Experiment 11

Aim:

To perform data analytics using **Apache Spark** on the Amazon Food Dataset to identify **pairs of products that are frequently reviewed together** by the same users.

Theory:

Apache Spark is a distributed computing framework used for large-scale data processing. It provides RDD and DataFrame APIs to perform parallel operations on large datasets across multiple cluster nodes.

The Amazon Food Review dataset contains user IDs and product IDs reviewed by users.
To find frequently co-reviewed products, we must:

- Transform raw (user, product) data into:
  user_id → list of product_ids
- Generate all product pairs reviewed by each user.
- Count how many users reviewed each product pair together.
- Filter pairs with frequency greater than 1.
- Sort by frequency to identify strongly associated product pairs.

Spark transformations used:

- map(), groupByKey(), flatMap()
- reduceByKey(), filter(), sortBy()
- DataFrame functions like collect_list() and explode()

This experiment demonstrates Spark's ability to handle large datasets and compute meaningful relationships at scale.

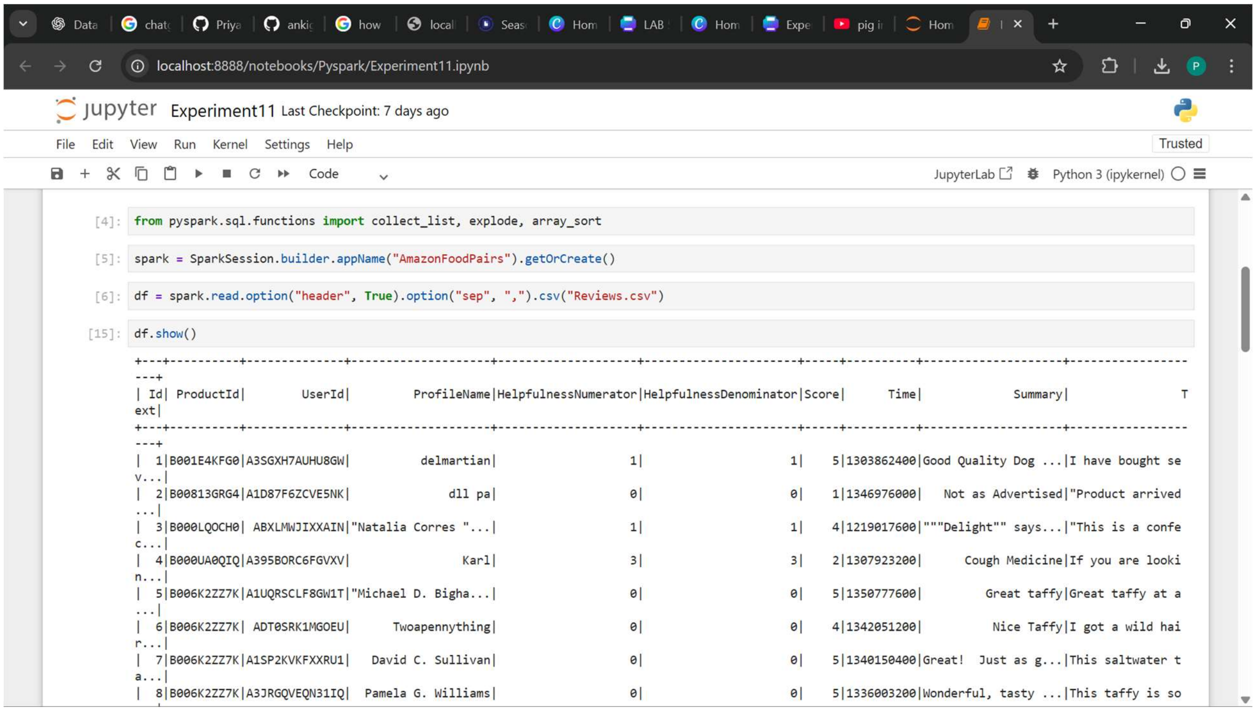
Procedure:

Step 1: Download Dataset

Download the **Amazon Fine Food Reviews** dataset from:
https://snap.stanford.edu/data/web-FineFoods.html

Step 2: Load Data into Spark

Read CSV/TSV file containing UserId and ProductId.


localhost:8888/notebooks/Pyspark/Experiment11.ipynb
Jupyter Experiment11 Last Checkpoint: 7 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[4]: from pyspark.sql.functions import collect_list, explode, array_sort

[5]: spark = SparkSession.builder.appName("AmazonFoodPairs").getOrCreate()

[6]: df = spark.read.option("header", True).option("sep", ",").csv("Reviews.csv")

[15]: df.show()
```

```
+---+----------+--------------+--------------------+--------------------+----------------------+-----+----------+--------------------+----------------
---+
| Id| ProductId|        UserId|         ProfileName|HelpfulnessNumerator|HelpfulnessDenominator|Score|      Time|             Summary|               T
ext|
+---+----------+--------------+--------------------+--------------------+----------------------+-----+----------+--------------------+----------------
---+
|  1|B001E4KFG0|A3SGXH7AUHU8GW|          delmartian|                   1|                     1|    5|1303862400|Good Quality Dog ...|I have bought se
v...|
|  2|B00813GRG4|A1D87F6ZCVE5NK|              dll pa|                   0|                     0|    1|1346976000|   Not as Advertised|"Product arrived
...|
|  3|B000LQOCH0| ABXLMWJIXXAIN|"Natalia Corres "...|                   1|                     1|    4|1219017600|"""Delight"" says...|"This is a confe
c...|
|  4|B000UA0QIQ|A395BORC6FGVXV|                Karl|                   3|                     3|    2|1307923200|      Cough Medicine|If you are looki
n...|
|  5|B006K2ZZ7K|A1UQRSCLF8GW1T|"Michael D. Bigha...|                   0|                     0|    5|1350777600|         Great taffy|Great taffy at a
...|
|  6|B006K2ZZ7K| ADT0SRK1MGOEU|      Twoapennything|                   0|                     0|    4|1342051200|          Nice Taffy|I got a wild hai
r...|
|  7|B006K2ZZ7K|A1SP2KVKFXXRU1|   David C. Sullivan|                   0|                     0|    5|1340150400|Great!  Just as g...|This saltwater t
a...|
|  8|B006K2ZZ7K|A3JRGQVEQN31IQ|  Pamela G. Williams|                   0|                     0|    5|1336003200|Wonderful, tasty ...|This taffy is so
```

Step 3: Extract User–Product Pairs

Convert rows into key-value pairs:
(UserId, ProductId)

Step 4: Group Products by User

Use groupByKey() or DataFrame groupBy().agg(collect_list()) to obtain:
UserId → [P1, P2, P3, ...]

Step 5: Generate Product Pairs

For each user's product list, generate all 2-item combinations:

(P1, P2), (P1, P3), (P2, P3), ...

Step 6: Count Pair Frequencies

Use reduceByKey() or DataFrame groupBy().count() to compute:

(ProductA, ProductB) → frequency

Step 7: Filter & Sort

Keep only those pairs appearing more than once and sort them in descending order of frequency.

Step 8: Save Output

Write results to HDFS/local storage as CSV/text.

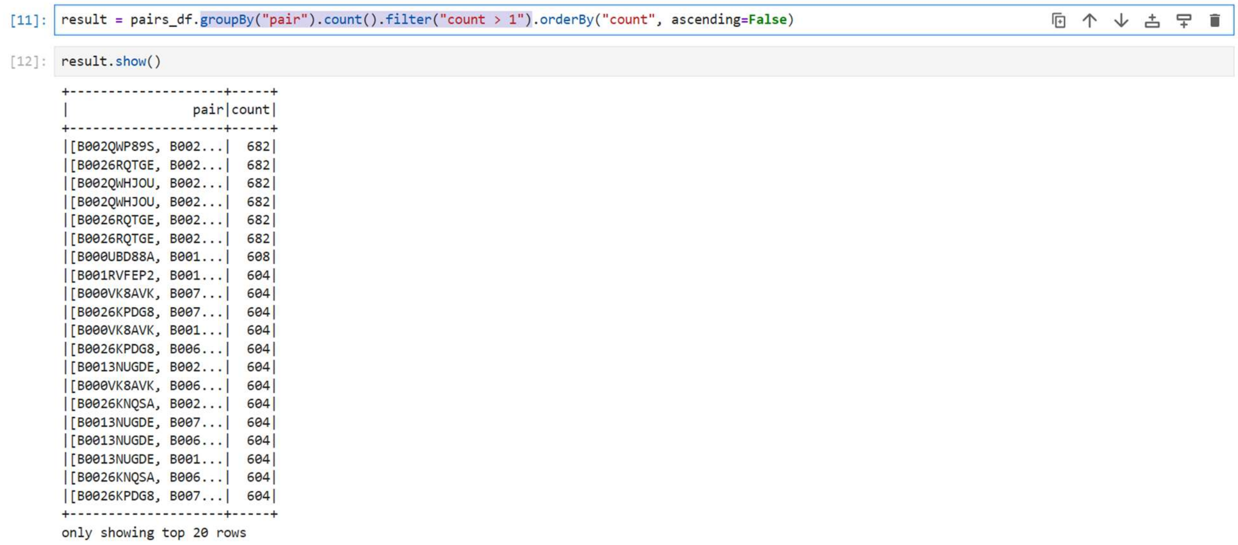
Output:

A list of product pairs reviewed together more than once, displayed or saved in sorted order. Example:

(B001E4KFG0, B00813GRG4) → 12

(B00474GHKQ, B00813GRG4) → 9

(B0078LX05C, B00474GHKQ) → 8



```
[11]: result = pairs_df.groupBy("pair").count().filter("count > 1").orderBy("count", ascending=False)

[12]: result.show()

+--------------------+-----+
|                pair|count|
+--------------------+-----+
|[B002QWP89S, B002...|  682|
|[B0026RQTGE, B002...|  682|
|[B002QWHJOU, B002...|  682|
|[B002QWHJOU, B002...|  682|
|[B0026RQTGE, B002...|  682|
|[B0026RQTGE, B002...|  682|
|[B000UBD88A, B001...|  608|
|[B001RVFEP2, B001...|  604|
|[B000VK8AVK, B007...|  604|
|[B0026KPDG8, B007...|  604|
|[B000VK8AVK, B001...|  604|
|[B0026KPDG8, B006...|  604|
|[B0013NUGDE, B002...|  604|
|[B000VK8AVK, B006...|  604|
|[B0026KNQSA, B002...|  604|
|[B0013NUGDE, B007...|  604|
|[B0013NUGDE, B006...|  604|
|[B0013NUGDE, B001...|  604|
|[B0026KNQSA, B006...|  604|
|[B0026KPDG8, B007...|  604|
+--------------------+-----+
only showing top 20 rows
```

Fig of Result

Conclusion:

In this experiment, we successfully applied Apache Spark to analyze a large, real-world dataset. We discovered frequently co-reviewed product pairs using transformations and aggregations. This demonstrates Spark's ability to efficiently handle parallel computation for big data analytics.