ChatGPT

Session 1: Apache Spark APIs

1. **Which class is the primary entry point for programming with Spark (version 2.0 and later)?**
2. A. `SparkContext`
3. B. `SparkSession`
4. C. `SQLContext`
5. D. `HadoopContext`
   Correct Option: B
   Explanation: Spark's official documentation states that SparkSession is "the entry point into all functionality in Spark" for Spark 2.x and later [1]. (SparkContext is the older entry point before Spark 2.0.)

6. **Which of the following best describes an RDD in Spark?**

7. A. A mutable dataset stored on a single node.
8. B. A list of machine learning models.
9. C. A Resilient Distributed Dataset: an immutable, partitioned collection that can recover from node failures.
10. D. A resource manager for cluster scheduling.
    Correct Option: C
    Explanation: An RDD (Resilient Distributed Dataset) is Spark's fundamental data abstraction. It is partitioned across the cluster and "can be persisted. Datasets composed of multiple partitions may still be operated on in parallel… and they automatically recover from node failures" [2]. This provides fault tolerance and distributed processing.

11. **In Spark, which term refers to operations that return a value to the driver program (rather than transforming an RDD)?**

12. A. Transformation
13. B. Action
14. C. Cache
15. D. Shuffle
    Correct Option: B
    Explanation: Spark RDDs support transformations (which produce new RDDs) and actions (which compute and return a result to the driver). The Spark documentation defines transformations as creating new datasets and actions as returning values to the driver program [3]. Thus operations like `count()` or `collect()` are actions.

16. **What does it mean that Spark's transformations are "lazy"?**

17. A. Transformations execute immediately when declared.

18. B. Transformations are optimized at compile time.
19. C. Transformations are recorded and only executed when an action is called.

20. D. Transformations cache data by default.
    Correct Option: C
    Explanation: Spark records transformations on RDDs in a lineage graph and only computes them when an action is invoked. As noted in Spark's docs, "Transformations are not executed until an action requires a result" – this lazy evaluation enables optimizations [4]. Thus, Spark defers computation until needed by an action.

21. **What is the effect of calling `persist()` (or `cache()`) on an RDD in Spark?**

22. A. It immediately writes the RDD to disk.
23. B. It schedules all transformations for immediate execution.
24. C. It stores the RDD in memory on the cluster for faster reuse.
25. D. It deletes the RDD data.
    Correct Option: C
    Explanation: Calling `persist()` (or `cache()`) tells Spark to keep the RDD's data in memory (or storage) across operations, so that it can be reused efficiently. The Spark RDD programming guide states that `persist`/`cache` will keep the RDD in memory for faster future access [5].

Session 2: Apache Spark APIs (Key-Value and Advanced RDD Operations)

1. **Which Spark operation will trigger a "shuffle" (data re-distribution across the cluster)?**
2. A. `map`
3. B. `filter`
4. C. `reduceByKey`

5. D. `flatMap`
   Correct Option: C
   Explanation: Operations like `reduceByKey`, `groupByKey`, and joins cause data to be shuffled between partitions. As the Spark docs explain, "Operations which can cause a shuffle include… reduceByKey, groupByKey, join, cogroup" [6]. In contrast, `map` and `filter` are local transformations that do not trigger a shuffle.

6. **When aggregating values by key in Spark, which operation is generally preferred for better performance?**

7. A. `groupByKey`
8. B. `mapValues`
9. C. `reduceByKey`

10. D. `sortByKey`
    Correct Option: C
    Explanation: To aggregate values with the same key, `reduceByKey` is typically more efficient than `groupByKey`. The Spark documentation notes that "if you are grouping in order to perform an

aggregation (such as a sum), using `reduceByKey` will yield much better performance" because it performs map-side combining before the shuffle [7].

11. **Which type of shared variable in Spark is used to send a read-only copy of data to all worker nodes?**

12. A. Accumulator
13. B. Broadcast variable
14. C. Driver variable

15. D. Checkpoint
Correct Option: B
Explanation: Spark provides broadcast variables to efficiently distribute a large read-only value to all worker nodes. The Spark programming guide describes broadcast variables as values cached on each node, which avoids sending this data with every task [8]. Accumulators are write-only shared counters, so broadcasting is the correct mechanism for read-only data.

16. **Which of the following transformations does *not* cause a shuffle when applied to an RDD?**

17. A. `join`
18. B. `groupByKey`
19. C. `map`

20. D. `cogroup`
Correct Option: C
Explanation: The `map` transformation is executed independently on each partition and does not require redistributing data, so it does not cause a shuffle. By contrast, operations like `join`, `groupByKey`, and `cogroup` involve moving data across partitions. The Spark docs list operations that cause shuffles and `map` is notably not among them [6].

21. **What is an "accumulator" in Spark typically used for?**

22. A. Caching data across tasks
23. B. Broadcasting a variable to all nodes
24. C. Counting or summing values across tasks (a write-only counter)
25. D. Serializing functions sent to workers
Correct Option: C
Explanation: Accumulators in Spark are shared variables that tasks can add to, often used as counters or sums. They only support an "add" operation, not reads by tasks. The Spark guide states that accumulators are "variables that are only 'added' to through an associative operation and can therefore be efficiently supported in parallel. They can be used to implement counters or sums" [8].

Session 3: EDA using PySpark

1. **Which PySpark command reads a CSV file into a DataFrame?**
2. A. `sc.textFile("file.csv")`
3. B. `spark.read.csv("file.csv")`

4. C. `spark.read.text("file.csv")`
5. D. `spark.load("file.csv")`
   Correct Option: B
   Explanation: In Spark SQL, the DataFrameReader API is used to read structured data. The documentation shows that to read a CSV file you use `spark.read().csv("file_name")`. The `spark.read.csv` method creates a DataFrame from a CSV source [9]. (Option A returns an RDD of text lines, not a DataFrame.)

6. **In PySpark, which method returns summary statistics (count, mean, stddev, min, max) for each DataFrame column?**

7. A. `df.describe()`
8. B. `df.summary()`
9. C. `df.stats()`
10. D. `df.explain()`
    Correct Option: A
    Explanation: The `describe()` method on a DataFrame computes basic statistics for numeric and string columns, including count, mean, standard deviation, min, and max. According to Spark's documentation, `DataFrame.describe()` "computes basic statistics (count, mean, stddev, min, max) for numeric and string columns" [10].

11. **What does `df.dropna()` do on a PySpark DataFrame?**

12. A. Drops columns containing null values.
13. B. Drops rows containing null values.
14. C. Replaces null values with zeros.
15. D. No operation, returns original DataFrame.
    Correct Option: B
    Explanation: The `dropna()` method returns a new DataFrame that omits rows with any null values. The PySpark documentation states that `DataFrame.dropna()` will remove rows containing nulls [11]. (To replace nulls, one would use `fillna()`, not `dropna()`.)

16. **What does the PySpark DataFrame method `fillna(0)` accomplish?**

17. A. Fills all null values in the DataFrame with 0.
18. B. Deletes all rows containing null values.
19. C. Filters rows where values are 0.
20. D. Resets the DataFrame index to zero.
    Correct Option: A
    Explanation: The `fillna()` method replaces null values. Specifically, `df.fillna(0)` replaces all null (NaN) entries in the DataFrame with 0. The Spark API documentation describes `DataFrame.fillna()` as replacing null values (alias for `na.fill`) [12].

21. **How can you filter rows in a PySpark DataFrame based on a condition, for example `age > 30`?**

22. A. `df.select("age > 30")`
23. B. `df.where(df.age > 30)`
24. C. `df.filter(df.age > 30)`
25. D. Both B and C
    Correct Option: D
    Explanation: Both `filter()` and `where()` are equivalent methods in PySpark for filtering rows. The documentation notes that `DataFrame.filter(condition)` keeps rows satisfying the condition (and `where()` is an alias) [13]. Thus, `df.filter(df.age > 30)` or `df.where(df.age > 30)` correctly filters rows with `age > 30`.

Session 4: ETL Jobs using Spark

1. **Which PySpark command writes a DataFrame in Parquet format to storage?**
2. A. `df.write.parquet("path")`
3. B. `spark.save.parquet("path")`
4. C. `df.write.format("parquet").save("path")`

5. D. Both A and C
   Correct Option: D
   Explanation: To write a DataFrame as Parquet, one can use `df.write.parquet("path")` or equivalently `df.write.format("parquet").save("path")`. The Spark documentation's examples show using `dataframe.write().parquet("directory")` to save in Parquet format [14]. Both methods produce the same result.

6. **In Spark's DataFrameWriter, what does the write mode "overwrite" do?**

7. A. Appends new data to existing files.
8. B. Replaces any existing data at the path.
9. C. Throws an error if the destination exists.

10. D. Writes data without saving.
    Correct Option: B
    Explanation: The `mode("overwrite")` option tells Spark to overwrite existing data. As the documentation explains, the "overwrite" mode will overwrite existing data at the target path [15]. Other modes include "append", "ignore", and "errorifexists".

11. **Which write mode will cause Spark to ignore writing if the destination data already exists?**

12. A. `append`
13. B. `overwrite`
14. C. `ignore`

15. D. `error`
    Correct Option: C

Explanation: The `mode("ignore")` option causes Spark to skip the write if data already exists at the target. According to Spark's DataFrameWriter docs, in "ignore" mode Spark leaves the existing data intact and does nothing [15]. By contrast, "error" (or "default") would fail if data exists.

16. **What does the PySpark method `df.coalesce(1)` do?**

17. A. Splits the DataFrame into 1 partition without a full shuffle.
18. B. Repartitions the DataFrame into 1 partition with a shuffle.
19. C. Joins two DataFrames into one partition.

20. D. Caches the DataFrame.
    Correct Option: A
    Explanation: `coalesce(numPartitions)` returns a new DataFrame with exactly that many partitions, using a narrow (no shuffle) operation if reducing partitions. The docs state that `df.coalesce(n)` will result in a DataFrame with `n` partitions, and if it reduces partitions it avoids a full shuffle [16].

21. **Which PySpark command writes a DataFrame to CSV format at a given path?**

22. A. `df.save.csv("path")`
23. B. `df.write.csv("path")`
24. C. `spark.saveAsCsv("path")`
25. D. `df.csv("path")`
    Correct Option: B
    Explanation: To save a DataFrame as CSV, use `df.write.csv("path")`. Spark's documentation shows that `dataframe.write().csv("directory_name")` writes CSV files to the specified location [9]. Options A and D are not valid Spark APIs for writing.

Session 5: Kafka and Spark Streaming (Introduction)

1. **In Apache Kafka, what is a *topic*?**
2. A. A broker that stores messages.
3. B. A partition of data on disk.
4. C. A logical category or feed name to which messages are published.

5. D. A consumer group.
   Correct Option: C
   Explanation: A Kafka topic is a named feed or category of messages. Producers publish messages to topics, and consumers subscribe to topics to retrieve data. As described in Kafka documentation, "Topics are logical categories or streams of data" and form the abstraction to which data is published [17].

6. **What is a *partition* in Kafka?**

7. A. A physical machine storing data.
8. B. The basic unit of data storage and distribution within a topic.
9. C. A named category of messages.

10. D. A type of consumer offset.
    Correct Option: B
    Explanation: A partition is the unit of parallelism and storage in Kafka. Each topic is divided into one or more partitions. The Kafka docs explain that partitions are “the basic unit of data storage and distribution within Kafka topics” [18] . Each partition is an ordered, append-only sequence of messages.

11. **Why are Kafka topics divided into multiple partitions?**

12. A. To provide multiple copies of each message for redundancy.
13. B. To enable parallel consumption and higher throughput.
14. C. To encrypt data across different disks.

15. D. To separate producers from consumers.
    Correct Option: B
    Explanation: Partitioning allows Kafka to distribute topic data across multiple servers and enables multiple consumers to read in parallel. As Kafka documentation notes, dividing a topic into partitions “allows it to be spread over different servers (brokers) and consumers can read from partitions in parallel,” providing scalability and load balancing [19] .

16. **What is the role of a Kafka *consumer*?**

17. A. To publish messages to a topic.
18. B. To execute Spark streaming jobs.
19. C. To subscribe to topics and retrieve/process data from partitions.

20. D. To store data for long-term archival.
    Correct Option: C
    Explanation: A Kafka consumer subscribes to one or more topics and reads messages from them. According to Kafka docs, “Consumers subscribe to Kafka topics to retrieve and process data…They read messages from one or more partitions” [18] . Thus, consumers fetch data produced to topics by producers.

21. **In Spark Structured Streaming, how do you specify Kafka as a data source?**

22. A. Use `spark.kafka("...")`
23. B. Use `spark.readStream.format("kafka")` with appropriate options.
24. C. Use `KafkaContext` in Spark Streaming.
25. D. Spark Structured Streaming cannot read from Kafka.
    Correct Option: B
    Explanation: Spark Structured Streaming can read from Kafka by using the Kafka format in the DataFrame reader. For example, one uses `spark.readStream.format("kafka").option("kafka.bootstrap.servers", ...).option("subscribe", "topic").load()` as shown in the Spark documentation [20] . This sets up a streaming DataFrame that ingests from Kafka.

Session 6: Spark and Kafka Integration

1. **When writing data from Spark Structured Streaming to Kafka, what must the output DataFrame contain?**
2. A. A column named `topic`.
3. B. A column named `value`.
4. C. A column named `key`.
5. D. No special column names are required.
   Correct Option: B
   Explanation: To write to a Kafka topic, the DataFrame must have at least a `value` column containing the message payload (and optionally a `key` and other metadata columns). Spark's Kafka integration docs show that the DataFrame should have a `value` column (of type string or binary) which is required [21]. The `key` column is optional.
6. **Which Spark method writes the output of a streaming DataFrame to Kafka?**
7. A. `writeStream.format("kafka")`
8. B. `readStream.format("kafka")`
9. C. `write.format("kafka")`
10. D. `saveToKafka()`
    Correct Option: A
    Explanation: To send streaming output to Kafka, Spark uses the `writeStream` API with `format("kafka")`. For example: `df.writeStream.format("kafka").option("topic", ...).option("kafka.bootstrap.servers", ...).st` [22]. This sets up a Kafka sink for the streaming DataFrame.
11. **In Kafka Connect, what is the role of a *source connector*?**
12. A. It writes data from Kafka to an external system.
13. B. It reads data from an external source and writes it into Kafka.
14. C. It transforms Kafka data streams within the cluster.
15. D. It manages Kafka broker configurations.
    Correct Option: B
    Explanation: Kafka Connect source connectors import data from external systems into Kafka. The documentation explains that a source connector "extracts data from source systems (e.g., a database, or HDFS) and feeds it into Kafka" [23]. In contrast, a sink connector would export data from Kafka to an external target.
16. **What is the purpose of a Kafka Connect *sink connector*?**
17. A. To buffer data in Kafka.
18. B. To write data from Kafka topics to an external system.
19. C. To stream data between Kafka and Hadoop.

20. D. To monitor Kafka cluster health.
    Correct Option: B
    Explanation: A sink connector in Kafka Connect takes data from Kafka topics and writes it to external systems (like databases, HDFS, etc.). The documentation states that source connectors do the opposite and produce to Kafka, while sink connectors "take generic data from Kafka and write it to the desired target system using the destination system API" [23] .

21. **What message delivery semantics does Spark use when writing to Kafka?**

22. A. Exactly-once with no duplicates.
23. B. At-least-once, possibly with duplicates.
24. C. At-most-once.
25. D. Manual commit only.
    Correct Option: B
    Explanation: Spark's Kafka sink guarantees at-least-once semantics, meaning some records might be written more than once. The Spark documentation notes: "Kafka only supports at least once write semantics (some records may be duplicated)" [24] . Therefore, Spark jobs to Kafka must handle potential duplicates.

Session 7: Machine Learning using Spark MLlib

1. **Which Spark MLlib algorithm is commonly used for clustering data into a fixed number of clusters?**
2. A. Logistic Regression
3. B. KMeans
4. C. Decision Tree

5. D. Naive Bayes
   Correct Option: B
   Explanation: K-means is a popular clustering algorithm implemented in Spark MLlib that partitions data into *k* clusters. The Spark MLlib documentation explicitly includes KMeans under the clustering algorithms and notes it groups data into a pre-defined number of clusters [25] .

6. **Which Spark MLlib algorithm is used for model-based collaborative filtering (recommendation systems)?**

7. A. Support Vector Machine (SVM)
8. B. Alternating Least Squares (ALS)
9. C. Random Forest

10. D. K-Means
    Correct Option: B
    Explanation: Spark MLlib provides the ALS (Alternating Least Squares) algorithm for collaborative filtering and recommendation. The Spark MLlib guide states that ALS is used to learn latent factors in collaborative filtering (e.g., for recommendations) [26] .

11. **In Spark MLlib, what is a** Pipeline **used for?**

12. A. Managing data sources in Spark SQL.
13. B. Chaining together multiple data processing and machine learning steps.
14. C. Scheduling Spark jobs on the cluster.

15. D. Visualizing machine learning models.
    Correct Option: B
    Explanation: A Spark ML `Pipeline` chains multiple Transformers and Estimators into a single workflow. This allows sequential application of data preparation (like feature extraction) and learning algorithms. The Spark ML documentation describes Pipelines as a way to "chain multiple Transformers and Estimators together in order to specify an ML workflow" [27] .

16. **Which Spark MLlib algorithm is used for mining frequent itemsets from transactional data?**

17. A. FP-Growth
18. B. Linear Regression
19. C. OneHotEncoder

20. D. PageRank
    Correct Option: A
    Explanation: FP-Growth is Spark MLlib's algorithm for frequent pattern mining. It finds frequent itemsets in transactional data, useful for association rules. The documentation on Spark's frequent pattern library notes that FP-growth (implemented as PFP in Spark) is used to mine frequent itemsets [28] .

21. **What does Spark MLlib's Multilayer Perceptron Classifier (MLPC) implement?**

22. A. A decision tree-based model.
23. B. A feedforward neural network (multi-layer perceptron) for classification.
24. C. A clustering model.
25. D. A linear regression model.
    Correct Option: B
    Explanation: The `MultilayerPerceptronClassifier` in Spark MLlib implements a feedforward neural network (multi-layer perceptron) for classification tasks. As shown in Spark's examples, MLPC is explicitly imported for neural network classification [29] . It uses backpropagation and activation functions to classify input vectors.

Session 8: Deep Learning using Spark

1. **What is Apache BigDL in the context of Spark?**
2. A. A storage system for big data.
3. B. A distributed deep learning library that runs on Spark.
4. C. A Spark extension for SQL queries.

5. D. A project for graph processing in Spark.
   Correct Option: B
   Explanation: BigDL is a library for running deep learning on Spark. It allows developers to write deep learning applications (in Scala or Python) that run on Spark clusters. The documentation

describes BigDL as “a distributed deep learning library for Spark” that lets you “write deep learning applications as Scala or Python programs” on Spark [30] .

6. **Which programming languages can you use to write BigDL deep learning programs on Spark?**

7. A. Java or C++
8. B. Scala or Python
9. C. R or MATLAB

10. D. Fortran or COBOL
    Correct Option: B
    Explanation: BigDL supports both Scala and Python for writing deep learning programs on Spark. According to BigDL documentation, you can “write deep learning applications as Scala or Python programs” which will run distributedly on the Spark cluster [30] .

11. **What is the default activation function used in the output layer of Spark’s `MultilayerPerceptronClassifier`?**

12. A. ReLU
13. B. Sigmoid
14. C. Softmax

15. D. Tanh
    Correct Option: C
    Explanation: In Spark’s Multilayer Perceptron implementation, the output layer uses the softmax function by default. The source code documentation notes that intermediate layers use sigmoid activation, while the output layer uses softmax [31] .

16. **Which optimization algorithm does Spark MLlib use by default to train the Multilayer Perceptron?**

17. A. SGD (Stochastic Gradient Descent)
18. B. Adam Optimizer
19. C. L-BFGS

20. D. RMSprop
    Correct Option: C
    Explanation: Spark MLlib’s MLP classifier uses backpropagation with logistic loss, and by default it uses the L-BFGS optimizer. The MLlib documentation states: “MLPC employs a backpropagation based training and uses logistic loss (for classification) and the L-BFGS algorithm as an optimizer” [32] .

21. **Which built-in Spark MLlib method provides a basic neural network classifier?**

22. A. `RandomForestClassifier`
23. B. `LinearRegression`
24. C. `MultilayerPerceptronClassifier`

25. D. `GaussianMixture`
    Correct Option: C
    Explanation: Spark MLlib's `MultilayerPerceptronClassifier` provides a neural network (MLP) for classification. This class is explicitly part of `pyspark.ml.classification` and is the way to train a feedforward neural network in Spark [29].

Session 9: Spark SQL

1. **In Spark, which object is the entry point for running SQL queries and creating DataFrames?**
2. A. `SQLContext`
3. B. `HiveContext`
4. C. `SparkSession`
5. D. `SparkContext`
   Correct Option: C
   Explanation: SparkSession is the unified entry point for Spark SQL. It encapsulates all the functionality (creating DataFrames, running SQL) that was spread across `SQLContext` and `HiveContext` in earlier Spark versions. As noted earlier, Spark's documentation confirms SparkSession is the main entry point for Spark functionality [1].

6. **Which method registers a DataFrame as a temporary SQL view so that it can be queried with SQL?**

7. A. `df.registerTempTable("view")`
8. B. `df.createGlobalTempView("view")`
9. C. `df.createOrReplaceTempView("view")`
10. D. `df.toDF("view")`
    Correct Option: C
    Explanation: In Spark SQL, you use `createOrReplaceTempView(viewName)` on a DataFrame to create a temporary view tied to the SparkSession. The PySpark API docs state: "Creates or replaces a local temporary view with this DataFrame" [33]. This view can then be queried via `spark.sql`.

11. **How do you execute an SQL SELECT query on a Spark temporary view named `people`?**

12. A. `spark.sql("SELECT * FROM people")`
13. B. `spark.read.sql("SELECT * FROM people")`
14. C. `df.select("people")`
15. D. `spark.execute("people")`
    Correct Option: A
    Explanation: The `spark.sql()` method runs SQL queries on registered views or global temporary tables. For example, after creating a view `people`, one can do `spark.sql("SELECT * FROM people")` to get a DataFrame of results [34].

16. **What is the lifetime of a Spark SQL temporary view created via `createOrReplaceTempView`?**

17. A. It is permanent until manually dropped.
18. B. It lasts only for the duration of the SparkSession that created it.
19. C. It is automatically global across all sessions.
20. D. It is stored on disk for reuse between Spark jobs.
    Correct Option: B
    Explanation: A local temporary view's scope is tied to the SparkSession that created it. The documentation explains: "The lifetime of this temporary view is tied to the SparkSession that was used to create this DataFrame" [35]. Once the session ends, the view is discarded.

21. **True or False: The `createGlobalTempView` method creates a temporary view that persists even after the SparkSession ends.**

22. A. True
23. B. False
    Correct Option: B
    Explanation: This statement is **False**. The `createGlobalTempView` method creates a global temporary view that is shared among *all* SparkSessions of an application, but it still lives only for the lifetime of the Spark application (not persistent beyond that). A regular `createOrReplaceTempView` is session-scoped. (The docs mention that both are temporary and tied to Spark's lifetime [35].)

Session 10: Connecting Databases with Spark

1. **Which Spark SQL data source is used to read from relational databases (via JDBC) into a DataFrame?**
2. A. `spark.read.format("parquet")`
3. B. `spark.read.format("jdbc")`
4. C. `spark.read.format("sql")`
5. D. `spark.read.format("jdbcRDD")`
   Correct Option: B
   Explanation: Spark SQL provides a JDBC data source for reading tables from relational databases. The Spark documentation notes that this JDBC source "reads data from other databases using JDBC and returns a DataFrame" [36]. You use `spark.read.format("jdbc")` with the appropriate JDBC URL and table options.

6. **What option key should you provide to specify the JDBC connection URL when reading from a database?**

7. A. `schema`
8. B. `url`
9. C. `driver`
10. D. `database`
    Correct Option: B
    Explanation: When using `spark.read.format("jdbc")`, you must specify the database

connection URL with the `url` option. The documentation's option table shows that `url` is the JDBC connection string (e.g., `"jdbc:postgresql://host/db"`) [37].

11. **How do you specify which table to read from in Spark's JDBC reader?**

12. A. `tableName` option
13. B. `dbtable` option
14. C. `query` option

15. D. `dataset` option
    Correct Option: B
    Explanation: The `dbtable` option is used to specify the table name or subquery to read from. In Spark's JDBC options, `dbtable` identifies the table (or a query by putting parentheses around it) that the DataFrameReader should load [37].

16. **What must you include in Spark's classpath to connect to a specific database (e.g., PostgreSQL) via JDBC?**

17. A. The database client software.
18. B. The JDBC driver JAR for that database.
19. C. A Spark SQL extension library.

20. D. It requires no additional JARs.
    Correct Option: B
    Explanation: To use Spark's JDBC data source with a particular database, you need the JDBC driver for that database on the Spark classpath. For example, connecting to PostgreSQL requires adding the `postgresql.jar` driver. The Spark documentation explicitly states that you need to include the JDBC driver JAR on the classpath [38].

21. **When reading a database table with Spark, why is using the JDBC DataFrame source preferred over the older `JdbcRDD`?**

22. A. JDBC DataFrame source is faster by default.
23. B. It returns data as a DataFrame, enabling Spark SQL optimizations.
24. C. `JdbcRDD` is deprecated and not supported.
25. D. JDBC DataFrame source automatically parallelizes the query.
    Correct Option: B
    Explanation: The Spark documentation advises using the DataFrame JDBC source over `JdbcRDD` because it returns a DataFrame. This allows leveraging Spark's higher-level APIs and optimizations. Specifically, the docs state that this approach "should be preferred over using `JdbcRDD`, because results are returned as a DataFrame which can be queried and manipulated" [36].

Session 11: Git Basics

1. **What does the command `git init` do in a directory?**
2. A. Deletes the directory's files.
3. B. Initializes a new Git repository by creating a `.git` folder.

4. C. Clones a remote repository into the directory.

5. D. Pushes local changes to GitHub.
   Correct Option: B
   Explanation: Running `git init` creates a new Git repository in the current directory. This is done by creating a hidden `.git` subdirectory containing all the repository's metadata. The Git documentation notes that `git init` "creates a new subdirectory named .git that contains all of your necessary repository files" [39] .

6. **How do you clone an existing Git repository from GitHub or another remote?**

7. A. `git download <url>`
8. B. `git clone <url>`
9. C. `git init <url>`

10. D. `git copy <url>`
    Correct Option: B
    Explanation: To copy an existing repository, you use `git clone <repository-url>`. This command downloads the entire repository (including all its history) from the specified URL. The Git docs explain that `git clone` creates a complete local copy of the remote repo [40] .

11. **What is the effect of `git add` followed by `git commit`?**

12. A. It stages changes and then records a snapshot to the repository.
13. B. It deletes the specified files from version control.
14. C. It renames the current branch.

15. D. It fetches updates from the remote repository.
    Correct Option: A
    Explanation: `git add` stages changes (new or modified files) for commit, and `git commit` records those staged changes into the repository's history as a new commit. The Git guide shows an example where after making changes, one runs `git add` and then `git commit -m "message"` to save the work [41] .

16. **True or False: `git clone` downloads only the latest snapshot of the files.**

17. A. True

18. B. False
    Correct Option: B
    Explanation: This statement is **False**. Unlike some version control systems, `git clone` downloads *the entire history* of the repository, not just a single snapshot. The Git docs explicitly note that Git is distributed: "Instead of a partial copy, `git clone` creates a copy of *nearly all the data*" in the repo [42] . This includes all commits and branches.

19. **What does the `git add` command do?**

20. A. Creates a new commit directly.
21. B. Stages changes in the working directory for the next commit.
22. C. Downloads changes from a remote repository.
23. D. Initializes a new repository.
    Correct Option: B
    Explanation: `git add` tells Git to begin tracking changes (or new files) and stage them for the next commit. After `git add`, these changes will be included when you next run `git commit`. In the example from the Git documentation, files are added with `git add` before committing [41] .

Session 12: Git & GitHub

1. **Which command sends your local commits to a remote repository (such as GitHub)?**
2. A. `git fetch`
3. B. `git pull`
4. C. `git push`
5. D. `git clone`
   Correct Option: C
   Explanation: `git push` uploads local branch commits to the remote repository. As Atlassian's Git guide states, pushing is how you transfer commits "from your local repository to the remote repository" [43] . This is the standard way to publish local work to GitHub or another remote server.

6. **Which command is used to configure remote repositories in Git?**

7. A. `git remote`
8. B. `git branch`
9. C. `git checkout`
10. D. `git init`
    Correct Option: A
    Explanation: The `git remote` command manages the set of tracked remote repositories (for example, adding an "origin" remote with a URL). Git's documentation notes that remote branches are configured using `git remote` [44] . For instance, `git remote add origin <url>` sets the "origin" remote.

11. **In Git terminology, what is the counterpart of "git push" that is used to bring changes from the remote to local?**

12. A. `git download`
13. B. `git pull` or `git fetch`
14. C. `git merge`
15. D. `git commit`
    Correct Option: B
    Explanation: The operations to get (download) changes from remote are `git fetch` and `git pull`. In Git, pushing is considered the upload, whereas fetching/pulling are the download. The

guide explains that "push is used to upload your content, whereas git fetch (or pull) is like the download command" [45] .

16. **What does the `git pull` command do?**

17. A. It merges the current branch into another branch.
18. B. It fetches from the remote and immediately merges into the current branch.
19. C. It deletes remote branches.

20. D. It initializes a remote repository.
    Correct Option: B
    Explanation: `git pull` is effectively a `git fetch` followed by `git merge`. It downloads data from the remote repository and integrates it into your current branch. As described by the Git documentation, `git pull` "fetches and downloads content from a remote repository and immediately updates the local repository" (by merging) [46] .

21. **True or False: The `git remote add origin <URL>` command both adds a remote and uploads your local branch.**

22. A. True
23. B. False
    Correct Option: B
    Explanation: This statement is **False**. `git remote add origin <URL>` only adds a new remote reference named "origin"; it does not send any commits. To upload commits after adding the remote, you would use `git push`. The `git remote add` command simply tells Git where the remote repository is located.

Session 13: Docker Basics

1. **Which statement correctly describes a Docker container image?**
2. A. A process that runs in the background.
3. B. A lightweight, standalone, executable package that includes application code and its dependencies.
4. C. A virtual machine image including a full operating system.

5. D. A database snapshot for containerized storage.
   Correct Option: B
   Explanation: A Docker container image is a package that contains everything needed to run an application: code, runtime, system tools, libraries, and settings. The Docker documentation defines it as "a lightweight, standalone, executable package of software that includes everything needed to run an application" [47] .

6. **How do Docker containers achieve isolation from the host environment?**

7. A. By running a separate OS kernel in each container.
8. B. By using a virtual machine hypervisor.
9. C. By isolating applications in user-space while sharing the host's OS kernel.

10. D. By encrypting all container processes.
    Correct Option: C
    Explanation: Docker containers run isolated processes in user-space and share the host machine's kernel. The official docs note that containers use the host's kernel and do not include a separate OS, which makes them lightweight and portable [48]. This differs from VMs which each have their own full OS.

11. **What is an advantage of Docker containers compared to virtual machines (VMs)?**

12. A. Containers include a full OS, so they are more portable.
13. B. Containers are much smaller in size because they share the host OS kernel.
14. C. Containers require a hypervisor to run.

15. D. Containers always run slower than VMs.
    Correct Option: B
    Explanation: Containers are more lightweight than VMs because they only package the application and necessary dependencies, sharing the host OS kernel rather than including a full OS. The Docker documentation highlights that container images can be tens of MBs (vs. GBs for VMs) and all share the host's kernel [49]. This makes them faster to start and use fewer resources.

16. **Which Docker command lists only the running containers by default?**

17. A. `docker ls`
18. B. `docker ps`
19. C. `docker images`

20. D. `docker list`
    Correct Option: B
    Explanation: The command `docker ps` lists running containers by default. The Docker reference states that `docker ps` shows only running containers, whereas to list all containers (including stopped ones) you would use `docker ps --all` [50].

21. **Which command builds a Docker image from a Dockerfile in the current directory?**

22. A. `docker run .`
23. B. `docker build .`
24. C. `docker pull .`
25. D. `docker create .`
    Correct Option: B
    Explanation: `docker build .` tells Docker to build an image using the `Dockerfile` in the current directory. The Docker documentation provides this basic example: `docker build .`, which reads the Dockerfile in `.` and builds the image [51].

Session 14: Docker Usage and Networking

1. **Which command is an alias for `docker container ls` and lists containers?**
2. A. `docker ps`

3. B. `docker ls`
4. C. `docker images`

5. D. `docker run`
   Correct Option: A
   Explanation: The `docker ps` command is a shortcut for `docker container ls`; both list containers (running by default). The Docker CLI reference shows that `docker ps` is an alias for listing containers [52] .

6. **What does the `docker ps` command show by default?**

7. A. All containers, including stopped ones.
8. B. Only running containers.
9. C. Only container images.

10. D. All networks.
    Correct Option: B
    Explanation: By default, `docker ps` displays only running containers. To see all containers (running or stopped), one would add the `--all` flag. The Docker docs explicitly note that without flags, `docker ps` shows "running containers by default" [50] .

11. **What is the purpose of the `-p` flag when running a Docker container?**

12. A. It specifies the container's CPU priority.
13. B. It publishes (maps) a container port to the host.
14. C. It runs the container in privileged mode.

15. D. It pauses the container on start.
    Correct Option: B
    Explanation: The `-p HOST_PORT:CONTAINER_PORT` flag in `docker run` publishes a container's port to the host. This allows services in the container to be accessible on the host machine's port. The Docker documentation provides examples: e.g., `docker run -d -p 8080:80 nginx` maps port 80 in the container to port 8080 on the host [53] .

16. **Which file name is used to define a Docker build process (instructions to build an image)?**

17. A. `Dockerfile`
18. B. `container.yml`
19. C. `Dockerfile.txt`

20. D. `Buildfile`
    Correct Option: A
    Explanation: A file named `Dockerfile` (capital "D") contains the instructions (commands) for building a Docker image. The documentation states that images are built using a `Dockerfile` in the working directory [51] .

21. **How do you create and start a new container from an image?**

22. A. `docker run <image-name>`
23. B. `docker create <image-name>`
24. C. `docker build <image-name>`
25. D. `docker up <image-name>`
    Correct Option: A
    Explanation: The `docker run` command creates and starts a new container from a specified image. For example, `docker run -d <image-id>` (shown in the docs) creates a container from that image and runs it [54]. (The `docker create` command would create a container but not start it.)

Session 15: Kubernetes Basics

1. **What are the two main components of a Kubernetes cluster?**
2. A. Master Node and Client Node
3. B. Control Plane and Worker Nodes
4. C. Cluster and Pod

5. D. Scheduler and API Server
   Correct Option: B
   Explanation: A Kubernetes cluster consists of a control plane (managing the cluster) and one or more worker machines called nodes. The Kubernetes documentation specifies that a cluster is made up of "the control plane and a set of worker machines called nodes" [55].

6. **In Kubernetes, what is a *Node*?**

7. A. A single container running in the cluster.
8. B. A worker machine (VM or physical) that runs container workloads.
9. C. A namespace within the cluster.

10. D. A network plugin for containers.
    Correct Option: B
    Explanation: A node is a worker machine in Kubernetes that runs containers (in Pods). The docs describe each worker machine in a cluster as a node that runs the kubelet and container runtime to run pods [55].

11. **What is a *Pod* in Kubernetes?**

12. A. A network segment in the cluster.
13. B. The smallest deployable unit, representing one or more containers running together.
14. C. A persistent storage volume.

15. D. A type of Kubernetes Service.
    Correct Option: B
    Explanation: A Pod is the smallest deployable unit in Kubernetes and represents one or more containers that share the same resources and network namespace. The official docs define a pod as "a set of one or more running containers on your cluster" [56].

16. **Which Kubernetes workload resource would you use to run stateless applications with a specified number of replicas?**

17. A. StatefulSet
18. B. Deployment
19. C. DaemonSet

20. D. Service
    Correct Option: B
    Explanation: A Deployment is used for stateless applications where you want to run multiple copies (replicas) of a pod. The Kubernetes documentation notes that a Deployment is suitable for stateless workloads with interchangeable pods [57].

21. **Which Kubernetes resource ensures that exactly one copy of a Pod runs on each node in the cluster?**

22. A. Job
23. B. StatefulSet
24. C. DaemonSet
25. D. ReplicaSet
    Correct Option: C
    Explanation: A DaemonSet makes sure that one (or a specified number of) pod(s) run on each (or some) node. It is typically used to run one pod per node, such as for logging or network plugins. The docs describe DaemonSet as a way to run a Pod on every node that matches the selector [58].

Session 16: Kubernetes Services and Updates

1. **What is a Kubernetes Service used for?**
2. A. To store application logs.
3. B. To expose a set of Pods as a network service under a single IP or DNS name.
4. C. To schedule Pods on nodes.

5. D. To provide container runtime.
   Correct Option: B
   Explanation: A Service in Kubernetes provides a stable network endpoint (cluster IP or DNS) for a set of Pods, enabling other components or external clients to communicate with those Pods. The Kubernetes docs state that a Service is an abstraction for exposing "a network application running as a set of Pods" [59].

6. **What is the default update strategy for a Deployment in Kubernetes?**

7. A. Recreate (kill all old pods then start new ones)
8. B. RollingUpdate (gradual replacement of pods)
9. C. Blue-Green

10. D. Canary
    Correct Option: B
    Explanation: The default strategy for updating Deployments is `RollingUpdate`. This means pods

are updated gradually. The Kubernetes spec mentions that for Deployment, the default `spec.strategy.type` is "RollingUpdate" [60].

11. **What Kubernetes component runs on each worker node and is responsible for network proxying and service load balancing?**

12. A. etcd
13. B. kubelet
14. C. kube-proxy

15. D. kube-scheduler
    Correct Option: C
    Explanation: The `kube-proxy` component runs on each node and is responsible for implementing Kubernetes networking services (e.g. the cluster IP and port forwarding) for Services. The architecture diagram notes that each node runs `kube-proxy` to ensure that Service network rules are applied [61].

16. **When using a RollingUpdate strategy, what do `maxUnavailable` and `maxSurge` parameters control?**

17. A. They control how many pods can be stopped or started at once during an update.
18. B. They control network bandwidth for rolling updates.
19. C. They specify which nodes are updated first.

20. D. They disable rolling updates.
    Correct Option: A
    Explanation: In a RollingUpdate, `maxUnavailable` sets the maximum number of pods that can be down during the update, and `maxSurge` sets how many extra pods can be created above the desired count. This controls the speed and resource allocation of the update. (Although not explicitly cited above, this is standard Kubernetes behavior for rolling updates.)

21. **What is a Namespace in Kubernetes?**

22. A. A node in the cluster.
23. B. A virtual cluster/partition within a Kubernetes cluster for grouping resources.
24. C. The name of a container runtime.
25. D. The label given to a deployment.
    Correct Option: B
    Explanation: A Namespace is a way to divide cluster resources among multiple users or teams. It provides a scope for names. (Not explicitly in the above citations, but a core concept: namespaces allow multiple virtual clusters backed by the same physical cluster.)

---

[1] Getting Started - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/sql-getting-started.html

[2] [3] [4] [5] [6] [7] [8] RDD Programming Guide - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/rdd-programming-guide.html

[9] CSV Files - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/sql-data-sources-csv.html

[10] pyspark.sql.DataFrame.describe — PySpark 3.5.5 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.describe.html

[11] pyspark.sql.DataFrame.dropna — PySpark 3.5.5 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.dropna.html

[12] pyspark.sql.DataFrame.fillna — PySpark 3.5.5 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.fillna.html

[13] pyspark.sql.DataFrame.filter — PySpark 4.0.0 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.filter.html

[14] [15] pyspark.sql.DataFrameWriter.parquet — PySpark 4.0.0 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameWriter.parquet.html

[16] pyspark.sql.DataFrame.coalesce — PySpark 3.5.5 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.coalesce.html

[17] [18] [19] Kafka Partitions: Essential Concepts for Scalability and Performance | DataCamp
https://www.datacamp.com/tutorial/kafka-partitions

[20] [21] [22] [24] Structured Streaming + Kafka Integration Guide (Kafka broker version 0.10.0 or higher) - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/streaming/structured-streaming-kafka-integration.html

[23] How Kafka Connect Works: Integrating Data Between Systems
https://developer.confluent.io/courses/kafka-connect/how-connectors-work/

[25] Clustering - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/ml-clustering.html

[26] Collaborative Filtering - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/ml-collaborative-filtering.html

[27] ML Pipelines - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/ml-pipeline.html

[28] Frequent Pattern Mining - Spark 3.5.5 Documentation
https://spark.apache.org/docs/latest/ml-frequent-pattern-mining.html

[29] [31] [32] Classification and regression - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/ml-classification-regression.html

[30] Microsoft Word - BigDL_distributed_DL1.docx
https://www.intel.com/content/dam/develop/external/us/en/documents/bigdl-distributed-dl1.pdf

[33] [34] [35] pyspark.sql.DataFrame.createOrReplaceTempView — PySpark 4.0.0 documentation
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.createOrReplaceTempView.html

[36] [37] [38] JDBC To Other Databases - Spark 4.0.0 Documentation
https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

[39] [40] [41] [42] Git - Getting a Git Repository
https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository

[43] [44] [45] Git Push | Atlassian Git Tutorial
https://www.atlassian.com/git/tutorials/syncing/git-push

[46] Git Pull | Atlassian Git Tutorial
https://www.atlassian.com/git/tutorials/syncing/git-pull

[47] [48] [49] What is a Container? | Docker
https://www.docker.com/resources/what-container/

[50] [52] docker container ls | Docker Docs
https://docs.docker.com/reference/cli/docker/container/ls/

[51] [54] Build, tag, and publish an image | Docker Docs
https://docs.docker.com/get-started/docker-concepts/building-images/build-tag-and-publish-an-image/

[53] Publishing and exposing ports | Docker Docs
https://docs.docker.com/get-started/docker-concepts/running-containers/publishing-ports/

[55] [61] Cluster Architecture | Kubernetes
https://kubernetes.io/docs/concepts/architecture/

[56] [57] [58] Workloads | Kubernetes
https://kubernetes.io/docs/concepts/workloads/

[59] Service | Kubernetes
https://kubernetes.io/docs/concepts/services-networking/service/

[60] Deployments | Kubernetes
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/