Most Asked Spark Interview Questions

12LPA - 20LPA

Question 1:

1. Student Grade Classification

Problem:

You have a Data Frame of students with the following columns: student_id, name, score, and subject.

Create a new column grade based on the score:

o 'A' if score >= 90

o 'B' if 80 <= score < 90

o 'C' if 70 <= score < 80

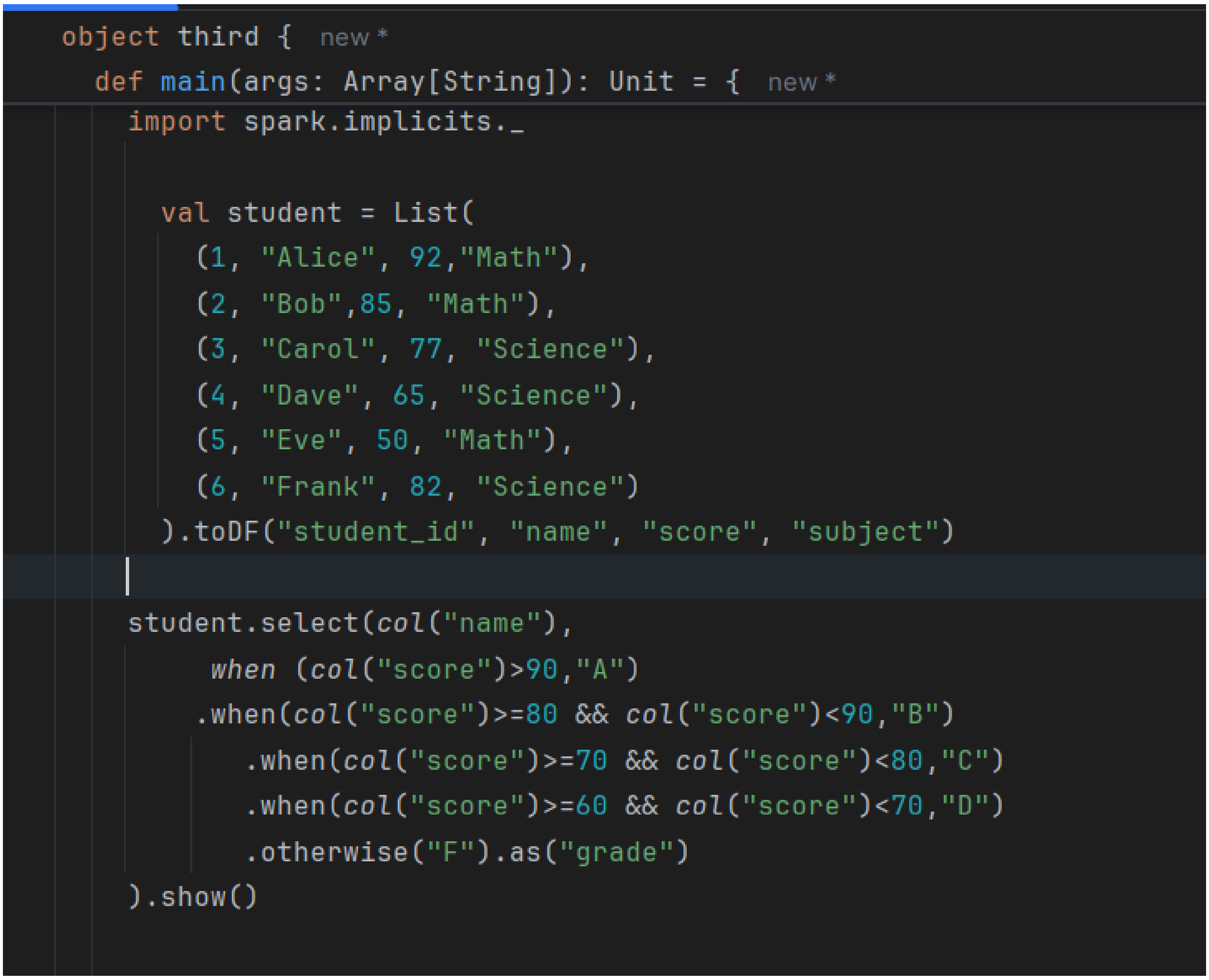
o 'D' if 60 <= score < 70

o 'F' if score < 60

Data Set

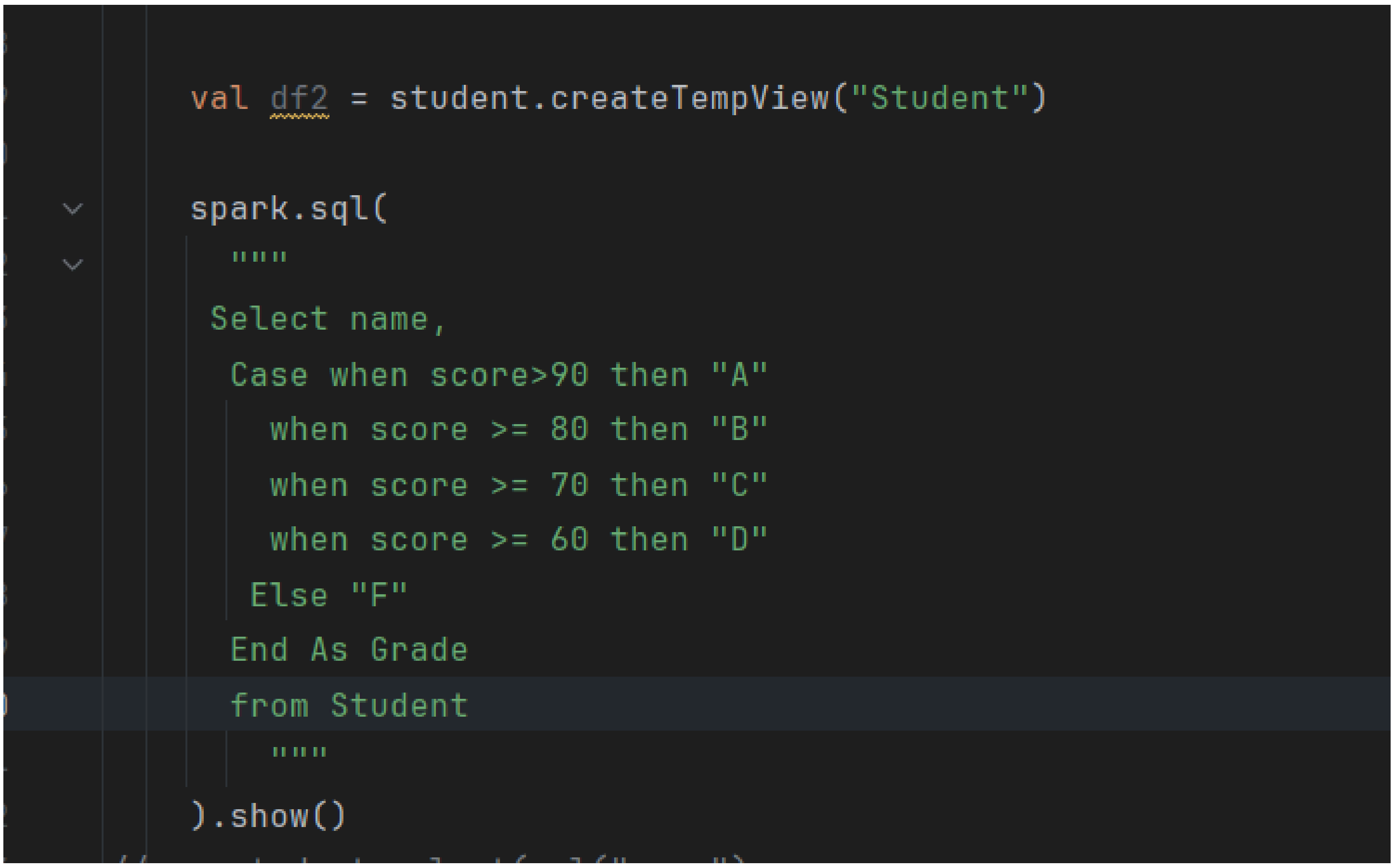
student_id	name	score	subject
1	Alice	92	Math
2	Bob	85	Math
3	Carol	77	Science
4	Dave	65	Science
5	Eve	50	Math
6	Frank	82	Science

Scala Spark



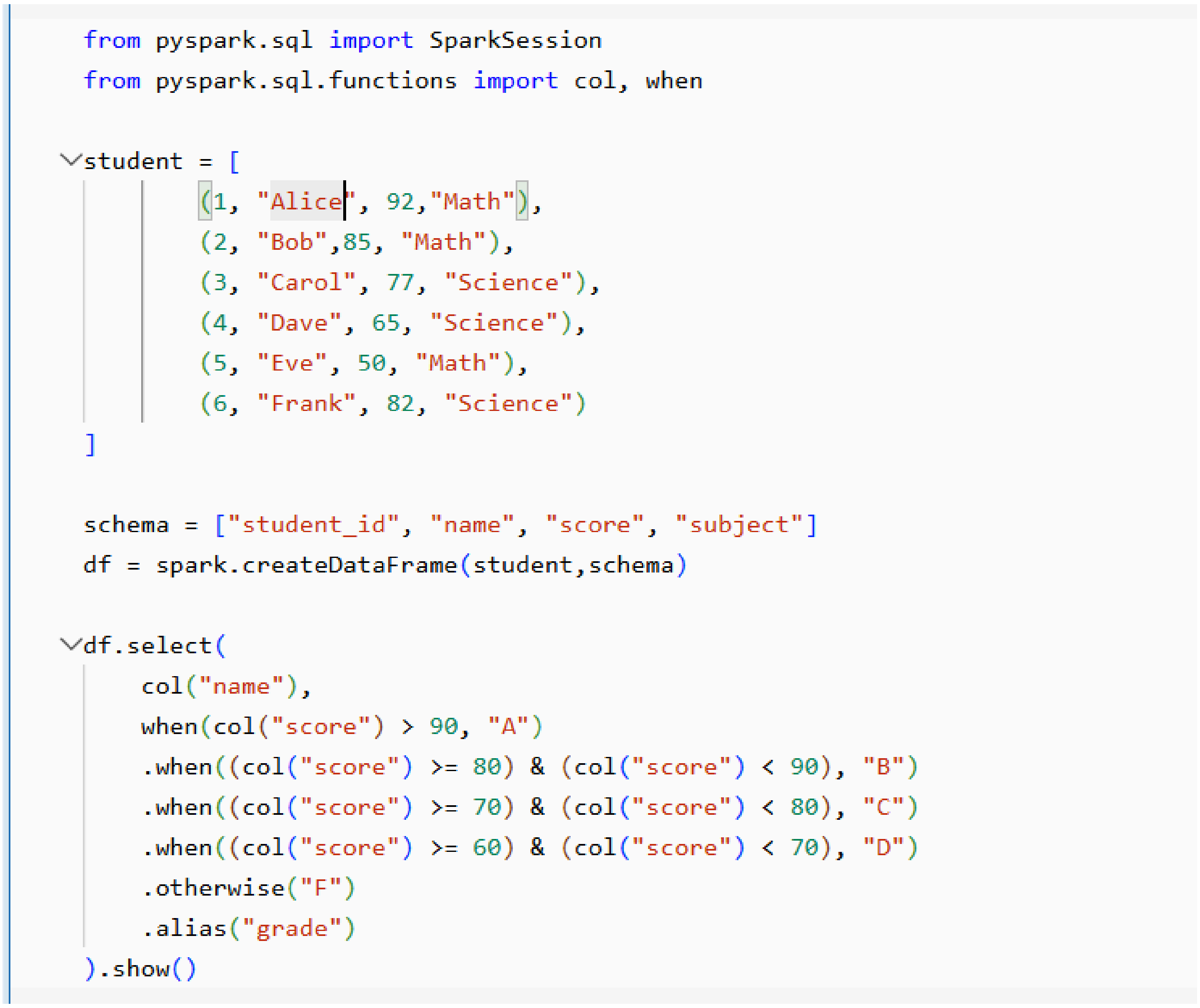
```
object third {  new *
  def main(args: Array[String]): Unit = {  new *
    import spark.implicits._

      val student = List(
        (1, "Alice", 92,"Math"),
        (2, "Bob",85, "Math"),
        (3, "Carol", 77, "Science"),
        (4, "Dave", 65, "Science"),
        (5, "Eve", 50, "Math"),
        (6, "Frank", 82, "Science")
      ).toDF("student_id", "name", "score", "subject")

    student.select(col("name"),
         when (col("score")>90,"A")
        .when(col("score")>=80 && col("score")<90,"B")
           .when(col("score")>=70 && col("score")<80,"C")
           .when(col("score")>=60 && col("score")<70,"D")
           .otherwise("F").as("grade")
    ).show()
```

Spark - SQL



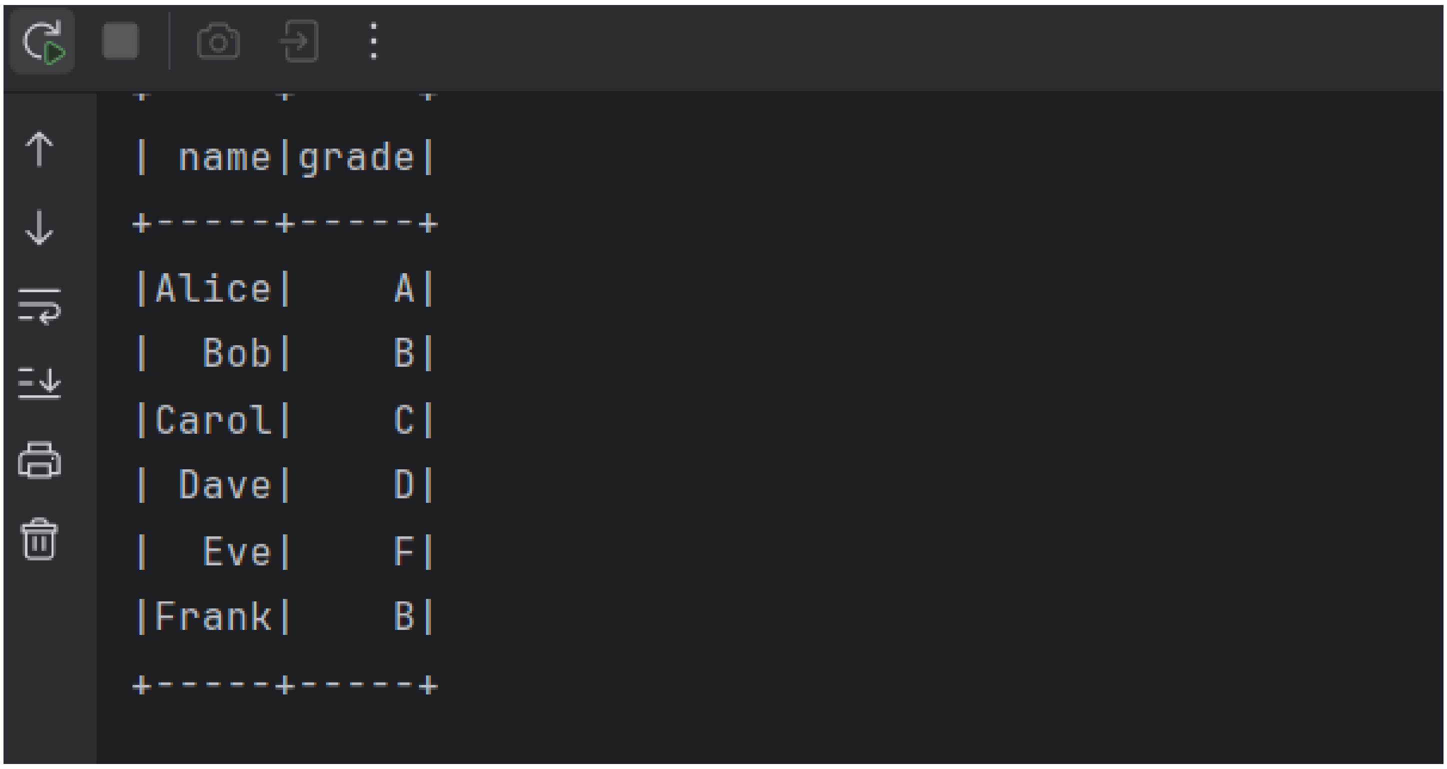
```scala
val df2 = student.createTempView("Student")

spark.sql(
  """
 Select name,
  Case when score>90 then "A"
    when score >= 80 then "B"
    when score >= 70 then "C"
    when score >= 60 then "D"
   Else "F"
  End As Grade
  from Student
    """
).show()
```

PySpark



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

student = [
        (1, "Alice", 92,"Math"),
        (2, "Bob",85, "Math"),
        (3, "Carol", 77, "Science"),
        (4, "Dave", 65, "Science"),
        (5, "Eve", 50, "Math"),
        (6, "Frank", 82, "Science")
]

schema = ["student_id", "name", "score", "subject"]
df = spark.createDataFrame(student,schema)

df.select(
    col("name"),
    when(col("score") > 90, "A")
    .when((col("score") >= 80) & (col("score") < 90), "B")
    .when((col("score") >= 70) & (col("score") < 80), "C")
    .when((col("score") >= 60) & (col("score") < 70), "D")
    .otherwise("F")
    .alias("grade")
).show()
```

Output -



```
| name|grade|
+-----+-----+
|Alice|    A|
|  Bob|    B|
|Carol|    C|
| Dave|    D|
|  Eve|    F|
|Frank|    B|
+-----+-----+
```

Question 2:

You have a DataFrame employees with columns: employee_id, name, age, and salary.

 Create a new column age_group based on age:

o 'Young' if age < 30

o 'Mid' if 30 <= age <= 50

o 'Senior' if age > 50

 Create a new column salary_range based on salary:

o 'High' if salary > 100000

o 'Medium' if 50000 <= salary <= 100000

o 'Low' if salary < 50000

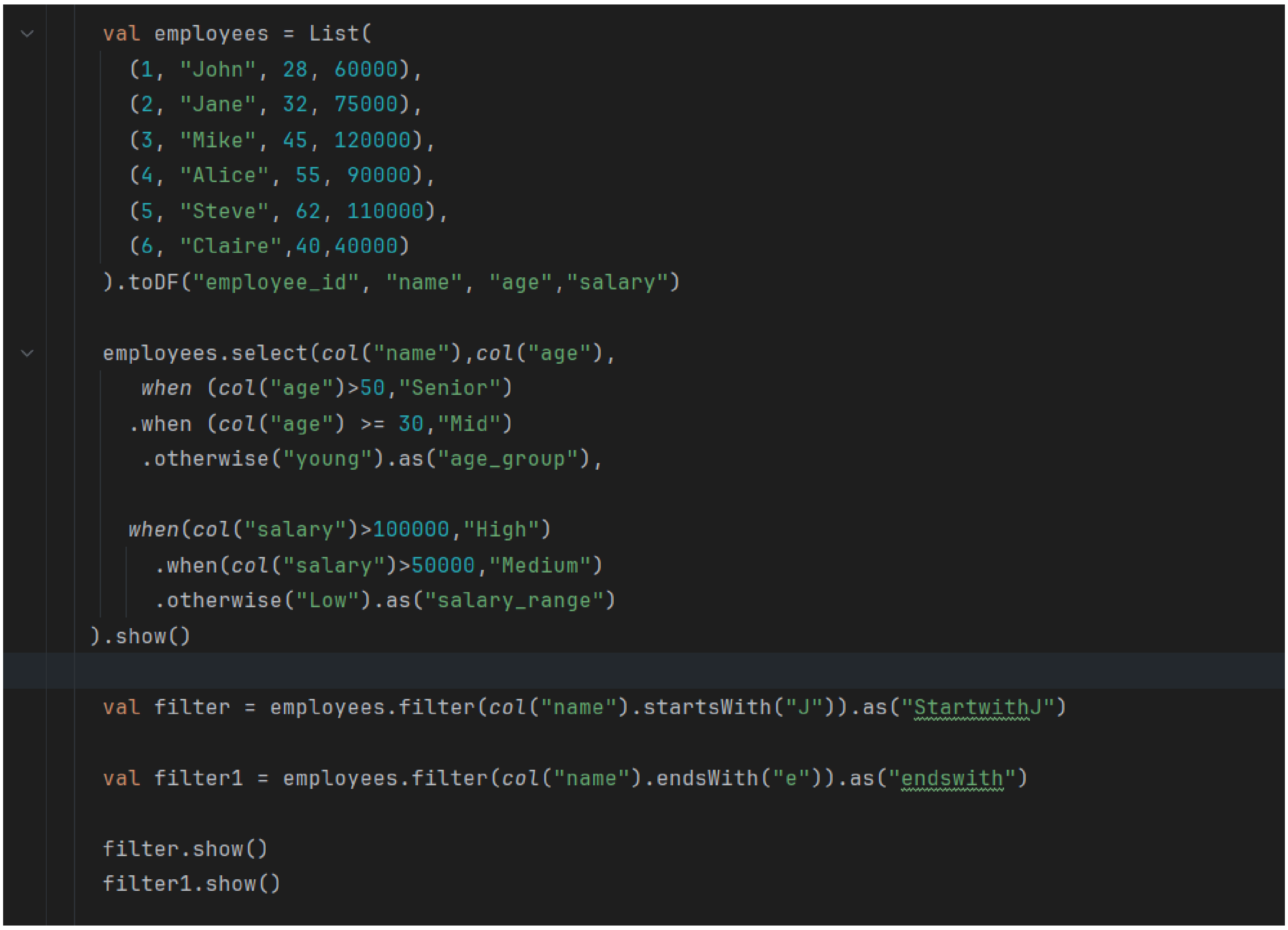
 Filter employees whose name starts with 'J'.

 Filter employees whose name ends with 'e'.

Data Set -

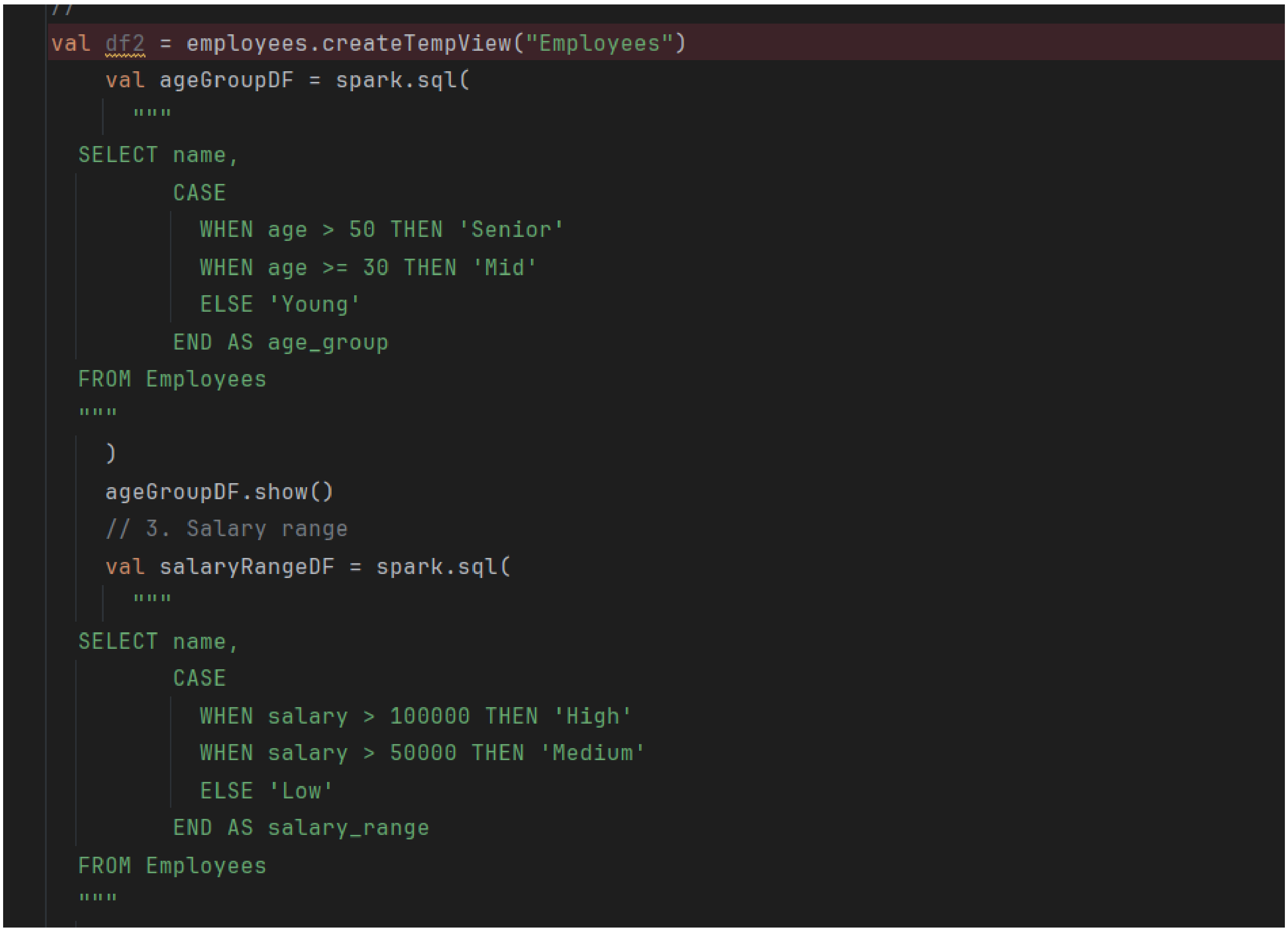
```
data = [

  (1, "John", 28, 60000),

  (2, "Jane", 32, 75000),

  (3, "Mike", 45, 120000),

  (4, "Alice", 55, 90000),

  (5, "Steve", 62, 110000),

  (6, "Claire", 40, 40000)

]
```

Scala Spark -

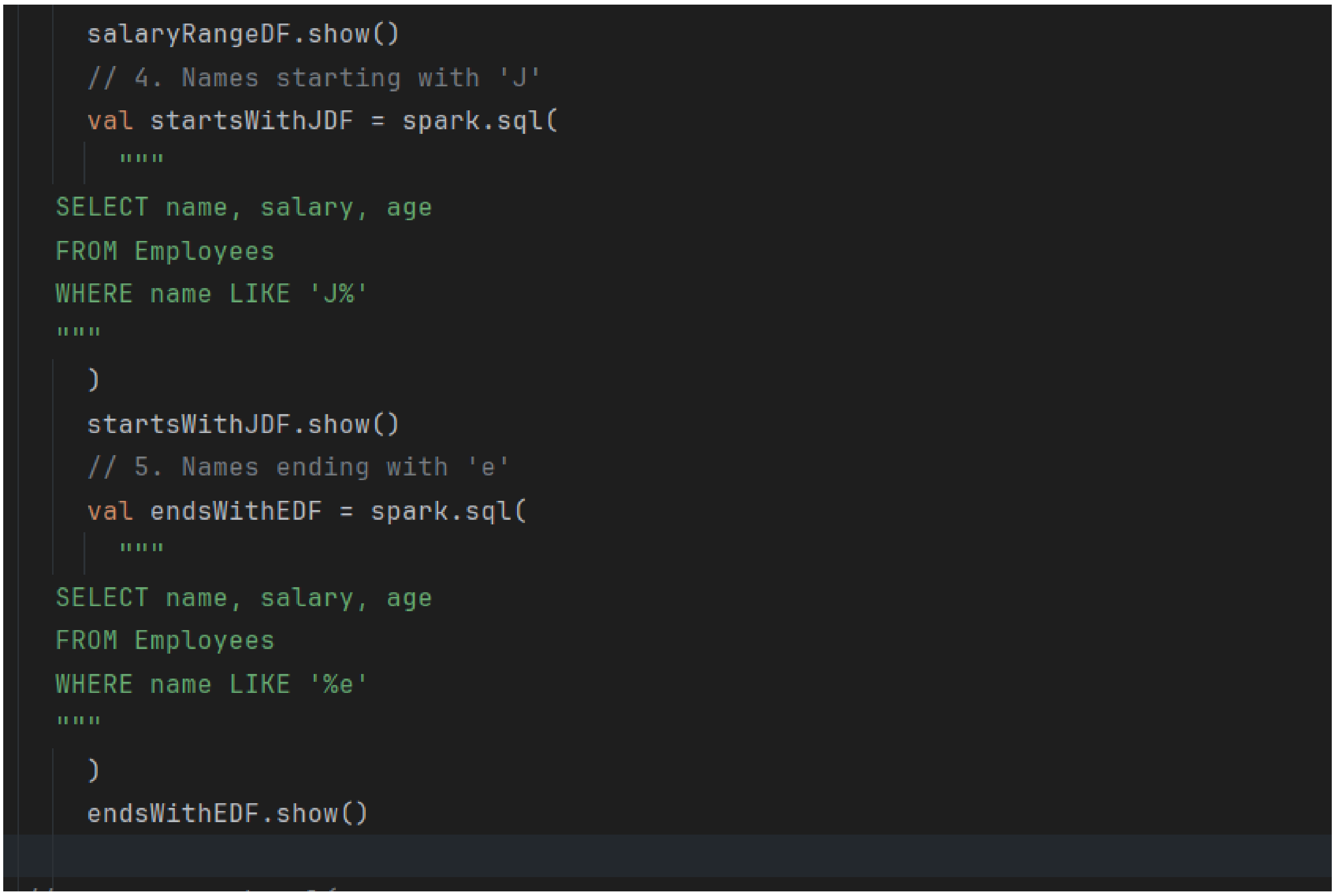


```
val employees = List(
  (1, "John", 28, 60000),
  (2, "Jane", 32, 75000),
  (3, "Mike", 45, 120000),
  (4, "Alice", 55, 90000),
  (5, "Steve", 62, 110000),
  (6, "Claire",40,40000)
).toDF("employee_id", "name", "age","salary")

employees.select(col("name"),col("age"),
   when (col("age")>50,"Senior")
  .when (col("age") >= 30,"Mid")
   .otherwise("young").as("age_group"),

  when(col("salary")>100000,"High")
    .when(col("salary")>50000,"Medium")
    .otherwise("Low").as("salary_range")
).show()

val filter = employees.filter(col("name").startsWith("J")).as("StartwithJ")

val filter1 = employees.filter(col("name").endsWith("e")).as("endswith")

filter.show()
filter1.show()
```

Spark - SQL

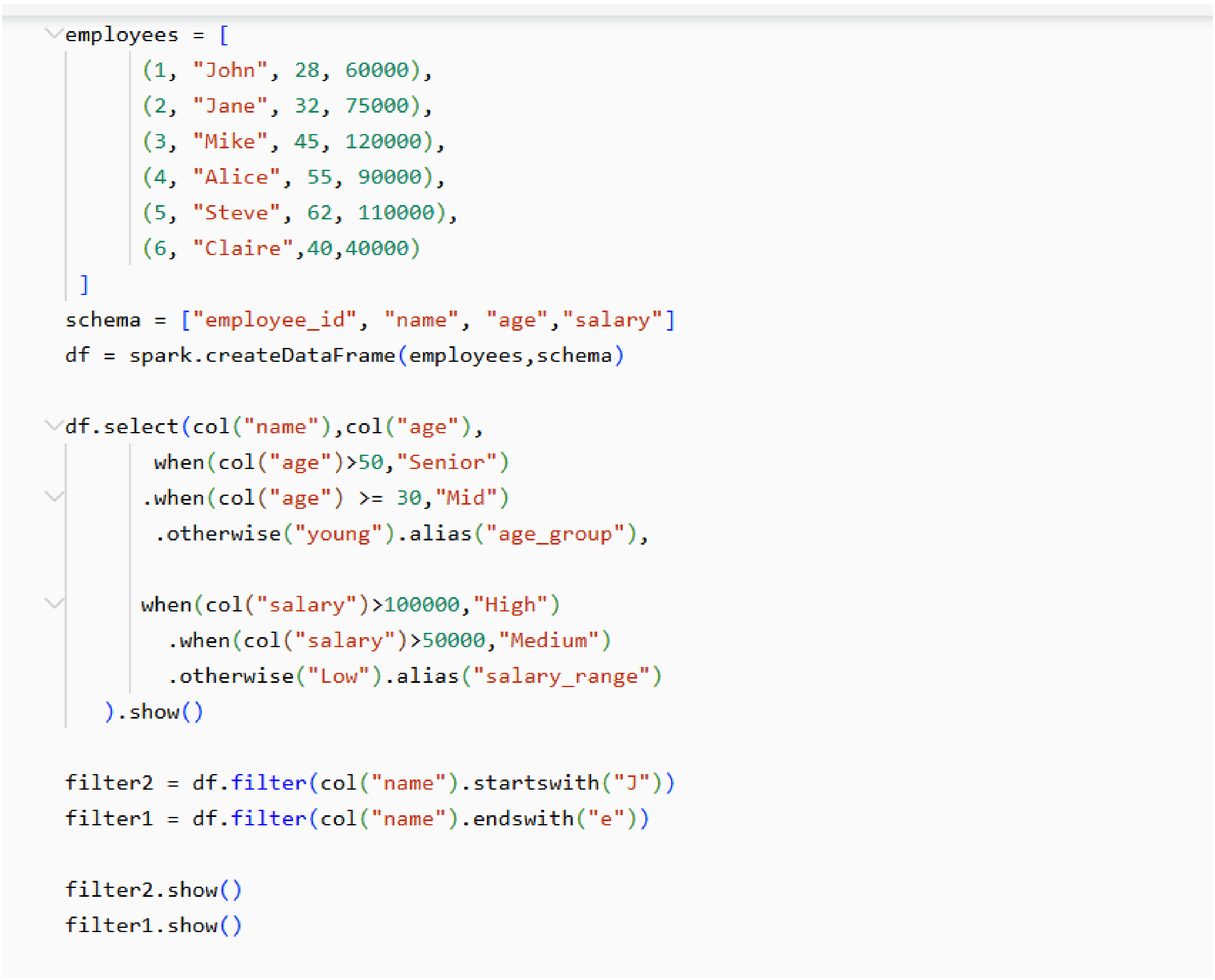


```
val df2 = employees.createTempView("Employees")
    val ageGroupDF = spark.sql(
      """
  SELECT name,
         CASE
           WHEN age > 50 THEN 'Senior'
           WHEN age >= 30 THEN 'Mid'
           ELSE 'Young'
         END AS age_group
  FROM Employees
  """
    )
    ageGroupDF.show()
    // 3. Salary range
    val salaryRangeDF = spark.sql(
      """
  SELECT name,
         CASE
           WHEN salary > 100000 THEN 'High'
           WHEN salary > 50000 THEN 'Medium'
           ELSE 'Low'
         END AS salary_range
  FROM Employees
  """
```



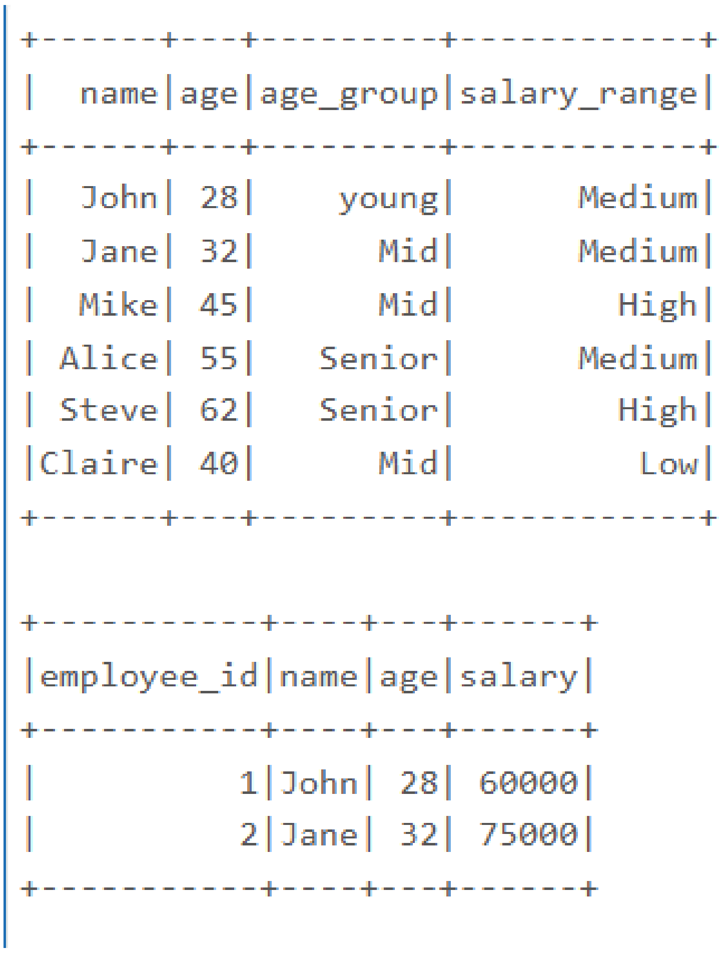
```
    salaryRangeDF.show()
    // 4. Names starting with 'J'
    val startsWithJDF = spark.sql(
      """
SELECT name, salary, age
FROM Employees
WHERE name LIKE 'J%'
"""
    )
    startsWithJDF.show()
    // 5. Names ending with 'e'
    val endsWithEDF = spark.sql(
      """
SELECT name, salary, age
FROM Employees
WHERE name LIKE '%e'
"""
    )
    endsWithEDF.show()
```

PySpark -

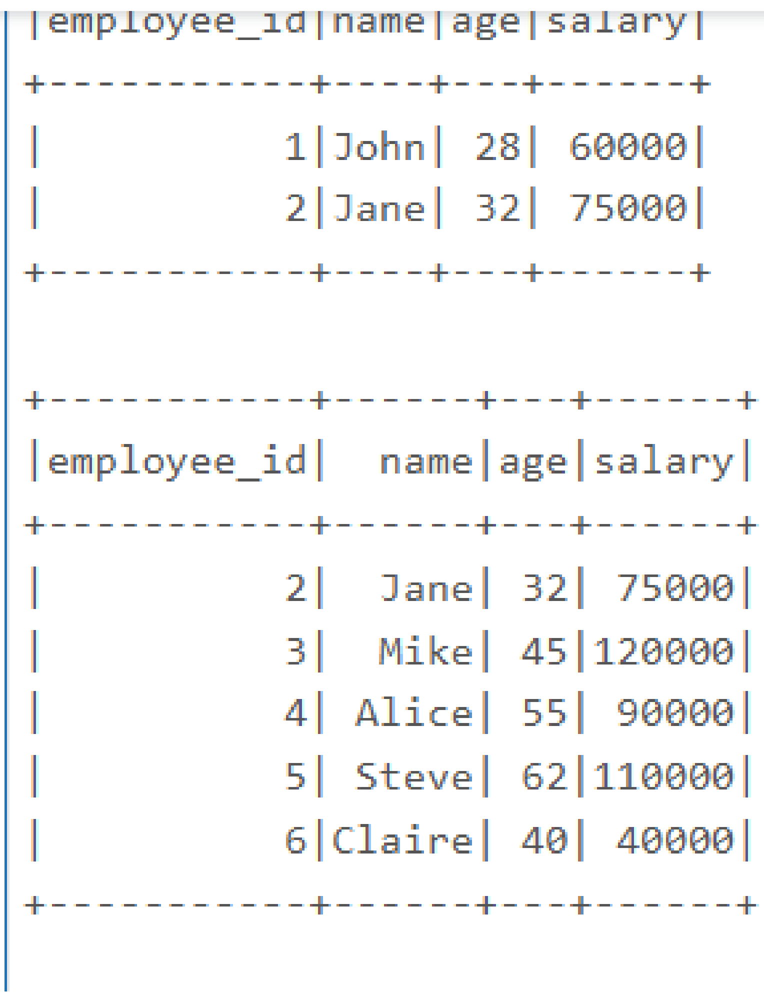


```
employees = [
        (1, "John", 28, 60000),
        (2, "Jane", 32, 75000),
        (3, "Mike", 45, 120000),
        (4, "Alice", 55, 90000),
        (5, "Steve", 62, 110000),
        (6, "Claire",40,40000)
 ]
schema = ["employee_id", "name", "age","salary"]
df = spark.createDataFrame(employees,schema)

df.select(col("name"),col("age"),
         when(col("age")>50,"Senior")
        .when(col("age") >= 30,"Mid")
         .otherwise("young").alias("age_group"),

        when(col("salary")>100000,"High")
          .when(col("salary")>50000,"Medium")
          .otherwise("Low").alias("salary_range")
    ).show()

filter2 = df.filter(col("name").startswith("J"))
filter1 = df.filter(col("name").endswith("e"))

filter2.show()
filter1.show()
```

Output -



```
+------+---+---------+------------+
|  name|age|age_group|salary_range|
+------+---+---------+------------+
|  John| 28|    young|      Medium|
|  Jane| 32|      Mid|      Medium|
|  Mike| 45|      Mid|        High|
| Alice| 55|   Senior|      Medium|
| Steve| 62|   Senior|        High|
|Claire| 40|      Mid|         Low|
+------+---+---------+------------+


+-----------+----+---+------+
|employee_id|name|age|salary|
+-----------+----+---+------+
|          1|John| 28| 60000|
|          2|Jane| 32| 75000|
+-----------+----+---+------+
```



```
|employee_id|name|age|salary|
+-----------+----+---+------+
|          1|John| 28| 60000|
|          2|Jane| 32| 75000|
+-----------+----+---+------+


+-----------+------+---+------+
|employee_id|  name|age|salary|
+-----------+------+---+------+
|          2|  Jane| 32| 75000|
|          3|  Mike| 45|120000|
|          4| Alice| 55| 90000|
|          5| Steve| 62|110000|
|          6|Claire| 40| 40000|
+-----------+------+---+------+
```

Question 3:

You have a DataFrame purchase_history with columns: purchase_id, customer_id, purchase_amount, and purchase_date.

 Create a new column purchase_category based on purchase_amount:

o 'Large' if purchase_amount > 2000

o 'Medium' if 1000 <= purchase_amount <= 2000

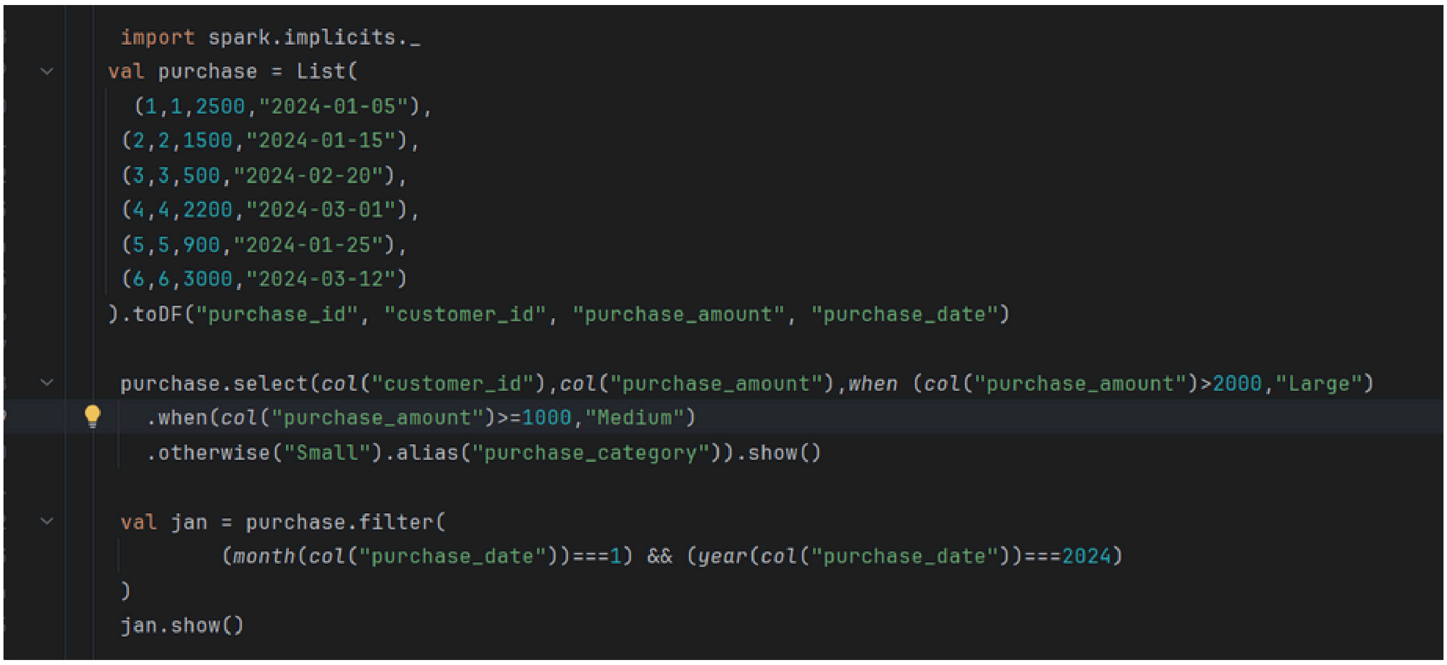
o 'Small' if purchase_amount < 1000

 Filter purchases that occurred in 'January 2024'

Data Set -

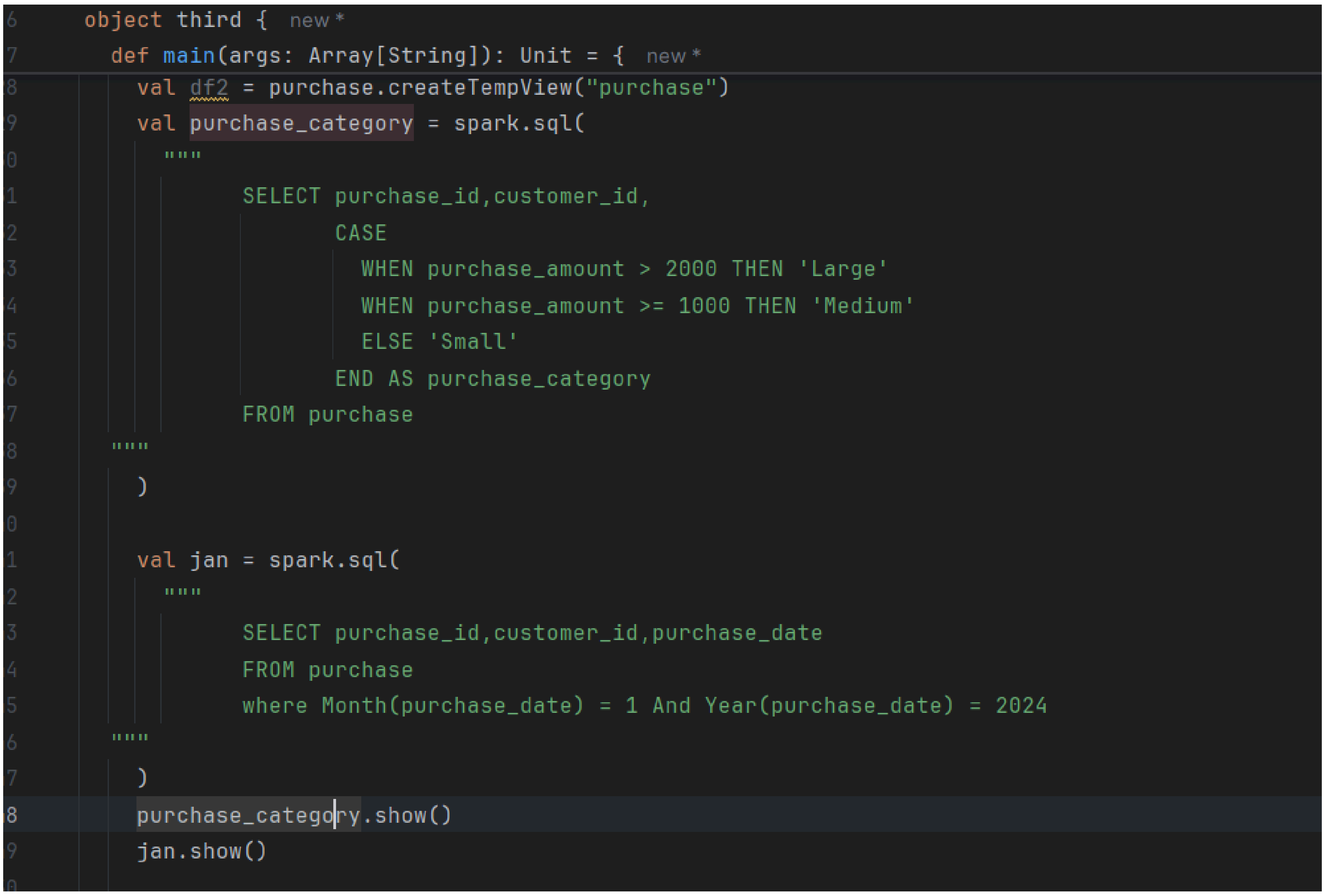
```
[(1,1,2500,"2024-01-05"),

(2,2,1500,"2024-01-15"),

(3,3,500,"2024-02-20"),

(4,4,2200,"2024-03-01"),

(5,5,900,"2024-01-25"),

(6,6,3000,"2024-03-12")]
```

Scala Spark



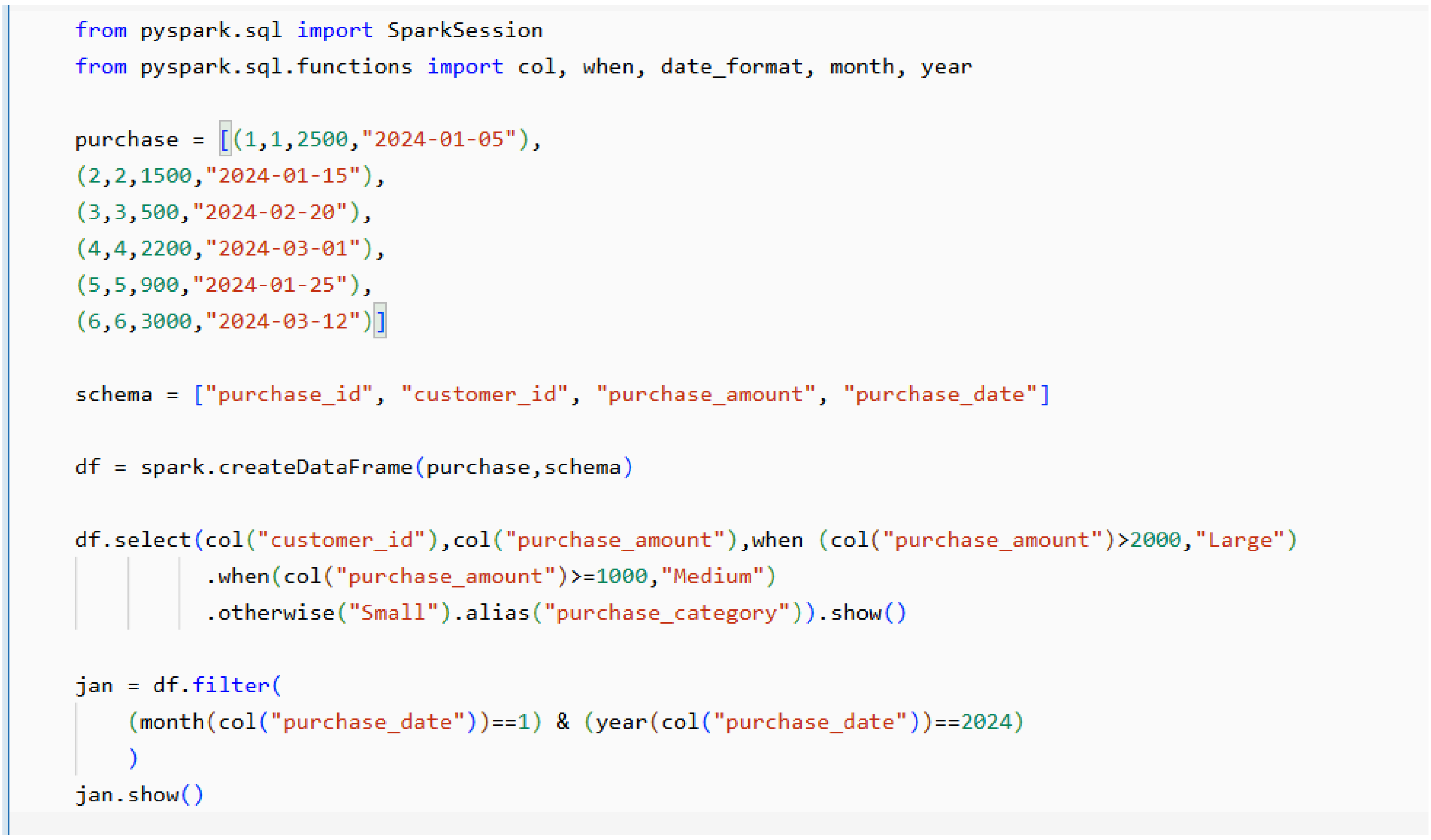
```scala
 import spark.implicits._
val purchase = List(
  (1,1,2500,"2024-01-05"),
 (2,2,1500,"2024-01-15"),
 (3,3,500,"2024-02-20"),
 (4,4,2200,"2024-03-01"),
 (5,5,900,"2024-01-25"),
 (6,6,3000,"2024-03-12")
).toDF("purchase_id", "customer_id", "purchase_amount", "purchase_date")

 purchase.select(col("customer_id"),col("purchase_amount"),when (col("purchase_amount")>2000,"Large")
   .when(col("purchase_amount")>=1000,"Medium")
   .otherwise("Small").alias("purchase_category")).show()

 val jan = purchase.filter(
         (month(col("purchase_date"))===1) && (year(col("purchase_date"))===2024)
 )
 jan.show()
```

Spark - SQL



```scala
object third {  new *
  def main(args: Array[String]): Unit = {  new *
    val df2 = purchase.createTempView("purchase")
    val purchase_category = spark.sql(
      """
          SELECT purchase_id,customer_id,
                 CASE
                   WHEN purchase_amount > 2000 THEN 'Large'
                   WHEN purchase_amount >= 1000 THEN 'Medium'
                   ELSE 'Small'
                 END AS purchase_category
          FROM purchase
  """
    )

    val jan = spark.sql(
      """
          SELECT purchase_id,customer_id,purchase_date
          FROM purchase
          where Month(purchase_date) = 1 And Year(purchase_date) = 2024
  """
    )
    purchase_category.show()
    jan.show()
```

PySpark

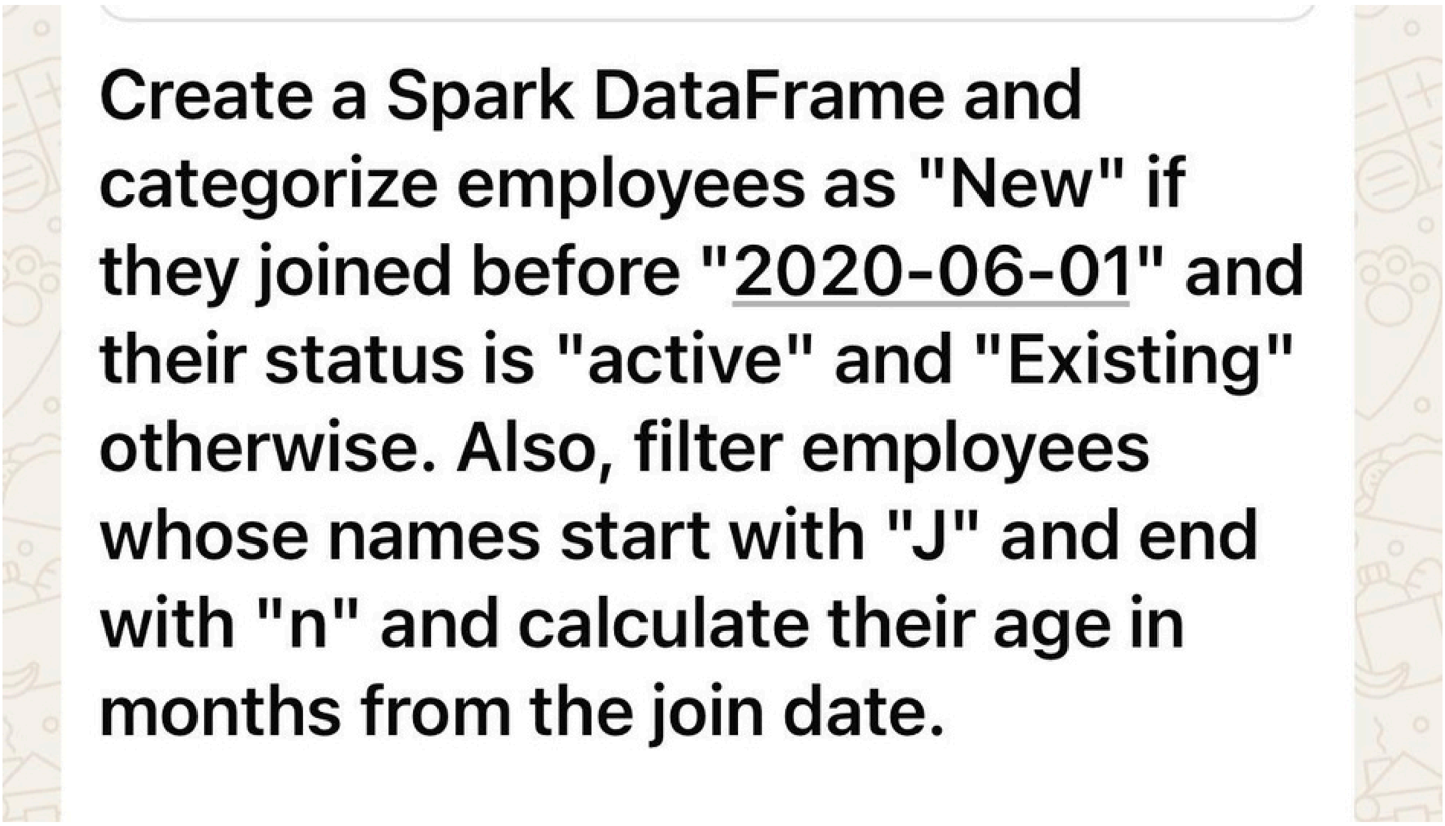


```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, date_format, month, year

purchase = [(1,1,2500,"2024-01-05"),
(2,2,1500,"2024-01-15"),
(3,3,500,"2024-02-20"),
(4,4,2200,"2024-03-01"),
(5,5,900,"2024-01-25"),
(6,6,3000,"2024-03-12")]

schema = ["purchase_id", "customer_id", "purchase_amount", "purchase_date"]

df = spark.createDataFrame(purchase,schema)

df.select(col("customer_id"),col("purchase_amount"),when (col("purchase_amount")>2000,"Large")
          .when(col("purchase_amount")>=1000,"Medium")
          .otherwise("Small").alias("purchase_category")).show()

jan = df.filter(
    (month(col("purchase_date"))==1) & (year(col("purchase_date"))==2024)
    )
jan.show()
```

Output -

```
+-----------+---------------+----------------+
|customer_id|purchase_amount|purchase_category|
+-----------+---------------+----------------+
|          1|           2500|           Large|
|          2|           1500|          Medium|
|          3|            500|           Small|
|          4|           2200|           Large|
|          5|            900|           Small|
|          6|           3000|           Large|
+-----------+---------------+----------------+

+-----------+-----------+---------------+------------+
|purchase_id|customer_id|purchase_amount|purchase_date|
+-----------+-----------+---------------+------------+
|          1|          1|           2500|  2024-01-05|
|          2|          2|           1500|  2024-01-15|
|          5|          5|            900|  2024-01-25|
+-----------+-----------+---------------+------------+
```

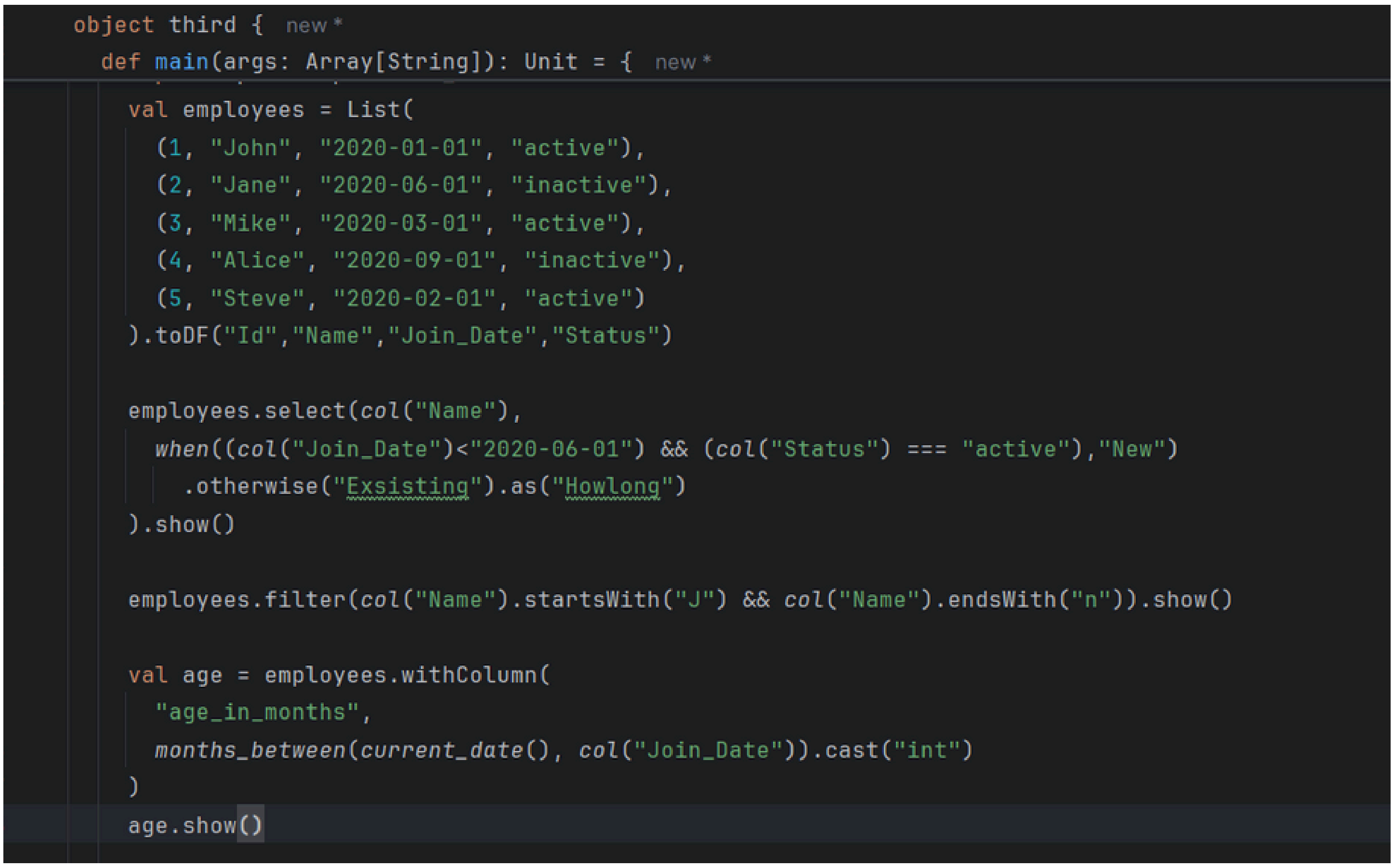
Question 4:


Create a Spark DataFrame and categorize employees as "New" if they joined before "2020-06-01" and their status is "active" and "Existing" otherwise. Also, filter employees whose names start with "J" and end with "n" and calculate their age in months from the join date.

Data set -

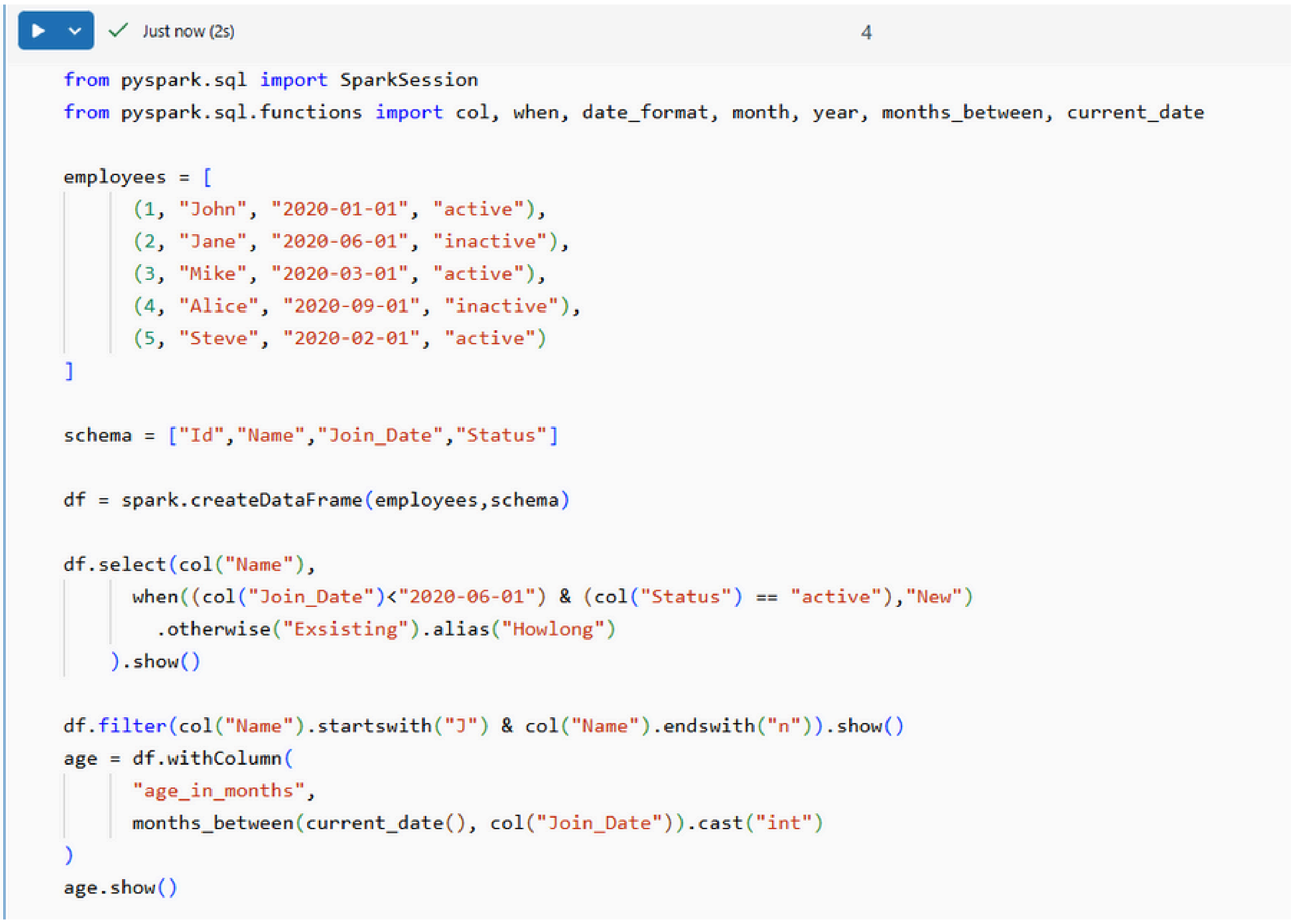
```
val employees = List(
  (1, "John", "2020-01-01", "active"),
  (2, "Jane", "2020-06-01", "inactive"),
  (3, "Mike", "2020-03-01", "active"),
  (4, "Alice", "2020-09-01", "inactive"),
  (5, "Steve", "2020-02-01", "active")
)
```

Scala Spark -



```
object third {  new *
  def main(args: Array[String]): Unit = {  new *
    val employees = List(
      (1, "John", "2020-01-01", "active"),
      (2, "Jane", "2020-06-01", "inactive"),
      (3, "Mike", "2020-03-01", "active"),
      (4, "Alice", "2020-09-01", "inactive"),
      (5, "Steve", "2020-02-01", "active")
    ).toDF("Id","Name","Join_Date","Status")

    employees.select(col("Name"),
      when((col("Join_Date")<"2020-06-01") && (col("Status") === "active"),"New")
        .otherwise("Exsisting").as("Howlong")
    ).show()

    employees.filter(col("Name").startsWith("J") && col("Name").endsWith("n")).show()

    val age = employees.withColumn(
      "age_in_months",
      months_between(current_date(), col("Join_Date")).cast("int")
    )
    age.show()
```

PySpark -



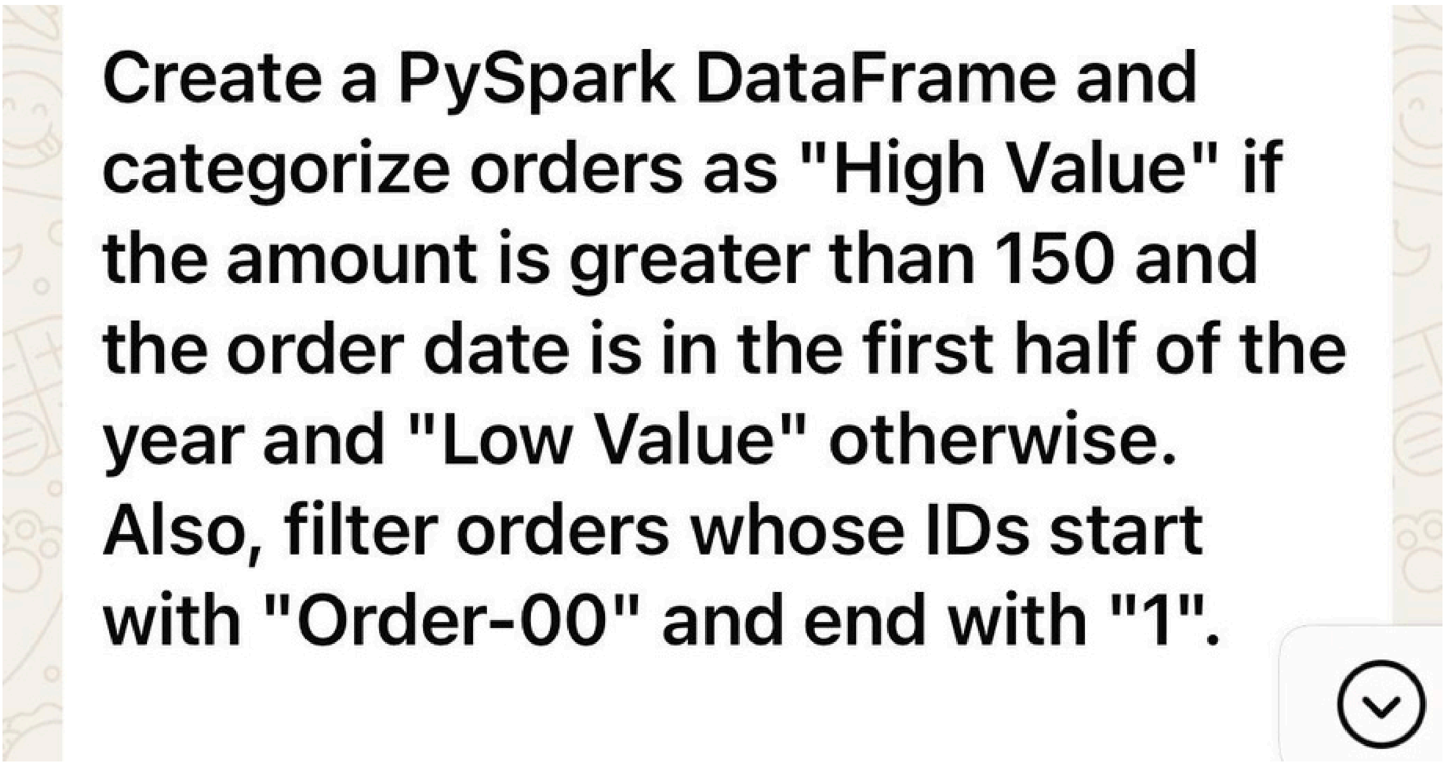
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, date_format, month, year, months_between, current_date

employees = [
      (1, "John", "2020-01-01", "active"),
      (2, "Jane", "2020-06-01", "inactive"),
      (3, "Mike", "2020-03-01", "active"),
      (4, "Alice", "2020-09-01", "inactive"),
      (5, "Steve", "2020-02-01", "active")
]

schema = ["Id","Name","Join_Date","Status"]

df = spark.createDataFrame(employees,schema)

df.select(col("Name"),
      when((col("Join_Date")<"2020-06-01") & (col("Status") == "active"),"New")
        .otherwise("Exsisting").alias("Howlong")
    ).show()

df.filter(col("Name").startswith("J") & col("Name").endswith("n")).show()
age = df.withColumn(
      "age_in_months",
      months_between(current_date(), col("Join_Date")).cast("int")
)
age.show()
```

Output -

```
+-----+---------+
| Name|  Howlong|
+-----+---------+
| John|      New|
| Jane|Exsisting|
| Mike|      New|
|Alice|Exsisting|
|Steve|      New|
+-----+---------+


+---+----+----------+------+
| Id|Name| Join_Date|Status|
+---+----+----------+------+
|  1|John|2020-01-01|active|
+---+----+----------+------+
```

```
+---+-----+----------+--------+-------------+
| Id| Name| Join_Date|  Status|age_in_months|
+---+-----+----------+--------+-------------+
|  1| John|2020-01-01|  active|           65|
|  2| Jane|2020-06-01|inactive|           60|
|  3| Mike|2020-03-01|  active|           63|
|  4|Alice|2020-09-01|inactive|           57|
|  5|Steve|2020-02-01|  active|           64|
+---+-----+----------+--------+-------------+
```

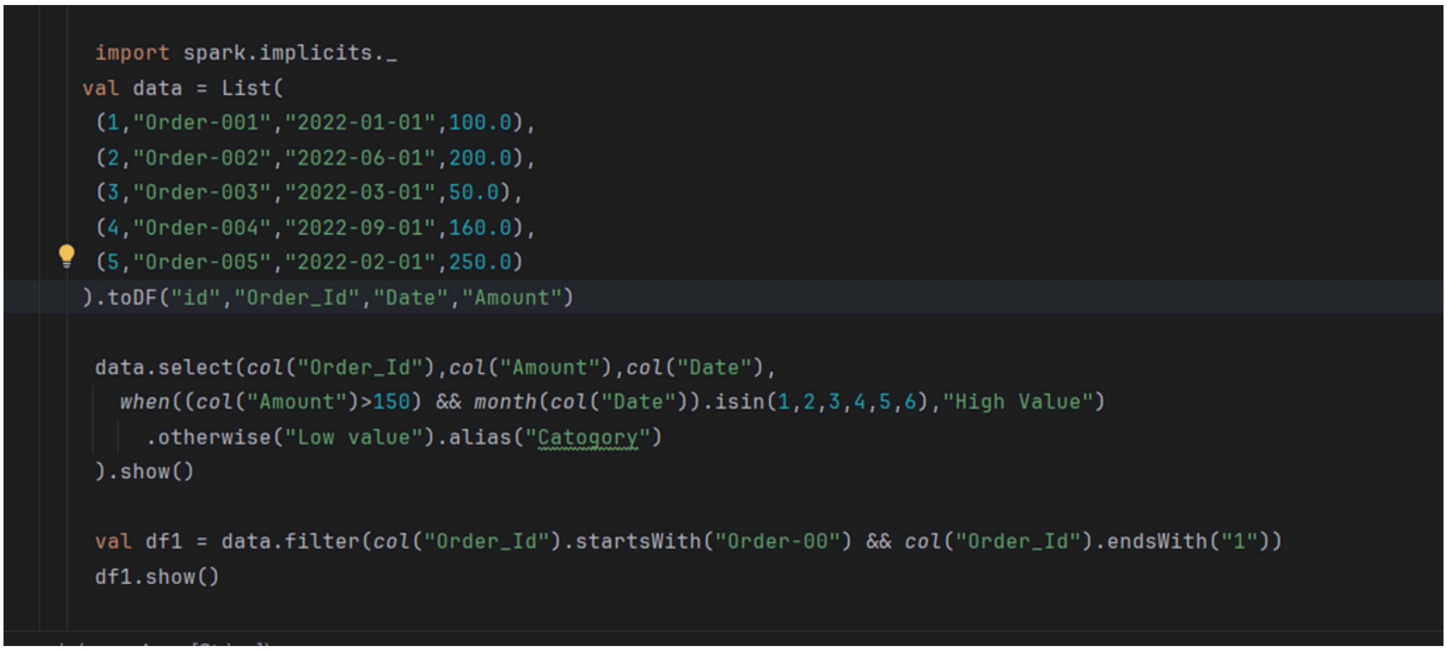
Question 5 -


Create a PySpark DataFrame and categorize orders as "High Value" if the amount is greater than 150 and the order date is in the first half of the year and "Low Value" otherwise. Also, filter orders whose IDs start with "Order-00" and end with "1".

Data Set -

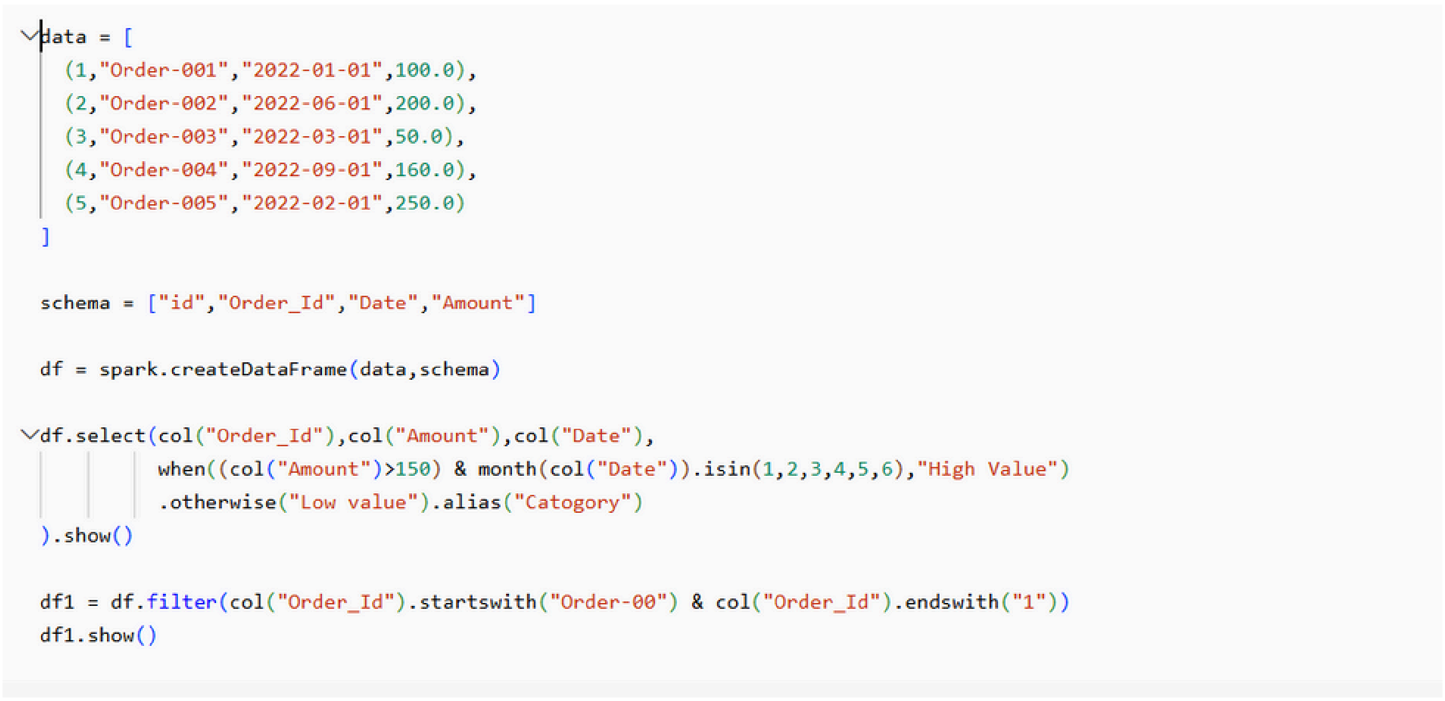
```
data = [

 (1,"Order-001","2022-01-01",100.0),

 (2,"Order-002","2022-06-01",200.0),

 (3,"Order-003","2022-03-01",50.0),

 (4,"Order-004","2022-09-01",160.0),

 (5,"Order-005","2022-02-01",250.0)

]
```

Scala Spark -



```scala
 import spark.implicits._
val data = List(
 (1,"Order-001","2022-01-01",100.0),
 (2,"Order-002","2022-06-01",200.0),
 (3,"Order-003","2022-03-01",50.0),
 (4,"Order-004","2022-09-01",160.0),
 (5,"Order-005","2022-02-01",250.0)
).toDF("id","Order_Id","Date","Amount")

 data.select(col("Order_Id"),col("Amount"),col("Date"),
   when((col("Amount")>150) && month(col("Date")).isin(1,2,3,4,5,6),"High Value")
     .otherwise("Low value").alias("Catogory")
 ).show()

 val df1 = data.filter(col("Order_Id").startsWith("Order-00") && col("Order_Id").endsWith("1"))
 df1.show()
```

PySpark -



```python
data = [
  (1,"Order-001","2022-01-01",100.0),
  (2,"Order-002","2022-06-01",200.0),
  (3,"Order-003","2022-03-01",50.0),
  (4,"Order-004","2022-09-01",160.0),
  (5,"Order-005","2022-02-01",250.0)
]

schema = ["id","Order_Id","Date","Amount"]

df = spark.createDataFrame(data,schema)

df.select(col("Order_Id"),col("Amount"),col("Date"),
          when((col("Amount")>150) & month(col("Date")).isin(1,2,3,4,5,6),"High Value")
          .otherwise("Low value").alias("Catogory")
).show()

df1 = df.filter(col("Order_Id").startswith("Order-00") & col("Order_Id").endswith("1"))
df1.show()
```

Output -

```
+---------+------+----------+----------+
| Order_Id|Amount|      Date|  Catogory|
+---------+------+----------+----------+
|Order-001| 100.0|2022-01-01| Low value|
|Order-002| 200.0|2022-06-01|High Value|
|Order-003|  50.0|2022-03-01| Low value|
|Order-004| 160.0|2022-09-01| Low value|
|Order-005| 250.0|2022-02-01|High Value|
+---------+------+----------+----------+


+---+---------+----------+------+
| id| Order_Id|      Date|Amount|
+---+---------+----------+------+
|  1|Order-001|2022-01-01| 100.0|
+---+---------+----------+------+
```

Created By :

Harshavardhana I

Data Engineer