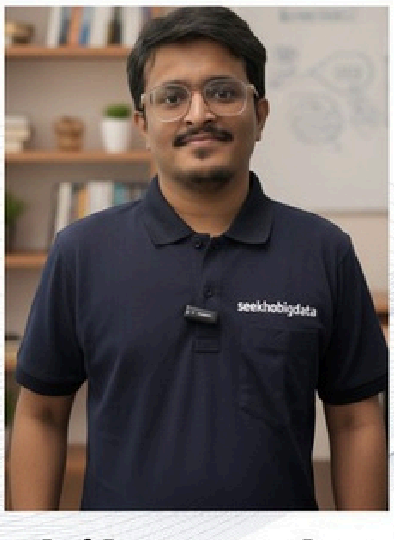
Seekho Bigdata
Data is the New Oil

PySpark Scenario-Based Interview Questions (Complete Notes Series)

DAY 14 — Structured Streaming



Karthik Kondpak
9989454737

PySpark Scenario-Based Interview Questions (Complete Notes Series)

DAY 14 — Structured Streaming (Real-Time Data Pipelines)

Concepts Covered Today

- What is Structured Streaming
- Micro-batch vs Continuous processing
- Sources & sinks (Kafka, Delta)
- Checkpointing & fault tolerance
- Event time, watermarking
- Streaming joins & aggregations

Scenario

Youworkfor an **Indian UPI / payments platform**.

Stream: transactions_stream (from Kafka)

- transaction_id
- merchant_id
- amount
- event_time

Requirement: Calculate **real-time total amount per merchant every 5 minutes**.

What is Structured Streaming?

✅ Definition

Structured Streaming treats streaming data as an **unbounded table** and processes it using **Spark SQL semantics**.

Key idea: *“Streaming = continuous incremental batch processing.”*

Question 1: Read from Kafka

PySpark Code

```
stream_df = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "broker:9092")
\
    .option("subscribe", "upi_transactions") \
    .load()
```

🔄 Question 2: Parse Streaming Data

🔹

Kafka value is JSON.
Scenario

```
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import *

schema = StructType([
    StructField("transaction_id", StringType()),
    StructField("merchant_id", StringType()),
    StructField("amount", DoubleType()),
    StructField("event_time", TimestampType())
])

parsed_df = stream_df.select(
    from_json(col("value").cast("string"),
schema).alias("data")
).select("data.*")
```

Question 3: Windowed Aggregation

◆ Scenario

Calculate merchant-wise total every 5 minutes.

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

agg_df = parsed_df \
    .withWatermark("event_time", "10 minutes")\
    .groupBy(
        window("event_time", "5 minutes"),
        "merchant_id"
    ).agg(sum("amount").alias("total_amount"))
```

💧 Watermarking (Interview Favourite)

🔹 Why Needed?

- Handles late arriving data
- Prevents unbounded state growth

Example: Allow events up to **10 minutes late**.

Question 4: Checkpointing (Fault Tolerance)

🔹 Scenario

Ensure exactly-once processing.

```
query=  agg_df.writeStream \
    .format("delta") \
    .outputMode("append") \
    .option("checkpointLocation", "/chk/upi_txn") \
    .start("/delta/merchant_agg")
```

Output Modes

Mode	Use Case
append	New rows only
update	Changed rows
complete	Full result table

🔁 Question 5: Streaming Join (Advanced)

◆ Scenario

Join transaction stream with static merchant data.

```
joined_df  =  parsed_df.join(
broadcast(merchants_df),
"merchant_id"
)
```

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/