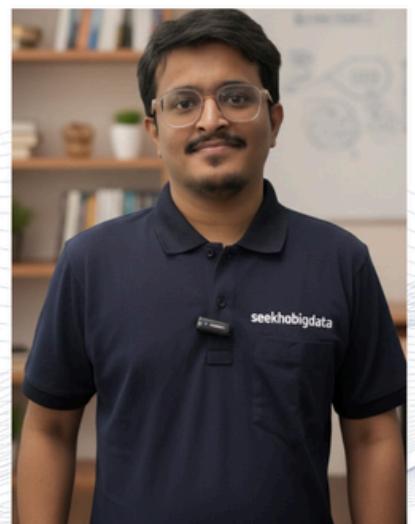




# PySpark Scenario-Based Interview Questions

DAY 6 –

## Date & Time Handling



**Karthik Kondpak**  
**9989454737**

# PySpark Scenario-Based Interview

## Questions (Complete Notes Series)

### DAY 6 — Date & Time Handling

#### Concepts Covered Today

- Date parsing & formatting
- Date difference calculations
- Month-wise & day-wise aggregations
- Rolling windows using dates
- Business logic with dates

#### Sample Data: orders\_df

order_id	customer_id	order_date	delivery_date	amount
1	101	2024-01-01	2024-01-05	2000
2	101	2024-01-15	2024-01-20	1500
3	102	2024-02-01	2024-02-10	3000
4	103	2024-02-18	NULL	2500

## Question 1: Convert String Columns to Date Type

### ◆ Scenario

Incoming data has dates as **string**. Convert them to proper date format before processing.



### PySpark Solution

```
from pyspark.sql.functions import to_date, col

converted_df = orders_df \
    .withColumn("order_date",
    to_date(col("order_date"))) \
    .withColumn("delivery_date",
    to_date(col("delivery_date")))

converted_df.printSchema()
```



### Explanation

- Always validate schema before date operations

- Prevents runtime errors in aggregations

## Question 2: Calculate Order Delivery Duration (Date Difference)

### ◆ Scenario

Business wants to calculate **delivery duration in days** for completed orders.



### PySpark Solution

```
from pyspark.sql.functions import datediff

delivery_time_df = converted_df.withColumn(
    "delivery_days",
    datediff(col("delivery_date"), col("order_date"))
)

delivery_time_df.show()
```



## Explanation

- datediff() returns number of days between two dates
- Common KPI in logistics dashboards



## Question 3: Extract Year and Month for Reporting

### ◆ Scenario

Finance team wants **monthly revenue reporting**.



## PySpark Solution

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

monthly_sales_df = converted_df \
    .withColumn("year", year("order_date")) \
    .withColumn("month", month("order_date")) \
    .groupBy("year", "month") \
    .agg(sum("amount").alias("monthly_revenue"))

monthly_sales_df.show()
```



## Explanation

- Month-wise aggregation is extremely common
- Helps demonstrate reporting layer design



## Question 4: Identify Orders Not Delivered Within SLA (7 Days)

### ◆ Scenario

Orders should be delivered within **7 days**. Identify delayed orders.



## PySpark Solution

```
sla_df = delivery_time_df.filter(col("delivery_days")  
> 7)  
  
sla_df.show()
```



## Explanation

- Business SLA checks are common interview scenarios
- Shows real operational thinking



## Question 5: Calculate Rolling 30-Day Sales

### ◆ Scenario

Analytics team needs **rolling 30-day revenue** per customer.



## PySpark Solution

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

rolling_window = Window.partitionBy("customer_id") \
    .orderBy(col("order_date").cast("timestamp")) \
    .rangeBetween(-30 * 86400, 0)

rolling_df = converted_df.withColumn(
    "rolling_30_day_sales",
    sum("amount").over(rolling_window)
```

```
)  
  
rolling_df.show()
```



### Explanation

- rangeBetween works with timestamps
- Very popular in **fintech & analytics interviews**



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

 Call us directly at: 9989454737

 <https://seekhobigdata.com/>

