🔹 Beginner Level PySpark Coding Questions with Answers

1️⃣ Read CSV file into DataFrame

```
df = spark.read.csv("path/to/file.csv", header=True, inferSchema=True)
```

✅ **Explanation:** Reads CSV with headers and infers data types automatically.

2️⃣ Select specific columns from a DataFrame

```
df.select("column1", "column2").show()
```

✅ **Explanation:** Selects and displays specific columns from the DataFrame.

3️⃣ Filter rows based on a condition

```
df.filter(df["age"] > 30).show()
```

✅ **Explanation:** Filters rows where the age is greater than 30.

4️⃣ Add a new column using withColumn()

```
df = df.withColumn("new_col", df["salary"] * 0.1)
```

✅ **Explanation:** Adds a new column new_col as 10% of the salary.

5️⃣ Group by and aggregate

```
df.groupBy("department").agg({"salary": "avg"}).show()
```

✅ **Explanation:** Computes the average salary for each department.

🔸 Intermediate PySpark Coding Questions with Answers

6️⃣ Join two DataFrames

```
joined_df = df1.join(df2, df1.id == df2.id, "inner")
```

✅ **Explanation:** Performs an inner join on the id column between two DataFrames.

7️⃣ Drop duplicates

```
df.dropDuplicates(["name", "age"]).show()
```

✅ **Explanation:** Removes rows that have the same combination of name and age.

8️⃣ Handle null values

```
df.na.fill("Unknown").na.drop()
```

✅ **Explanation:** Fills nulls with "Unknown" and then drops any remaining nulls.

9️⃣ Window functions (e.g., row_number, rank)

```
from pyspark.sql.window import Window

from pyspark.sql.functions import row_number


window_spec = Window.partitionBy("department").orderBy("salary")

df.withColumn("row_num", row_number().over(window_spec)).show()
```

✅ **Explanation:** Assigns a row number to each row within a department based on salary order.

🔟 Convert string to timestamp

```
from pyspark.sql.functions import to_timestamp


df = df.withColumn("timestamp_col", to_timestamp("date_col", "yyyy-MM-dd HH:mm:ss"))
```

✅ **Explanation:** Converts the date_col string to timestamp format.

🔥 Advanced PySpark Coding Questions with Answers

1️⃣1️⃣ Explode a nested array column

```
from pyspark.sql.functions import explode


df = df.withColumn("exploded_col", explode(df["array_col"]))
```

✅ **Explanation:** Flattens an array column into multiple rows.

1️⃣2️⃣ Pivot data

```
df.groupBy("name").pivot("year").agg({"sales": "sum"}).show()
```

✅ **Explanation:** Converts unique year values into separate columns with summed sales.

1️⃣3️⃣ Create and use UDF (User Defined Function)

```
from pyspark.sql.functions import udf

from pyspark.sql.types import StringType


def get_length(s):

    return len(s)


length_udf = udf(get_length, StringType())

df = df.withColumn("length", length_udf(df["name"]))
```

✅ **Explanation:** Defines a UDF to calculate string length and applies it to the name column.

1️⃣4️⃣ Broadcast join for performance

```
from pyspark.sql.functions import broadcast


df = df1.join(broadcast(df2), "id")
```

✅ **Explanation:** Broadcasts the smaller DataFrame to all executors to avoid shuffles during joins.

1️⃣5️⃣ Read from and write to Parquet

```
# Write to Parquet

df.write.parquet("path/to/output.parquet")


# Read from Parquet
```

```
df_parquet = spark.read.parquet("path/to/output.parquet")
```

✅ **Explanation:** Saves the DataFrame as a Parquet file and reads it back.

🚀 Performance & Optimization Questions with Answers

1️⃣6️⃣ How to cache or persist a DataFrame

```
df.cache()       # Stores in memory

df.persist()     # Can store in memory or disk (customizable)
```

✅ **Explanation:** Avoids recomputation by storing the DataFrame in memory or disk.

1️⃣7️⃣ How to check the execution plan

```
df.explain(True)
```

✅ **Explanation:** Shows a detailed physical execution plan, helping diagnose issues like shuffles or scans.

1️⃣8️⃣ Partitioning a DataFrame before saving

```
df.write.partitionBy("year", "month").parquet("path")
```

✅ **Explanation:** Saves data into folders based on year and month to optimize query performance.

🎯 Scenario-Based Questions with Solutions

1️⃣9️⃣ Find top N records per group (e.g., top 3 salaries per department)

```
from pyspark.sql.window import Window

from pyspark.sql.functions import row_number


windowSpec = Window.partitionBy("department").orderBy(df["salary"].desc())


df = df.withColumn("rank", row_number().over(windowSpec)).filter("rank <= 3")
```

✅ **Explanation:** Assigns a rank based on salary within each department and filters top 3.

2️⃣0️⃣ Detect duplicates and count them

```
df.groupBy("id").count().filter("count > 1").show()
```

✅ **Explanation:** Groups by id, counts occurrences, and filters where count is greater than 1 (duplicates).

🧠 Bonus – Common Conceptual Questions:

- What is the difference between transformations and actions in Spark?
- Explain wide vs. narrow transformations.
- What causes shuffles, and how do they impact performance?
- How does Spark handle fault tolerance?
- Explain Spark's DAG and stages.

✅ Conclusion:

Mastering these PySpark coding patterns not only prepares you for interviews but also makes you job-ready for real-world data engineering challenges on **Databricks, AWS, Azure, or GCP.** 💪 🚀