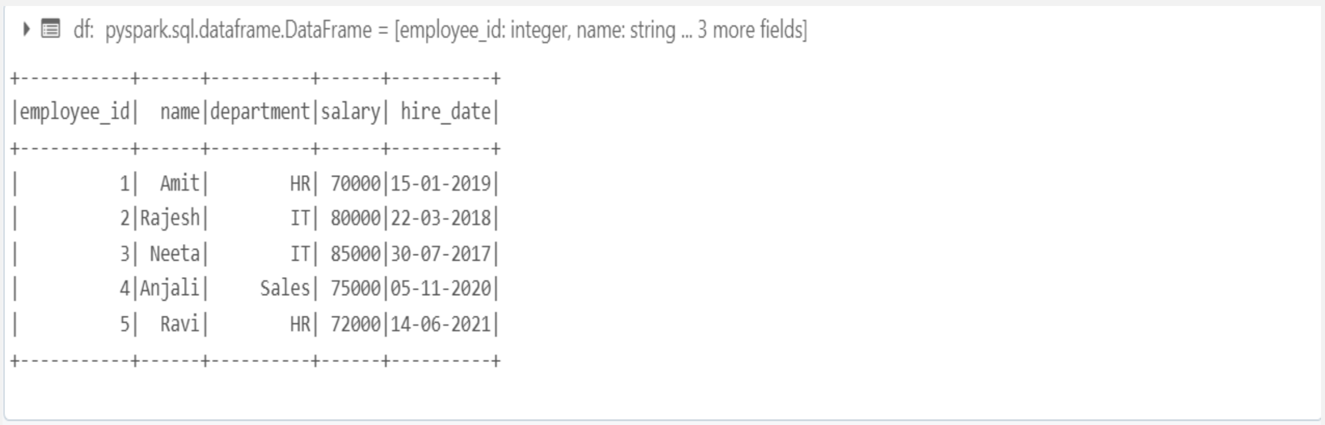
Top 5 PySpark Interview Questions and Answers

Create DataFrame

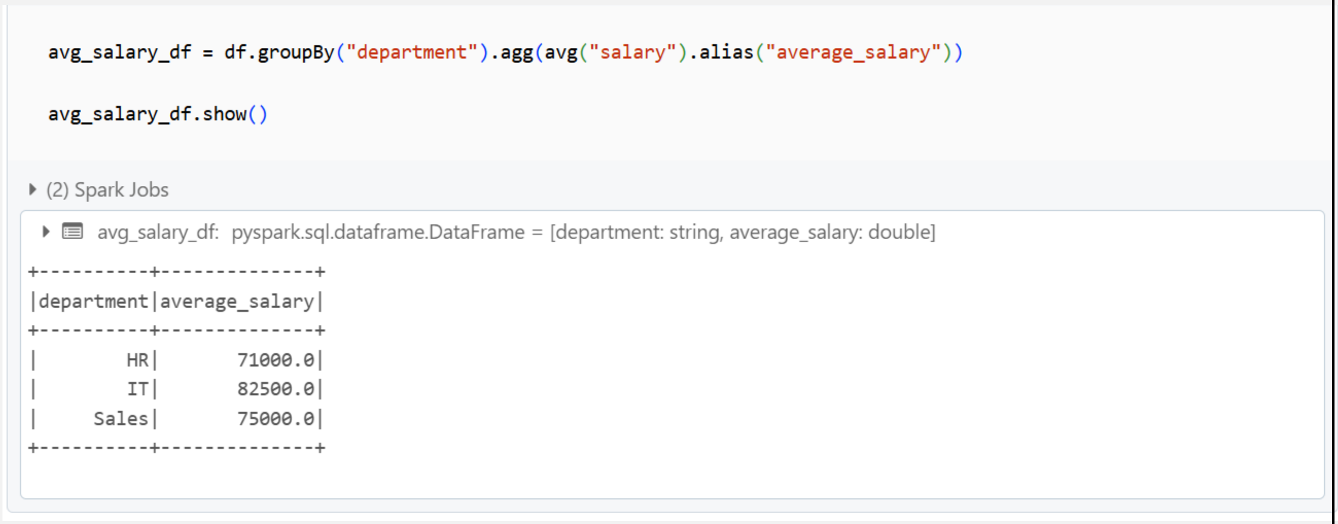
```
from pyspark.sql.functions import *

from pyspark.sql.types import *

# Define the schema with hire_date as StringType

schema = StructType([

   StructField("employee_id", IntegerType(), False),

   StructField("name", StringType(), False),

   StructField("department", StringType(), False),

   StructField("salary", IntegerType(), False),

   StructField("hire_date", StringType(), False)  # As StringType

])


# Define data with dates already formatted as dd-MM-yyyy

data = [

   (1, "Amit", "HR", 70000, "15-01-2019"),

   (2, "Rajesh", "IT", 80000, "22-03-2018"),

   (3, "Neeta", "IT", 85000, "30-07-2017"),

   (4, "Anjali", "Sales", 75000, "05-11-2020"),

   (5, "Ravi", "HR", 72000, "14-06-2021")

]


# Create the DataFrame

df = spark.createDataFrame(data, schema)

# Show the DataFrame

df.show()
```



```
df:  pyspark.sql.dataframe.DataFrame = [employee_id: integer, name: string ... 3 more fields]
+-----------+------+----------+------+----------+
|employee_id|  name|department|salary| hire_date|
+-----------+------+----------+------+----------+
|          1|  Amit|        HR| 70000|15-01-2019|
|          2|Rajesh|        IT| 80000|22-03-2018|
|          3| Neeta|        IT| 85000|30-07-2017|
|          4|Anjali|     Sales| 75000|05-11-2020|
|          5|  Ravi|        HR| 72000|14-06-2021|
+-----------+------+----------+------+----------+
```

Question 1: What is the average salary by department?

Solution:

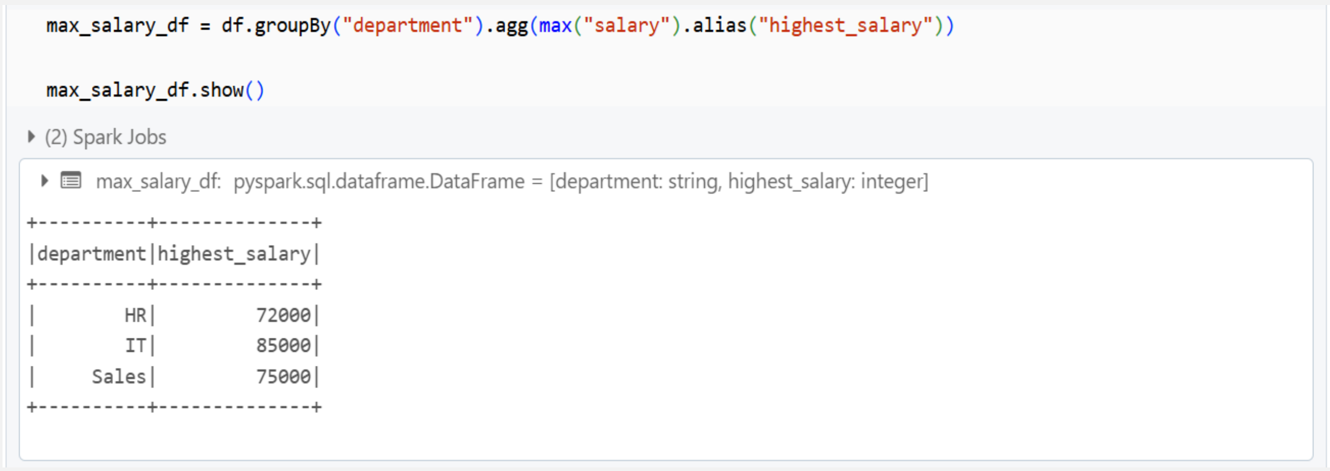


```python
avg_salary_df = df.groupBy("department").agg(avg("salary").alias("average_salary"))

avg_salary_df.show()
```

```
(2) Spark Jobs
avg_salary_df:  pyspark.sql.dataframe.DataFrame = [department: string, average_salary: double]
+----------+--------------+
|department|average_salary|
+----------+--------------+
|        HR|       71000.0|
|        IT|       82500.0|
|     Sales|       75000.0|
+----------+--------------+
```

Question 2: Find the highest salary in each department.

Solution:

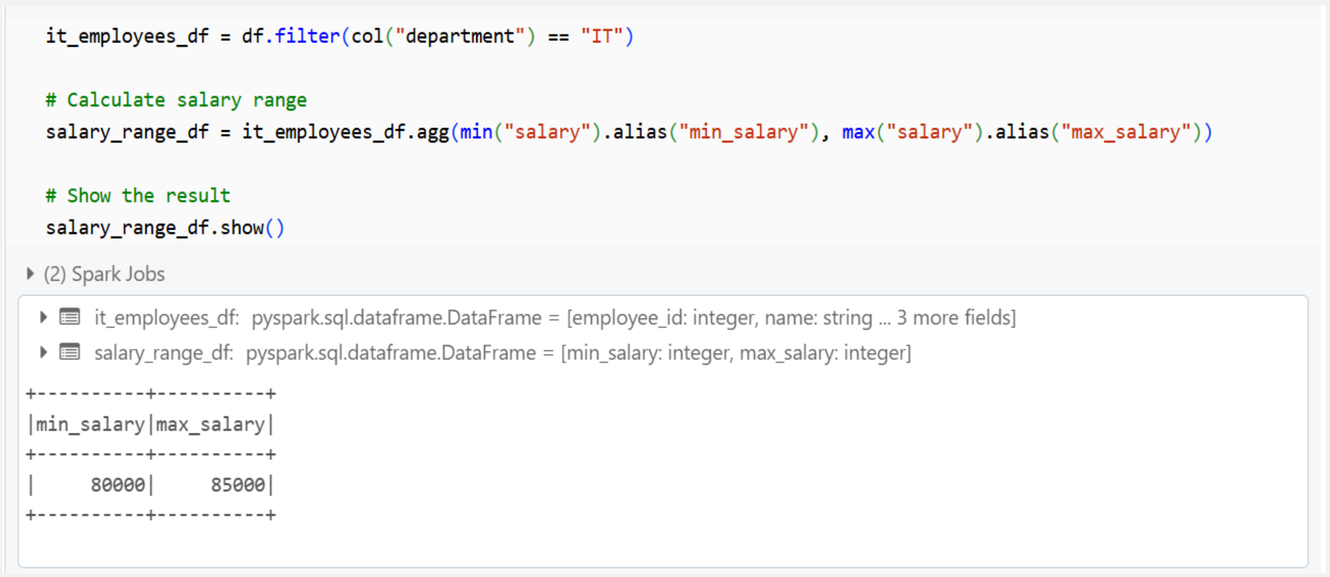


```
max_salary_df = df.groupBy("department").agg(max("salary").alias("highest_salary"))

max_salary_df.show()
```

```
(2) Spark Jobs
max_salary_df:  pyspark.sql.dataframe.DataFrame = [department: string, highest_salary: integer]
+----------+--------------+
|department|highest_salary|
+----------+--------------+
|        HR|         72000|
|        IT|         85000|
|     Sales|         75000|
+----------+--------------+
```

Question 3: Calculate the salary range (min and max) for employees in the IT department.

Solution:

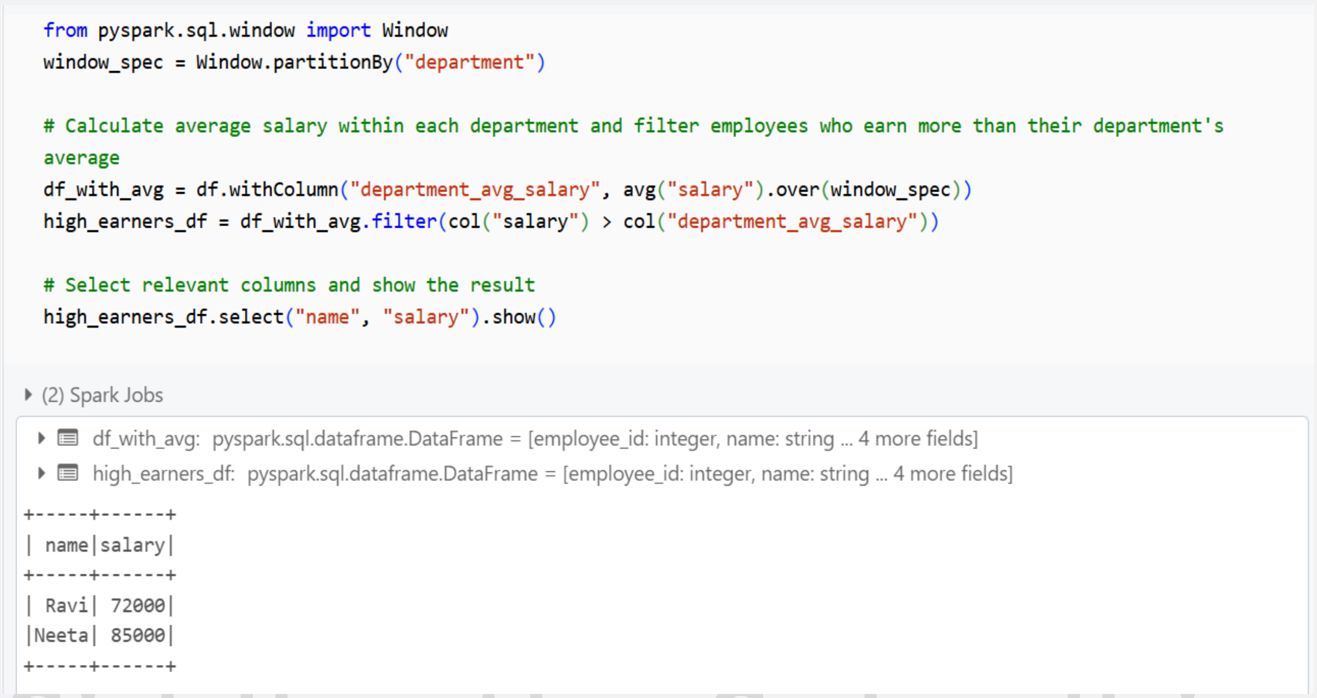


```
it_employees_df = df.filter(col("department") == "IT")

# Calculate salary range
salary_range_df = it_employees_df.agg(min("salary").alias("min_salary"), max("salary").alias("max_salary"))

# Show the result
salary_range_df.show()
```

```
(2) Spark Jobs
it_employees_df:  pyspark.sql.dataframe.DataFrame = [employee_id: integer, name: string ... 3 more fields]
salary_range_df:  pyspark.sql.dataframe.DataFrame = [min_salary: integer, max_salary: integer]
+----------+----------+
|min_salary|max_salary|
+----------+----------+
|     80000|     85000|
+----------+----------+
```

Question 4: List the names and salaries of employees who earn more than the average salary of their department.

Solution:



```
from pyspark.sql.window import Window
window_spec = Window.partitionBy("department")

# Calculate average salary within each department and filter employees who earn more than their department's
average
df_with_avg = df.withColumn("department_avg_salary", avg("salary").over(window_spec))
high_earners_df = df_with_avg.filter(col("salary") > col("department_avg_salary"))

# Select relevant columns and show the result
high_earners_df.select("name", "salary").show()
```

(2) Spark Jobs

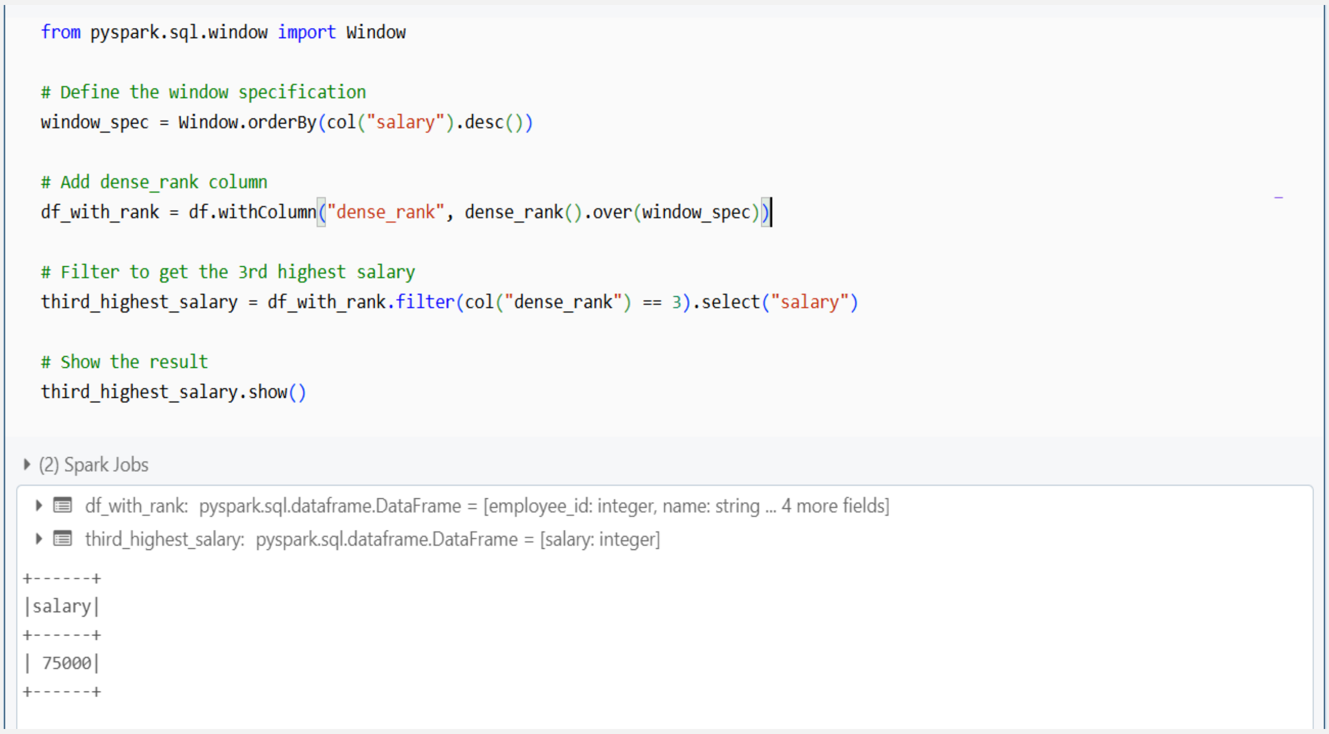
df_with_avg: pyspark.sql.dataframe.DataFrame = [employee_id: integer, name: string ... 4 more fields]

high_earners_df: pyspark.sql.dataframe.DataFrame = [employee_id: integer, name: string ... 4 more fields]

```
+-----+------+
| name|salary|
+-----+------+
| Ravi| 72000|
|Neeta| 85000|
+-----+------+
```

Question 5: Find the employee with the 3rd highest salary.

Solution:



```
from pyspark.sql.window import Window

# Define the window specification
window_spec = Window.orderBy(col("salary").desc())

# Add dense_rank column
df_with_rank = df.withColumn("dense_rank", dense_rank().over(window_spec))

# Filter to get the 3rd highest salary
third_highest_salary = df_with_rank.filter(col("dense_rank") == 3).select("salary")

# Show the result
third_highest_salary.show()
```

(2) Spark Jobs

df_with_rank: pyspark.sql.dataframe.DataFrame = [employee_id: integer, name: string ... 4 more fields]

third_highest_salary: pyspark.sql.dataframe.DataFrame = [salary: integer]

```
+------+
|salary|
+------+
| 75000|
+------+
```