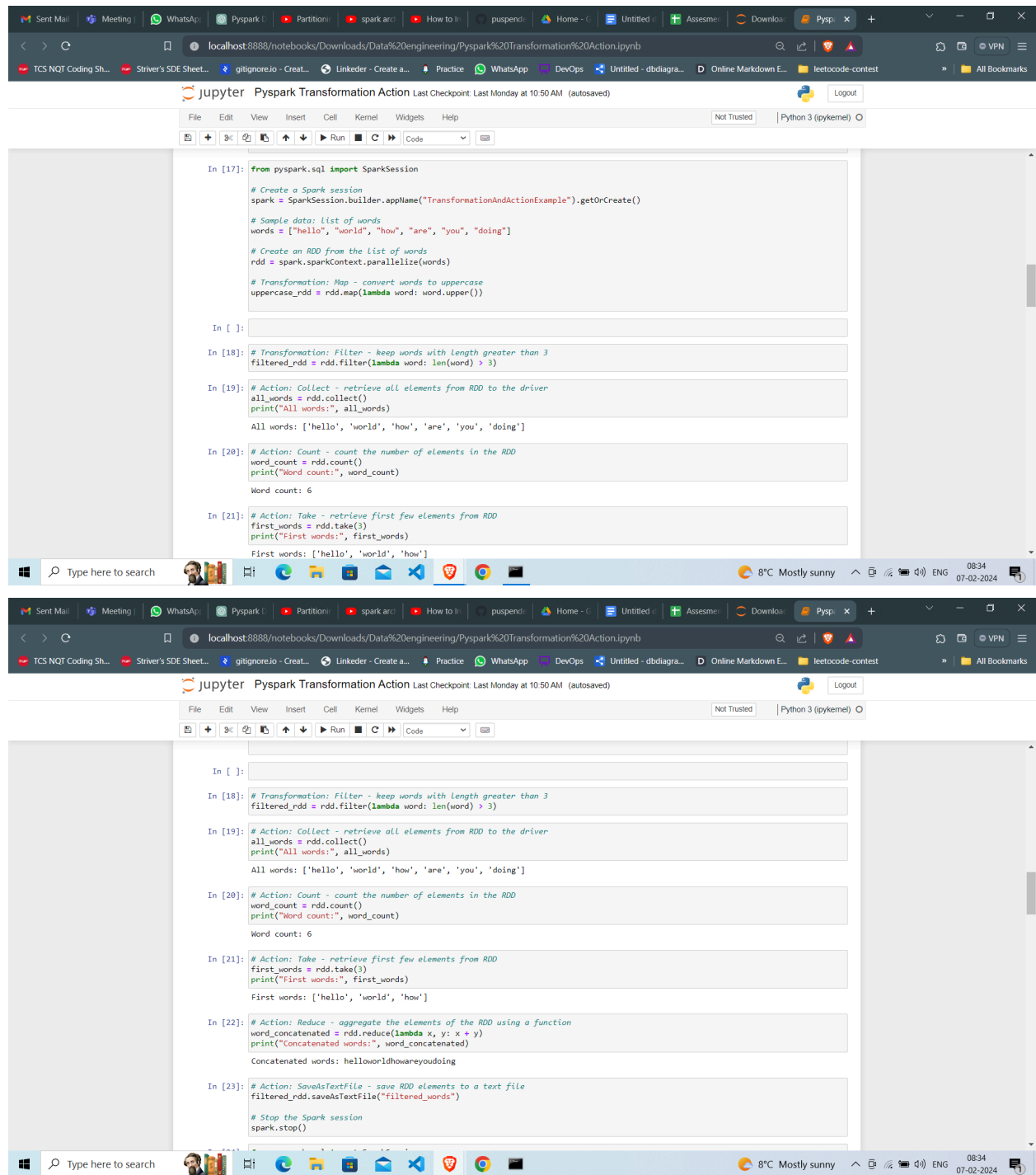


Transformation And Actions



The image displays two screenshots of a Jupyter Notebook interface, showing the execution of PySpark code for RDD transformations and actions. The notebook is titled "PySpark Transformation Action" and is running on a local host (localhost:8888).

Top Screenshot: Shows the initial setup and basic transformations.

```
In [17]: from pyspark.sql import SparkSession

# Create a Spark session
spark = SparkSession.builder.appName("TransformationAndActionExample").getOrCreate()

# Sample data: List of words
words = ["hello", "world", "how", "are", "you", "doing"]

# Create an RDD from the list of words
rdd = spark.sparkContext.parallelize(words)

# Transformation: Map - convert words to uppercase
uppercase_rdd = rdd.map(lambda word: word.upper())

In [18]: # Transformation: Filter - keep words with length greater than 3
filtered_rdd = rdd.filter(lambda word: len(word) > 3)

In [19]: # Action: Collect - retrieve all elements from RDD to the driver
all_words = rdd.collect()
print("All words:", all_words)
All words: ['hello', 'world', 'how', 'are', 'you', 'doing']

In [20]: # Action: Count - count the number of elements in the RDD
word_count = rdd.count()
print("Word count:", word_count)
Word count: 6

In [21]: # Action: Take - retrieve first few elements from RDD
first_words = rdd.take(3)
print("First words:", first_words)
First words: ['hello', 'world', 'how']
```

Bottom Screenshot: Shows further transformations and actions.

```
In [18]: # Transformation: Filter - keep words with length greater than 3
filtered_rdd = rdd.filter(lambda word: len(word) > 3)

In [19]: # Action: Collect - retrieve all elements from RDD to the driver
all_words = rdd.collect()
print("All words:", all_words)
All words: ['hello', 'world', 'how', 'are', 'you', 'doing']

In [20]: # Action: Count - count the number of elements in the RDD
word_count = rdd.count()
print("Word count:", word_count)
Word count: 6

In [21]: # Action: Take - retrieve first few elements from RDD
first_words = rdd.take(3)
print("First words:", first_words)
First words: ['hello', 'world', 'how']

In [22]: # Action: Reduce - aggregate the elements of the RDD using a function
word_concatenated = rdd.reduce(lambda x, y: x + y)
print("Concatenated words:", word_concatenated)
Concatenated words: helloworldhowareyoudoing

In [23]: # Action: SaveAsTextFile - save RDD elements to a text file
filtered_rdd.saveAsTextFile("filtered_words")

# Stop the Spark session
spark.stop()
```

SQL Query in PySpark

Jupyter Spark Transformation Action Last Checkpoint Last Monday at 10:50 AM (autosaved)

```
In [31]: import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import approx_count_distinct, collect_list
from pyspark.sql.functions import collect_set, sum, avg, max, countDistinct, count
from pyspark.sql.functions import first, last, kurtosis, min, mean, skewness
from pyspark.sql.functions import stddev, stddev_samp, stddev_pop, sumDistinct
from pyspark.sql.functions import variance, var_samp, var_pop

spark = SparkSession.builder.appName('pyspark_ex').getOrCreate()

simpleData = [("James", "Sales", 3000),
              ("Michael", "Sales", 4600),
              ("Robert", "Sales", 4100),
              ("Maria", "Finance", 3000),
              ("James", "Sales", 3000),
              ("Scott", "Finance", 3300),
              ("Jen", "Finance", 3900),
              ("Jeff", "Marketing", 3000),
              ("Kumar", "Marketing", 2000),
              ("Saif", "Sales", 4100)]

schema = ["employee_name", "department", "salary"]

df = spark.createDataFrame(data=simpleData, schema=schema)
df.printSchema()
df.show(truncate=False)

root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- salary: long (nullable = true)

+-----+-----+-----+
|employee_name|department|salary|
+-----+-----+-----+
|James       |Sales     |3000   |
|Michael     |Sales     |4600   |
|Robert      |Sales     |4100   |
|Maria       |Finance   |3000   |
|James       |Sales     |3000   |
|Scott       |Finance   |3300   |
|Jen         |Finance   |3900   |
|Jeff        |Marketing |3000   |
|Kumar       |Marketing |2000   |
|Saif        |Sales     |4100   |
+-----+-----+-----+
```

Jupyter Spark Transformation Action Last Checkpoint Last Monday at 10:50 AM (autosaved)

```
In [32]: print("approx_count_distinct: " + \
          str(df.select(approx_count_distinct("salary")).collect()[0][0]))

approx_count_distinct: 6

In [33]: print("avg: " + str(df.select(avg("salary")).collect()[0][0]))

avg: 3400.0

In [34]: df.select(collect_list("salary")).show(truncate=False)
df.select(collect_set("salary")).show(truncate=False)

+-----+-----+
|collect_list(salary)|
+-----+-----+
|[3000, 4600, 4100, 3000, 3000, 3300, 3900, 3000, 2000, 4100]|
+-----+-----+

+-----+-----+
|collect_set(salary)|
+-----+-----+
|[4600, 3000, 3900, 4100, 3300, 2000]|
+-----+-----+

In [35]: df2 = df.select(countDistinct("department", "salary"))
df2.show(truncate=False)
print("Distinct Count of Department & Salary: "+str(df2.collect()[0][0]))

+-----+-----+
|count(DISTINCT department, salary)|
+-----+-----+
|8|
+-----+-----+

Distinct Count of Department & Salary: 8
```

Browser tabs: Sent Mail, Meeting, WhatsApp, Pyspark I, Partitioni, spark arc, How to li, suspendi, Home - C, Untitled c, Assesmer, Download, Pysp. x, +

Address bar: localhost:8888/notebooks/Downloads/Data%20Engineering/Pyspark%20Transformation%20Action.ipynb

Browser extensions: TCS NQT Coding Sh..., Striver's SDE Sheet..., gitignorejo - Creat..., Linkeder - Create a..., Practice, WhatsApp, DevOps, Untitled - dbdiagra..., Online Markdown E..., leetcode-contest, All Bookmarks

Jupyter Pyspark Transformation Action Last Checkpoint: Last Monday at 10:50 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O

Code editor toolbar: +, -, <, >, Run, C, Code, Comment

```
[[4600, 3000, 3900, 4100, 3300, 2000]]
```

```
In [35]: df2 = df.select(countDistinct("department", "salary"))
df2.show(truncate=False)
print("Distinct Count of Department & Salary: "+str(df2.collect()[0][0]))
```

```
+-----+
|count(DISTINCT department, salary)|
+-----+
|8|
+-----+

Distinct Count of Department & Salary: 8
```

```
In [36]: print("count: "+str(df.select(count("salary")).collect()[0]))
df.select(first("salary")).show(truncate=False)
```

```
count: Row(count(salary)=10)
+-----+
|first(salary)|
+-----+
|3000|
+-----+
```

In []:

In []:

Windows taskbar: Type here to search, icons for various apps, Record low, 08:50, 07-02-2024