_2\_1\_df DataFrame, how many records will the solution\_2\_1\_df DataFrame contain?

Input:

exo\_2\_1\_df.show()

# +-------------------+

# | numbers |

# +-------------------+

# | [1, 2, 3, 4, 5] |

# | [5, 6, 7, 8, 9, 10]|

# +-------------------+

solution\_2\_1\_df = exo\_2\_1\_df.select(explode(col("numbers")))

Options:

1. 2

2. 5

3. 11

4. 6

Answer: 11

Explanation: The explode() function flattens the array in the numbers column into individual rows. The first array has 5 elements, and the second has 6, resulting in 11 rows in total.

---

Q24. Out of the 3 lines of code below, which line will fail?

book.printSchema()

# root

# |-- value: string (nullable = true)

from pyspark.sql.functions import col

book.select(book.value) # line 1

book.select(book["value"]) # line 2

book.select(col("value")) # line 3

Options:

1. Line 1

2. Line 2

3. Line 3

4. None of the above

Answer: None of the above

Explanation: None

---

Q25. The entry point in Spark SQL/DataFrame applications is:

Options:

1. Driver

2. SparkContext

3. SparkSession

4. Executor

Answer: SparkSession

Explanation: SparkSession is the entry point for Spark SQL and DataFrame applications. It combines functionality of the older SQLContext and HiveContext.

---

Q26. Which of the following are the common features of RDD and DataFrame?

Options:

1. Immutability

2. Lazy Evaluation

3. Fault Tolerance

4. All the above

Answer: All the above

Explanation: Both RDDs and DataFrames share features like immutability, lazy evaluation, and fault tolerance, which are core to Spark's distributed processing capabilities.

---

Q27. Which of the following is NOT true for DataFrame?

Options:

1. DataFrame in Apache Spark is behind RDD

2. We can build DataFrame from different data sources. structured data file, tables in Hive

3. The Application Programming Interface (APIs) of DataFrame is available in various languages

4. Both in Scala and Java, we represent DataFrame as Dataset of rows.

Answer: 1. DataFrame in Apache Spark is behind RDD

Explanation:

\* DataFrames are built on top of RDDs.

\* RDDs (Resilient Distributed Datasets) are the fundamental building blocks in Spark.

\* DataFrames provide a higher-level abstraction over RDDs, offering a more user-friendly and efficient way to work with structured and semi-structured data.

Key Points about DataFrames in Spark:

\* Built on RDDs: DataFrames are essentially optimized collections of RDDs organized into named columns.

\* Data Source Agnostic: They can be created from various sources like CSV, JSON, Hive tables, etc.

\* Immutable: Like RDDs, DataFrames are immutable. Transformations create new DataFrames.

\* Optimized Execution: Spark optimizes DataFrame operations through a process called Catalyst, leading to significant performance improvements.

\* Language Support: APIs are available in various languages like Scala, Java, Python, and R..

---

Q28. Which of the following is the fundamental data structure in PySpark SQL?

Options:

1. RDD

2. DataFrame

3. Dataset

4. None of the above

Answer: DataFrame

Explanation: DataFrames are the primary data structure in PySpark SQL, offering a schema-based view of distributed data.

---

Q29. What is an action in Spark RDD?

Options:

1. The way to send results from executors to the driver

2. Takes RDD as input and produces one or more RDDs as output

3. Creates one or many new RDDs

4. All the above

Answer: The way to send results from executors to the driver

Explanation: Actions trigger computation and return results from executors to the driver, such as collect() or count().

---

Q30. After giving the following command:

spark-submit --master yarn --deploy-mode cluster --driver-memory 2g --num-executors 2 --executor-memory 1g data\_pipeline.py

How can you see the logs of this application?

Options:

1. You can see the logs on the console

2. Use the YARN logs utility

3. Use the Spark logs utility

4. None of the above

Answer: Use the YARN logs utility

Explanation: In cluster mode on YARN, application logs can be retrieved using the YARN logs utility (yarn logs -applicationId <app\_id>).

---