



PySpark Scenario-Based Interview Questions

DAY 8 –

Joins + Aggregations



Karthik Kondpak
9989454737

PySpark Scenario-Based Interview

Questions (Complete Notes Series)

DAY 8 — Joins + Aggregations (Complex Business Metrics)

Concepts Covered Today

- Joins followed by aggregations
- Multi-table business metrics
- Revenue, conversion & performance KPIs
- Handling NULLs after joins
- Join + aggregation optimization mindset

Sample Data (Indian E-Commerce Scenario) `customers_df`

<code>customer_id</code>	<code>customer_name</code>	<code>city</code>
101	Rahul	Bangalore
102	Priya	Mumbai

103	Aman	Hyderabad
104	Neha	Delhi

orders_df

order_id	customer_id	order_date	amount
1	101	2024-01-01	2000
2	101	2024-01-10	1500
3	102	2024-01-15	3000
4	103	2024-02-01	2500

✓ Question 1: Total Revenue Per City

◆ Scenario

Management wants to analyze **total revenue generated from each Indian city.**

✍ PySpark Solution

```
from pyspark.sql.functions import sum

city_revenue_df = customers_df.join(
    orders_df,
    "customer_id",
    "inner"
```

```
 ).groupBy("city") \
 .agg(sum("amount").alias("total_revenue"))

city_revenue_df.show()
```



Explanation

- Join customer and order data
- Aggregate after join to calculate KPIs
- Very common dashboard metric



Question 2: Average Order Value Per Customer

◆ Scenario

Analytics team wants **Average Order Value (AOV)** per customer.

PySpark Solution

```
from pyspark.sql.functions import avg
```

```
customer_aov_df = orders_df.groupBy("customer_id") \
    .agg(avg("amount").alias("avg_order_value"))

customer_aov_df.show()
```



Explanation

- AOV is a core e-commerce KPI
- Often extended with joins to customer table

✓ Question 3: Customers With No Orders (Join + Aggregation Logic)

◆ Scenario

Business wants to find **customers who have never placed any order.**

PySpark Solution

```
from pyspark.sql.functions import count

customer_order_count_df = customers_df.join(
    orders_df,
    "customer_id",
    "left"
).groupBy("customer_id", "customer_name") \
.agg(count("order_id").alias("order_count")) \
.filter(col("order_count") == 0)

customer_order_count_df.show()
```



Explanation

- Left join keeps all customers
- Aggregation identifies zero-order customers
- Alternative: left_anti join (discuss in interview)

Question 4: Monthly Revenue Per City

◆ Scenario

Finance team needs **month-wise revenue per city** for trend analysis.



PySpark Solution

```
from pyspark.sql.functions import month, year

monthly_city_df = customers_df.join(
    orders_df,
    "customer_id",
    "inner"
).withColumn("year", year("order_date")) \
.withColumn("month", month("order_date")) \
.groupBy("city", "year", "month") \
.agg(sum("amount").alias("monthly_revenue"))

monthly_city_df.show()
```



Explanation

- Combines join + date logic + aggregation
- Typical reporting pipeline requirement

Question 5: Identify Top Revenue-Generating City

◆ Scenario

Leadership wants to know **which city contributes the highest revenue.**



PySpark Solution

```
from pyspark.sql.window import Window
from pyspark.sql.functions import rank, desc

rank_window = Window.orderBy(desc("total_revenue"))

ranked_city_df = city_revenue_df.withColumn(
    "rank",
    rank().over(rank_window)
).filter(col("rank") == 1)
```

```
ranked_city_df.show()
```



Explanation

- Ranking after aggregation
- Common executive-level metric



**Let's build your Data
Engineering journey
together!**



Call us directly at: 9989454737



<https://seekhobigdata.com/>

