# Cloudera

# CDP-3002

## CDP Data Engineer- Certification Exam



## For More Information – Visit link below:

## https://www.examsboost.com/

## Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

# Latest Version: 6.2

## Question: 1

You are deploying a Spark application in a Kubernetes environment. Your application is designed to process large datasets using Spark's data frame API. You have created a Docker image for your Spark application. Which of the following 'kubectl* commands should you use to deploy your
Spark application onto the Kubernetes cluster?

A. *kubectl apply -f spark-app.yamr
B. 'kubectl create deployment my-spark-app --image=my-spark-app-image'
C. *kubectl config set-context —current —namespace=my-spark-app'
D. 'kubectl expose deployment my-spark-app --type=LoadBalancer —port=808ff

**Answer: A**

Explanation:
To deploy a Spark application in Kubernetes, you should use a YAML configuration file that defines the SparkApplication resource. The correct command to apply this configuration is 'kubectl apply -f spark-app.yamr , as it will create or update resources in the cluster based on the YAML file.

## Question: 2

As a data engineer, you are working with PySpark to analyze data stored in an HDFS cluster. You need to read a CSV file into a Spark DataFrame. Which of the following code snippets correctly reads the data from HDFS into a DataFrame?

A.
```
`df = spark.read.csv("s3://path/to/data.csv")`
```
B.
```
`df = spark.read.format("csv").load("hdfs:///path/to/data.csv")`
```
C.
```
`df = spark.read.text("local:///path/to/data.txt")`
```
D.
```
`df = spark.read.parquet("hdfs:///path/to/data.parquet")`
```

**Answer: B**

Explanation:
The correct way to read a CSV file from HDFS into a Spark DataFrame in PySpark is using 'spark.read.format("csv").load("hdfs:///path/to/data.csv")*. This method specifies the format as CSV and loads the file from the specified HDFS path.

## Question: 3

Your team is using Spark on Kubernetes, and you've been tasked with improving the efficiency of resource utilization. You decide to enable dynamic allocation of executors based on workload. Which of the following PySpark session configurations correctly enables dynamic allocation?

A.
```
`spark = SparkSession.builder.appName("EfficientApp").config("spark.executor.instances", "10").getOrCreate()`
```
B.
```
`spark = SparkSession.builder.appName("EfficientApp").config("spark.dynamicAllocation.enabled", "true").getOrCreate()`
```
C.
```
`spark = SparkSession.builder.appName("EfficientApp").config("spark.kubernetes.container.image", "my-spark-image").getOrCreate()`
```
D.
```
`spark = SparkSession.builder.appName("EfficientApp").config("spark.executor.cores", "4").getOrCreate()`
```

### Answer: B

Explanation:
To enable dynamic allocation of executors in Spark, the configuration 'spark.dynamicAllocation.enabled' must be set to 'true'. This allows Spark to dynamically adjust the number of executors based on the workload.

## Question: 4

You are working on a project that involves processing large datasets stored in HDFS. You need to read a CSV file into a DataFrame using PySpark. Which of the following code snippets correctly achieves this?

A.
```
`df = spark.read.csv("file:///path/to/file.csv", header=True)`
```
B.
```
`df = spark.read.format("csv").load("hdfs://namenode:8020/path/to/file.csv")`
```
C.
```
`df = spark.read.csv("hdfs://namenode:8020/path/to/file.csv", header=True, inferSchema=True)`
```
D.
```
`df = spark.read.text("hdfs://namenode:8020/path/to/file.csv")`
```

### Answer: C

Explanation:
Option C is correct as it properly specifies the HDFS path and includes options for headers and schema inference, which are common requirements when reading CSV files into DataFrames in P S ark.

A data engineer needs to query a table stored in Apache Hive using SparkSQL. Which of the following commands correctly retrieves data from a Hive table named 'sales data'?

A.
`SELECT * FROM sales_data`
B.
`hiveCtx.sql("SELECT * FROM sales_data")`
C.
`spark.sql("SELECT * FROM sales_data")`
D.
`spark.read.table("sales_data")`

**Answer: C**

Explanation:
Option C is the correct choice because it uses the 'spark.sql' method to execute a SQL query on a Hive table.

## Question: 6

Your team is integrating PySpark with a MySQL database. You need to read data from a table named 'employees'. Which of the following PySpark code snippets correctly accomplishes this task?

A.
`df = spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/dbname").option("dbtable", "employees").option("user", "root").load()`
B.
`df = spark.read.jdbc("jdbc:mysql://localhost:3306/dbname", "employees", properties={"user": "root"})`
C.
`df = spark.read.format("mysql").option("url", "jdbc:mysql://localhost:3306/dbname").option("dbtable", "employees").load()`
D.
`df = spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/dbname").option("tablename", "employees").load()`

**Answer: A**

Explanation:
Option A is correct because it properly uses the JDBC format with all the necessary options including the URL, database table, and user credentials.

## Question: 7

A team is planning to use PySpark to read data from an Apache Cassandra database. Which of the following options correctly demonstrates how to load data from a Cassandra table named in the keyspace 'sales'?

A.
`df = spark.read.format("cassandra").option("keyspace", "sales").option("table", "customer_data").load()`
B.
`df = spark.read.format("org.apache.spark.sql.cassandra").options(table="customer_data", keyspace="sales").load()`
C.
`df = spark.read.cassandra("sales", "customer_data")`
D.
`df = spark.read.format("cassandra").load("sales.customer_data")`

**Answer: B**

Explanation:
Option B is correct as it uses the specific format for Cassandra ('org.apache.spark.sql.cassandra') and correctly specifies the keyspace and table options.

## Question: 8

You need to process data stored in AWS S3 using SparkSQL. Which of the following options correctly reads a JSON file stored in S3 into a DataFrame and performs a SQL query on it?

A.
`df = spark.read.json("s3a://bucket-name/path/to/file.json"); spark.sql("SELECT * FROM df")`
B.
`df = spark.read.format("json").load("s3://bucket-name/path/to/file.json"); df.createOrReplaceTempView("data"); spark.sql("SELECT * FROM data")`
C.
`df = spark.read.json("s3://bucket-name/path/to/file.json"); df.createOrReplaceTempView("data"); spark.sql("SELECT * FROM data")`
D.
`df = spark.read.json("s3a://bucket-name/path/to/file.json"); df.createOrReplaceTempView("data"); spark.sql("SELECT * FROM data")`

**Answer: D**

Explanation:
Option D is correct as it uses the 's3a' protocol to read the JSON file from AWS S3, creates a temporary view, and then performs a SQL query on the DataFrame.

## Question: 9

Your project involves integrating Spark with a NoSQL database, MongoDB. You need to write a DataFrame 'df into a MongoDB collection named 'orders'. Which PySpark code snippet correctly achieves this?

A.
`df.write.format("mongo").option("uri", "mongodb://localhost:27017/dbname.orders").save()`
B.
`df.write.format("com.mongodb.spark.sql").option("uri", "mongodb://localhost:27017/dbname.orders").save()`
C.
. `df.write.format("org.apache.spark.sql.mongo").option("uri", "mongodb://localhost:27017/dbname.orders").save()`
D.
. `df.write.mongo("mongodb://localhost:27017/dbname.orders")`

**Answer: B**

Explanation:
Option B is correct as it uses the official MongoDB Spark connector C com.mongodb.spark.sql') and specifies the URI correctly, pointing to the MongoDB collection.

## Question: 10

You are working with a large dataset in PySpark and notice that certain operations are being executed repeatedly, leading to performance issues. Which of the following approaches should you adopt to optimize these operations?

A. Use the method.
B. Increase the number of partitions using 'df.repartition()'.
C. Write the data to a CSV file and then read it back.
D. Use the 'df.persist()' method.

**Answer: D**

Explanation:
The 'df.persist()' method is used in PySpark to store the intermediate computation of a DataFrame in memory, so when it is accessed again, it does not need to be recomputed, thus optimizing performance. 'df.cache()' is a synonym but 'persist()' offers more control over storage level.

## Question: 11

A data analyst is performing a join operation in PySpark between a large DataFrame Clarge_df) and a much smaller DataFrame ('small_df). The process is very slow. What can the analyst do to optimize this join operation?

A. Use "key").

B. Apply before the join.
C. Utilize the 'broadcast' function in the join operation.
D. Increase the number of partitions in 'small_df'.

**Answer: C**

Explanation:
The 'broadcast' function is used in PySpark to optimize join operations where one DataFrame is significantly smaller than the other. It broadcasts the smaller DataFrame across all nodes so that it is available in memory, reducing the data shuffling over the network.

## Question: 12

After running a PySpark job, you want to analyze the performance and identify potential bottlenecks. Which tool should you use for this purpose?

A. PySpark SQL CLI.
B. Spark Web I-Jl.
C. PySpark DataFrame API.
D. Hadoop YARN ResourceManager.

**Answer: B**

Explanation:
The Spark Web IJI is the primary tool for monitoring Spark jobs. It provides detailed information about the job execution, stages, tasks, and resource utilization, which is crucial for identifying performance bottlenecks and optimizing Spark jobs.

# Thank You for Trying Our Product

**For More Information –** <span style="color:red">**Visit link below:**</span>

## https://www.examsboost.com/

**15 USD Discount Coupon Code:**

## G74JA8UF

# FEATURES

- ✓ **90 Days Free Updates**

- ✓ **Money Back Pass Guarantee**

- ✓ **Instant Download or Email Attachment**

- ✓ **24/7 Live Chat Support**

- ✓ **PDF file could be used at any Platform**

- ✓ **50,000 Happy Customer**