Seekho Bigdata
Data is the New Oil

PySpark Scenario-Based Interview Questions

DAY 4 — Handling Duplicates & Latest Records (SCD, Deduplication)

Karthik Kondpak
9989454737

PySpark Scenario-Based Interview Questions

DAY 4 — Handling Duplicates & Latest Records (SCD, Deduplication)

Concepts Covered Today

- Removing duplicates using window functions
- Latest record per key Slowly Changing
- Dimension (SCD) Type-1 Slowly Changing
- Dimension (SCD) Type-2 Real-time data
- warehouse scenarios

Common Sample Data: `customer_updates_df`

customer_id	customer_name	city	update_ts
101	Rahul	Bangalore	2024-01-01 10:00:00
101	Rahul	Chennai	2024-02-01 09:00:00
102	Priya	Mumbai	2024-01-15 11:30:00
102	Priya	Pune	2024-03-01 08:45:00

✅ Question 1: Remove Duplicate Records and Keep Latest Entry

🔹 Scenario

Due to multiple updates arriving for the same customer, duplicates exist. Keep **only the latest record per customer**.

🧪 PySpark Solution

```
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number

window_spec =
Window. partit ionBy ("cus tomer_ id").or derBy( col(" updat e
_ts").desc())

latest_df = customer_updates_df.withColumn(
    "rn",
    row_number().over(window_spec)
).filter(col("rn") == 1).drop("rn")
```

```
latest_df.show()
```

📝 Explanation

- row_number() helps identify latest record
- Sorting by timestamp in descending order
- Standard deduplication pattern in pipelines

✅ Question 2: SCD Type-1 — Overwrite Old Records

🔹 Scenario

A customer dimension table should always contain **only the latest customer information**, without preserving history.

🧪 PySpark Logic (Conceptual)

```
# Replace old records with latest ones
scd_type1_df = latest_df
```

```
scd_type1_df.show()
```

📝 Explanation

- SCD Type-1 **does not maintain history**
- Old values are overwritten
- Used when historical values are not required

✅ Question 3: SCD Type-2 — Maintain Full History

🔹 Scenario

Business wants to track **historical changes** in customer attributes like city.

📊 Target Table Structure

| customer_id | customer_name | city | start_date | end_date | is_current |

🧪 PySpark Solution (Simplified)

```
from pyspark.sql.functions import lead, lit

scd_window =
Window. partit ionBy ("cus tomer_ id").or derBy( "upda te_ts "
)

scd2_df = customer_updates_df \
    .withColumn("end_date",
lead("update_ts").over(scd_window)) \
    .withColumn("start_date", col("update_ts")) \
    .withColumn("is_current",
col("end_date").isNull())

scd2_df.show()
```

📝 Explanation

- Each change creates a new record
- end_date is populated using lead()
- is_current = true identifies active record

- Very common in **data warehouse interviews**

✅ Question 4: Identify Records Updated More Than Once

🔹 Scenario

Data quality team wants to find customers who have **multiple updates**.

🧪 PySpark Solution

```
from pyspark.sql.functions import count

multiple_updates_df =
customer_updates_df.groupBy("customer_id") \
.agg(count("*").alias("update_count")) \
.filter(col("update_count") > 1)

multiple_updates_df.show()
```

📝 Explanation

- Helps identify frequently changing records
- Useful for auditing and debugging pipelines

Let's build your Data Engineering journey together!

Call us directly at: 9989454737

https://seekhobigdata.com/