Project 7

Big Data Analysis

Kristie Kooken

Due: March 2, 2024

For Project 7, I investigated data from E-Commerce - Product Inventory Management (www.kaggle.com) to gain insights into sales trends and reordering points. The e-commerce dataset has about 500,000 records with 8 column variables: Invoice Number as a numeric variable, Stock Code as a categorical variable, Description as a text character description, Quantity as a numeric variable (integer) for the number of items in the Description, Invoice Date as a character variable (date of transaction), Unit Price as a numeric variable representing the cost of the item in the Description, Customer ID as a numeric variable representing the customer's identifier and Country as a text variable representing the country selling the item in the Description.

Specific questions of interest included:

- What are the top 5 countries that have the highest prices?
- What are the top 5 countries that have the lowest prices?
- Who are the top 5 performing countries (sells the most quantities) in the world?
- Who are the lowest performing countries (sells the least quantities) in the world?
- Does pricing predict quantity?

To investigate these questions, this data was ingested in NiFi and analyzed using Spark, Pyspark. Initially, I did use scala to read the .json format to a dataframe and I describe challenges I encountered with this process below. I wanted to bring the data in using NiFi as I was interested to learn more about how NiFi might work to ingest larger amounts of data in what I thought might be a more common data flow. Though I did a smaller

amount of analysis in scala – I ended up redoing all analysis in Pyspark. I picked Pyspark as it has many powerful components and is more intuitive from my perspective. As it is very widely used, I wanted to gain more experience with it.

To bring the data into NiFi, I copied the e-commerce dataset to our Google Cloud VM location using scp and copied the data to the appropriate folder location for NiFi. Then, I created a flow in NiFi to read the .csv into NiFi and create a .json file. Configuring the NiFi flow was challenging and after following a few different methods, I decided to use a tutorial and template as a guide (www.community.cloudera.com). Screenshot 1 shows the successful NiFi flow for this process and screenshot 2 shows the original file and the converted .json format of the file saved to the /dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data location. Screenshots 3 and 4 show contents of each file in the original and json format.

Once the data was uploaded to the data folder, I read the file into a dataframe using Spark scala (Screenshot 5). I encountered many challenges doing this, and then learned that json can be 1.5 to 3 times bigger than .csv format (www.jsoneditoronline.org). A file of this size was very challenging to process on our VM instance (Screenshot 6) where our cluster does not have a lot of compute and because of this, I decided to move forward with analyzing the .csv file instead.

Because I moved forward with analyzing the .csv format, I decided to use Pyspark for data cleaning and analyses. After initially reading in the data, I looked at the highest or lowest values of frequency counts as well as the country list to determine if there were values that should be removed from analysis. Data cleaning steps included filtering the Country variable to remove “Unspecified” as well as “European Community” and only

keep those values of Unit Price that were greater than 0.10. Also, any missing values as well as any duplicates were removed from the data. Screenshot 7 shows reading in the data and creating a dataframe and changing numeric strings to numeric columns. Screenshot 8a shows frequencies of each column and displays the head or the tail to inspect each column for values that are not allowed for analysis (e.g., a negative value for Unit Price). Screenshot 8b shows the frequency of the column Country to clean this variable of entries that do not represent a single country. Screenshot 8c shows data cleaning steps of filtering and then removing duplicate rows as well as missing values. Screenshot 9 is the first 20 rows of the dataset used for analysis.

To investigate the highest and lowest performing countries in terms of item prices and quantities of items, a grouped frequency that sorted the data from highest to lowest was run, Screenshot 10 has the results for quantity and Screenshot 11 has the results for pricing.

Initially I was interested in running correlations or descriptive statistics on the dataframe and used the Summarizer package in Pyspark to investigate running these analyses. However, I was not able to run this error-free and need to further investigate what needs to be corrected with my code. Screenshot 12 shows the test data run I conducted in order to better understand the input format and vectors (www.spark.apache.org).

The final analysis that was run in Pyspark was a linear regression using Quantity as the dependent variable and Price as the independent variable, Screenshot 13. This simple regression model did not yield good results, Screenshot 14. I was very interested

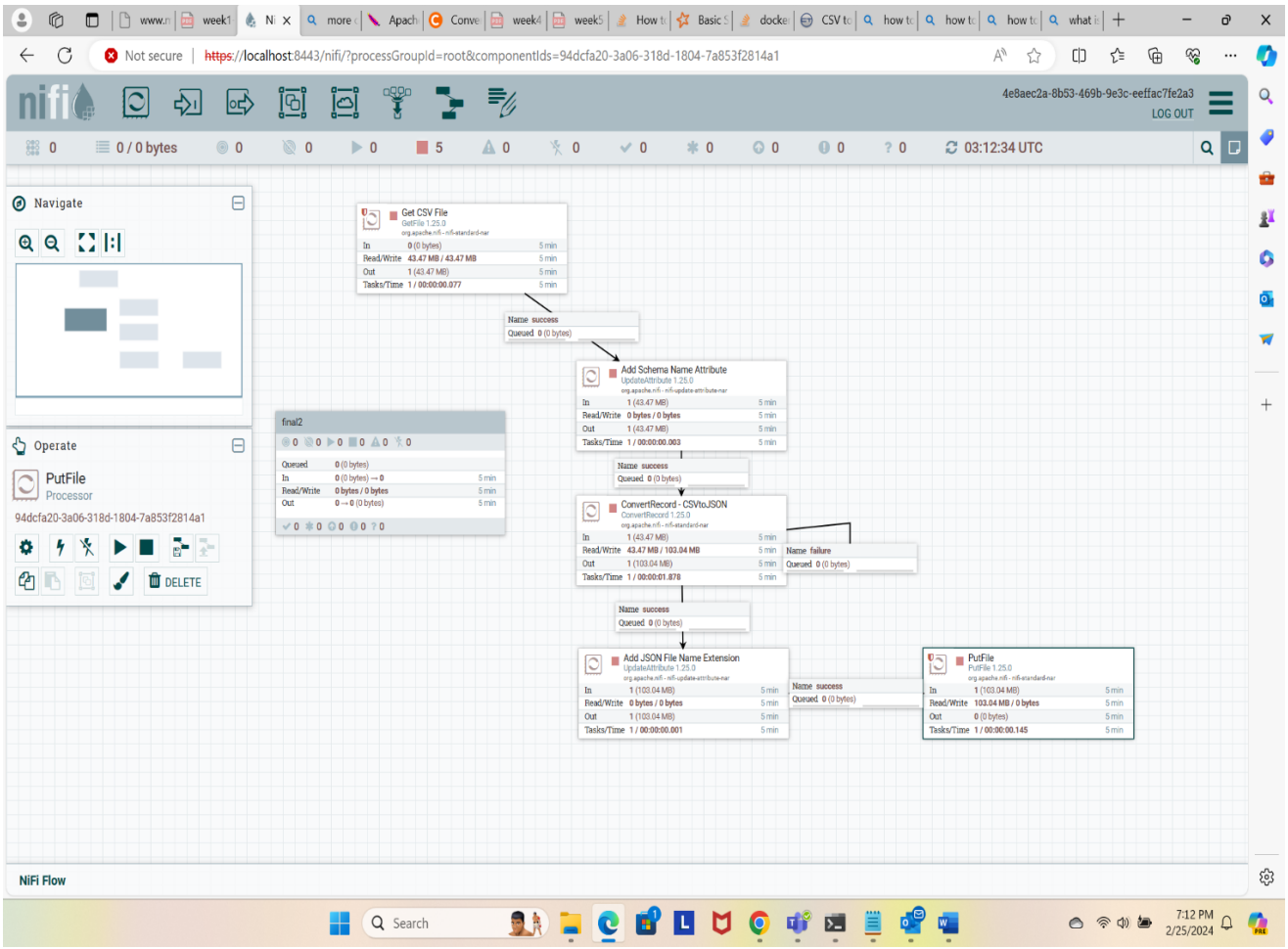
to get to this stage and explore how to run a model in Pyspark. There is a lot more to learn but it is interesting and motivating to scratch the surface of these analyses.

There were different issues that I ran into while completing this project, some I figured out and some are still being worked on. An issue that caused a lot of re-work was not knowing how to install packages of python that were missing (e.g., numpy) and though I figured this out (install on all nodes), it made me realize that it is quite complex to have a production environment even with installing a new package. Likewise, I spent a long time trying to install a plotting package until I read several articles on how plotting is often done outside of a big data architecture.

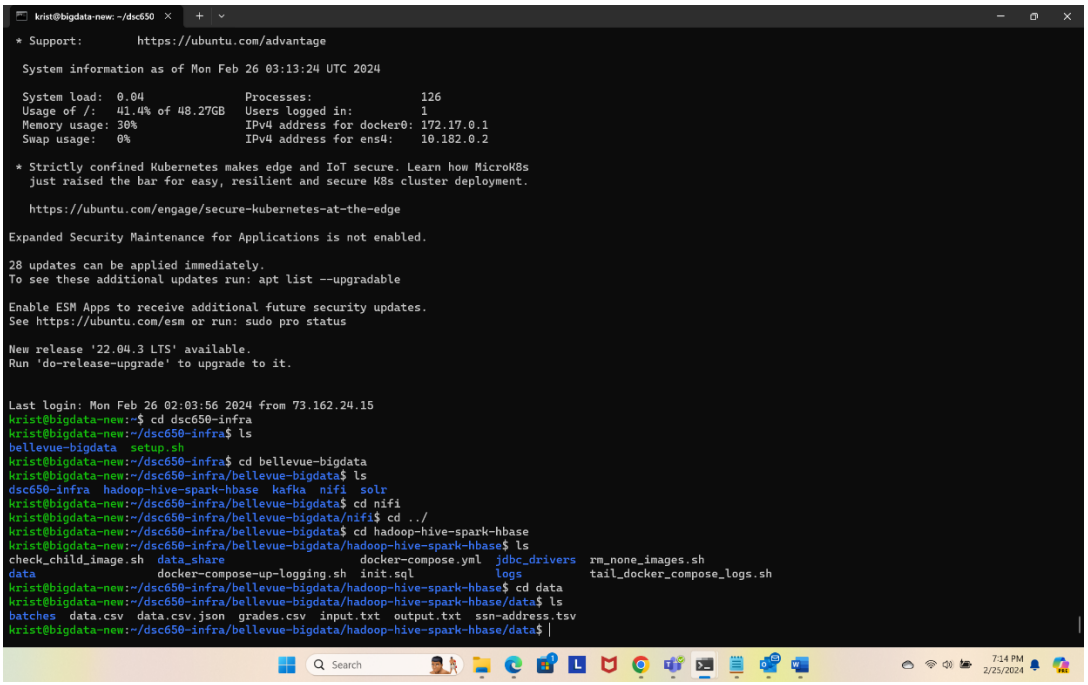
In closing, this project allowed me to explore a variety of tools and get a better understanding of how big data architecture works from data flow to analyses. There is a lot more to learn, however I enjoyed learning more about NiFi, scala and Pyspark as mentioned here.

Screenshots referenced in this document:

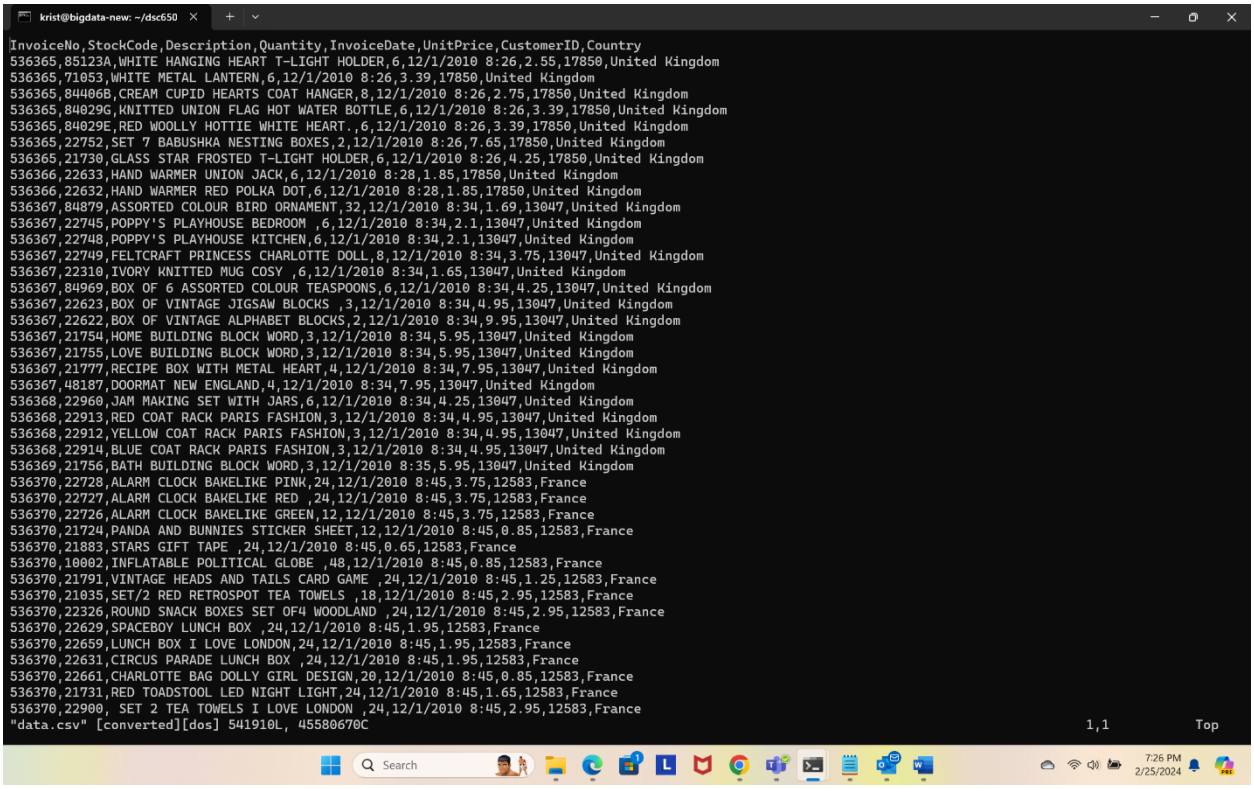
#1:



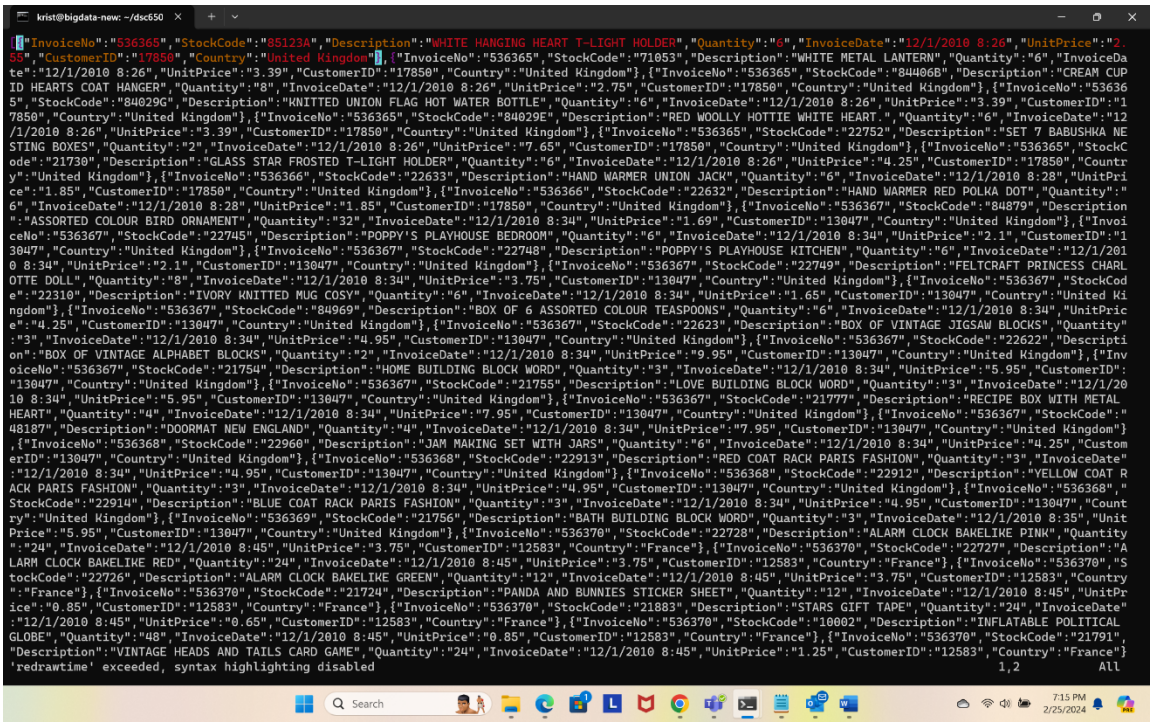
#2:


krist@bigdata-new: ~/dsc650
* Support: https://ubuntu.com/advantage
System information as of Mon Feb 26 03:13:24 UTC 2024
System load: 0.04 Processes: 126
Usage of /: 41.4% of 48.27GB Users logged in: 1
Memory usage: 30% IPv4 address for docker0: 172.17.0.1
Swap usage: 0% IPv4 address for ens4: 10.182.0.2
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge
Expanded Security Maintenance for Applications is not enabled.
28 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Mon Feb 26 02:03:56 2024 from 73.162.24.15
krist@bigdata-new:~$ cd dsc650-infra
krist@bigdata-new:~/dsc650-infra$ ls
bellevue-bigdata setup.sh
krist@bigdata-new:~/dsc650-infra$ cd bellevue-bigdata
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata$ ls
dsc650-infra hadoop-hive-spark-hbase kafka nifi solr
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata$ cd nifi
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata/nifi$ cd ../
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata$ cd hadoop-hive-spark-hbase
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase$ ls
check_child_image.sh data_share docker-compose.yml jdbc_drivers rm_none_images.sh
data docker-compose-up-logging.sh init.sql logs tail_docker_compose_logs.sh
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase$ cd data
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data$ ls
batches data.csv data.csv.json grades.csv input.txt output.txt ssn-address.tsv
krist@bigdata-new:~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase/data$
Search
7:14 PM
2/25/2024

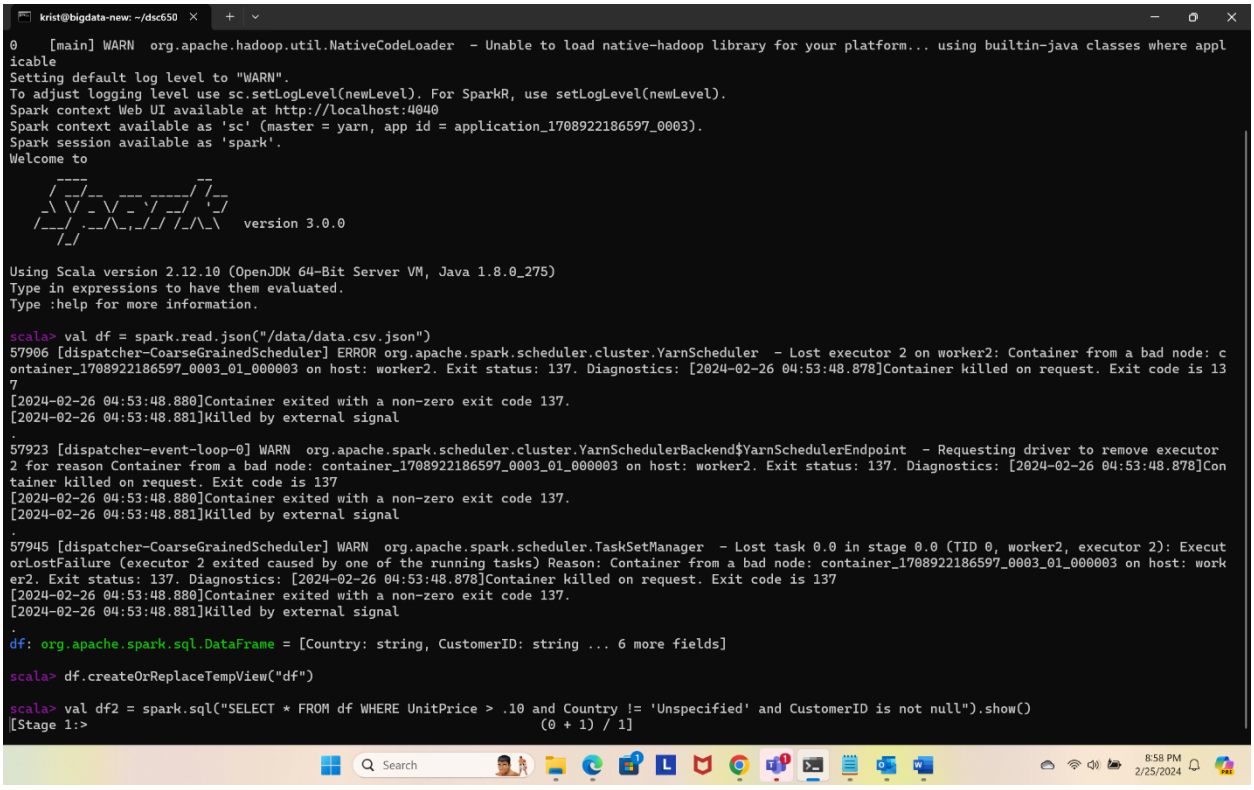
#3:


krist@bigdata-new: ~/dsc650
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010 8:26,2.55,17850,United Kingdom
536365,71053,WHITE METAL LANTERN,6,12/1/2010 8:26,3.39,17850,United Kingdom
536365,84406B,CREAM CUPID HEARTS COAT HANGER,8,12/1/2010 8:26,2.75,17850,United Kingdom
536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6,12/1/2010 8:26,3.39,17850,United Kingdom
536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6,12/1/2010 8:26,3.39,17850,United Kingdom
536365,22752,SET 7 BABUSHKA NESTING BOXES,2,12/1/2010 8:26,7.65,17850,United Kingdom
536365,21730,GLASS STAR FROSTED T-LIGHT HOLDER,6,12/1/2010 8:26,4.25,17850,United Kingdom
536366,22633,HAND WARMER UNION JACK,6,12/1/2010 8:28,1.85,17850,United Kingdom
536366,22632,HAND WARMER RED POLKA DOT,6,12/1/2010 8:28,1.85,17850,United Kingdom
536367,84879,ASSORTED COLOUR BIRD ORNAMENT,32,12/1/2010 8:34,1.69,13047,United Kingdom
536367,22745,POPPY'S PLAYHOUSE BEDROOM ,6,12/1/2010 8:34,2.1,13047,United Kingdom
536367,22748,POPPY'S PLAYHOUSE KITCHEN,6,12/1/2010 8:34,2.1,13047,United Kingdom
536367,22749,FELTCRAFT PRINCESS CHARLOTTE DOLL,8,12/1/2010 8:34,3.75,13047,United Kingdom
536367,22310,IVORY KNITTED MUG COSY ,6,12/1/2010 8:34,1.65,13047,United Kingdom
536367,84969,BOX OF 6 ASSORTED COLOUR TEASPOONS,6,12/1/2010 8:34,4.25,13047,United Kingdom
536367,22623,BOX OF VINTAGE JIGSAW BLOCKS ,3,12/1/2010 8:34,4.95,13047,United Kingdom
536367,22622,BOX OF VINTAGE ALPHABET BLOCKS,2,12/1/2010 8:34,9.95,13047,United Kingdom
536367,21754,HOME BUILDING BLOCK WORD,3,12/1/2010 8:34,5.95,13047,United Kingdom
536367,21755,LOVE BUILDING BLOCK WORD,3,12/1/2010 8:34,5.95,13047,United Kingdom
536367,21777,RECIPE BOX WITH METAL HEART,4,12/1/2010 8:34,7.95,13047,United Kingdom
536367,48187,DOORMAT NEW ENGLAND,4,12/1/2010 8:34,7.95,13047,United Kingdom
536368,22960,JAM MAKING SET WITH JARS,6,12/1/2010 8:34,4.25,13047,United Kingdom
536368,22913,RED COAT RACK PARIS FASHION,3,12/1/2010 8:34,4.95,13047,United Kingdom
536368,22912,YELLOW COAT RACK PARIS FASHION,3,12/1/2010 8:34,4.95,13047,United Kingdom
536368,22914,BLUE COAT RACK PARIS FASHION,3,12/1/2010 8:34,4.95,13047,United Kingdom
536369,21756,BATH BUILDING BLOCK WORD,3,12/1/2010 8:35,5.95,13047,United Kingdom
536370,22728,ALARM CLOCK BAKELIKE PINK,24,12/1/2010 8:45,3.75,12583,France
536370,22727,ALARM CLOCK BAKELIKE RED ,24,12/1/2010 8:45,3.75,12583,France
536370,22726,ALARM CLOCK BAKELIKE GREEN,12,12/1/2010 8:45,3.75,12583,France
536370,21724,PANDA AND BUNNIES STICKER SHEET,12,12/1/2010 8:45,0.85,12583,France
536370,21883,STARS GIFT TAPE ,24,12/1/2010 8:45,0.65,12583,France
536370,10002,INFLATABLE POLITICAL GLOBE ,48,12/1/2010 8:45,0.85,12583,France
536370,21791,VINTAGE HEADS AND TAILS CARD GAME ,24,12/1/2010 8:45,1.25,12583,France
536370,21035,SET/2 RED RETROSPOT TEA TOWELS ,18,12/1/2010 8:45,2.95,12583,France
536370,22326,ROUND SNACK BOXES SET OF4 WOODLAND ,24,12/1/2010 8:45,2.95,12583,France
536370,22629,SPACEBOY LUNCH BOX ,24,12/1/2010 8:45,1.95,12583,France
536370,22659,LUNCH BOX I LOVE LONDON,24,12/1/2010 8:45,1.95,12583,France
536370,22631,CIRCUS PARADE LUNCH BOX ,24,12/1/2010 8:45,1.95,12583,France
536370,22661,CHARLOTTE BAG DOLLY GIRL DESIGN,20,12/1/2010 8:45,0.85,12583,France
536370,21731,RED TOADSTOOL LED NIGHT LIGHT,24,12/1/2010 8:45,1.65,12583,France
536370,22900, SET 2 TEA TOWELS I LOVE LONDON ,24,12/1/2010 8:45,2.95,12583,France
"data.csv" [converted][dos] 541910L, 45580670C
1,1
Top
Search
7:26 PM
2/25/2024

#4:


krist@bigdata-new: ~/dsc650
[{"InvoiceNo":"536365","StockCode":"85123A","Description":"WHITE HANGING HEART T-LIGHT HOLDER","Quantity":"6","InvoiceDate":"12/1/2010 8:26","UnitPrice":"2.55","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"71053","Description":"WHITE METAL LANTERN","Quantity":"6","InvoiceDate":"12/1/2010 8:26","UnitPrice":"3.39","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"84406B","Description":"CREAM CUPID HEARTS COAT HANGER","Quantity":"8","InvoiceDate":"12/1/2010 8:26","UnitPrice":"2.75","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"84029G","Description":"KNITTED UNION FLAG HOT WATER BOTTLE","Quantity":"6","InvoiceDate":"12/1/2010 8:26","UnitPrice":"3.39","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"84029E","Description":"RED WOOLLY HOTTIE WHITE HEART.","Quantity":"6","InvoiceDate":"12/1/2010 8:26","UnitPrice":"3.39","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"22752","Description":"SET 7 BABUSHKA NESTING BOXES","Quantity":"2","InvoiceDate":"12/1/2010 8:26","UnitPrice":"7.65","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536365","StockCode":"21730","Description":"GLASS STAR FROSTED T-LIGHT HOLDER","Quantity":"6","InvoiceDate":"12/1/2010 8:26","UnitPrice":"4.25","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536366","StockCode":"22633","Description":"HAND WARMER UNION JACK","Quantity":"6","InvoiceDate":"12/1/2010 8:28","UnitPrice":"1.85","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536366","StockCode":"22632","Description":"HAND WARMER RED POLKA DOT","Quantity":"6","InvoiceDate":"12/1/2010 8:28","UnitPrice":"1.85","CustomerID":"17850","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"84879","Description":"ASSORTED COLOUR BIRD ORNAMENT","Quantity":"32","InvoiceDate":"12/1/2010 8:34","UnitPrice":"1.69","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22745","Description":"POPPY'S PLAYHOUSE BEDROOM","Quantity":"6","InvoiceDate":"12/1/2010 8:34","UnitPrice":"2.1","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22748","Description":"POPPY'S PLAYHOUSE KITCHEN","Quantity":"6","InvoiceDate":"12/1/2010 8:34","UnitPrice":"2.1","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22749","Description":"FELTCRAFT PRINCESS CHARLOTTE DOLL","Quantity":"8","InvoiceDate":"12/1/2010 8:34","UnitPrice":"3.75","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22310","Description":"IVORY KNITTED MUG COSY","Quantity":"6","InvoiceDate":"12/1/2010 8:34","UnitPrice":"1.65","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"84969","Description":"BOX OF 6 ASSORTED COLOUR TEASPOONS","Quantity":"6","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.25","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22623","Description":"BOX OF VINTAGE JIGSAW BLOCKS","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"22622","Description":"BOX OF VINTAGE ALPHABET BLOCKS","Quantity":"2","InvoiceDate":"12/1/2010 8:34","UnitPrice":"9.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"21754","Description":"HOME BUILDING BLOCK WORD","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"5.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"21755","Description":"LOVE BUILDING BLOCK WORD","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"5.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"21777","Description":"RECIPE BOX WITH METAL HEART","Quantity":"4","InvoiceDate":"12/1/2010 8:34","UnitPrice":"7.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536367","StockCode":"48187","Description":"DOORMAT NEW ENGLAND","Quantity":"4","InvoiceDate":"12/1/2010 8:34","UnitPrice":"7.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536368","StockCode":"22960","Description":"JAM MAKING SET WITH JARS","Quantity":"6","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.25","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536368","StockCode":"22913","Description":"RED COAT RACK PARIS FASHION","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536368","StockCode":"22912","Description":"YELLOW COAT RACK PARIS FASHION","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536368","StockCode":"22914","Description":"BLUE COAT RACK PARIS FASHION","Quantity":"3","InvoiceDate":"12/1/2010 8:34","UnitPrice":"4.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536369","StockCode":"21756","Description":"BATH BUILDING BLOCK WORD","Quantity":"3","InvoiceDate":"12/1/2010 8:35","UnitPrice":"5.95","CustomerID":"13047","Country":"United Kingdom"},{"InvoiceNo":"536370","StockCode":"22728","Description":"ALARM CLOCK BAKELIKE PINK","Quantity":"24","InvoiceDate":"12/1/2010 8:45","UnitPrice":"3.75","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"22727","Description":"ALARM CLOCK BAKELIKE RED","Quantity":"24","InvoiceDate":"12/1/2010 8:45","UnitPrice":"3.75","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"22726","Description":"ALARM CLOCK BAKELIKE GREEN","Quantity":"12","InvoiceDate":"12/1/2010 8:45","UnitPrice":"3.75","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"21724","Description":"PANDA AND BUNNIES STICKER SHEET","Quantity":"12","InvoiceDate":"12/1/2010 8:45","UnitPrice":"0.85","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"21883","Description":"STARS GIFT TAPE","Quantity":"24","InvoiceDate":"12/1/2010 8:45","UnitPrice":"0.65","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"10002","Description":"INFLATABLE POLITICAL GLOBE","Quantity":"48","InvoiceDate":"12/1/2010 8:45","UnitPrice":"0.85","CustomerID":"12583","Country":"France"},{"InvoiceNo":"536370","StockCode":"21791","Description":"VINTAGE HEADS AND TAILS CARD GAME","Quantity":"24","InvoiceDate":"12/1/2010 8:45","UnitPrice":"1.25","CustomerID":"12583","Country":"France"}
'redrawtime' exceeded, syntax highlighting disabled
1,2 All
Search
7:15 PM
2/25/2024

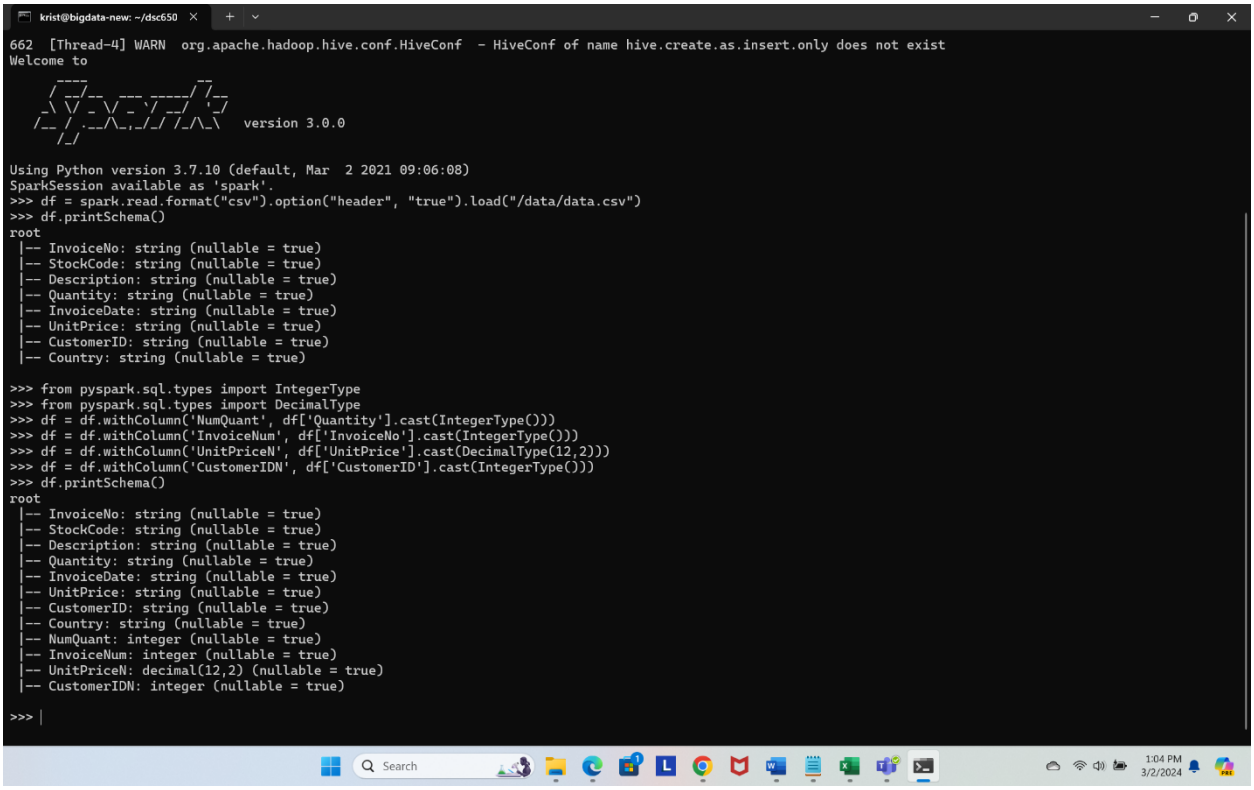
#5:


krist@bigdata-new: ~/dsc650
0 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://localhost:4040
Spark context available as 'sc' (master = yarn, app id = application_1708922186597_0003).
Spark session available as 'spark'.
Welcome to
version 3.0.0
Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_275)
Type in expressions to have them evaluated.
Type :help for more information.
scala> val df = spark.read.json("/data/data.csv.json")
57906 [dispatcher-CoarseGrainedScheduler] ERROR org.apache.spark.scheduler.cluster.YarnScheduler - Lost executor 2 on worker2: Container from a bad node: container_1708922186597_0003_01_000003 on host: worker2. Exit status: 137. Diagnostics: [2024-02-26 04:53:48.878]Container killed on request. Exit code is 137
[2024-02-26 04:53:48.880]Container exited with a non-zero exit code 137.
[2024-02-26 04:53:48.881]Killed by external signal
.
57923 [dispatcher-event-loop-0] WARN org.apache.spark.scheduler.cluster.YarnSchedulerBackend$YarnSchedulerEndpoint - Requesting driver to remove executor 2 for reason Container from a bad node: container_1708922186597_0003_01_000003 on host: worker2. Exit status: 137. Diagnostics: [2024-02-26 04:53:48.878]Container killed on request. Exit code is 137
[2024-02-26 04:53:48.880]Container exited with a non-zero exit code 137.
[2024-02-26 04:53:48.881]Killed by external signal
.
57945 [dispatcher-CoarseGrainedScheduler] WARN org.apache.spark.scheduler.TaskSetManager - Lost task 0.0 in stage 0.0 (TID 0, worker2, executor 2): ExecutorLostFailure (executor 2 exited caused by one of the running tasks) Reason: Container from a bad node: container_1708922186597_0003_01_000003 on host: worker2. Exit status: 137. Diagnostics: [2024-02-26 04:53:48.878]Container killed on request. Exit code is 137
[2024-02-26 04:53:48.880]Container exited with a non-zero exit code 137.
[2024-02-26 04:53:48.881]Killed by external signal
.
df: org.apache.spark.sql.DataFrame = [Country: string, CustomerID: string ... 6 more fields]
scala> df.createOrReplaceTempView("df")
scala> val df2 = spark.sql("SELECT * FROM df WHERE UnitPrice > .10 and Country != 'Unspecified' and CustomerID is not null").show()
[Stage 1:> (0 + 1) / 1]
Search
8:58 PM
2/25/2024

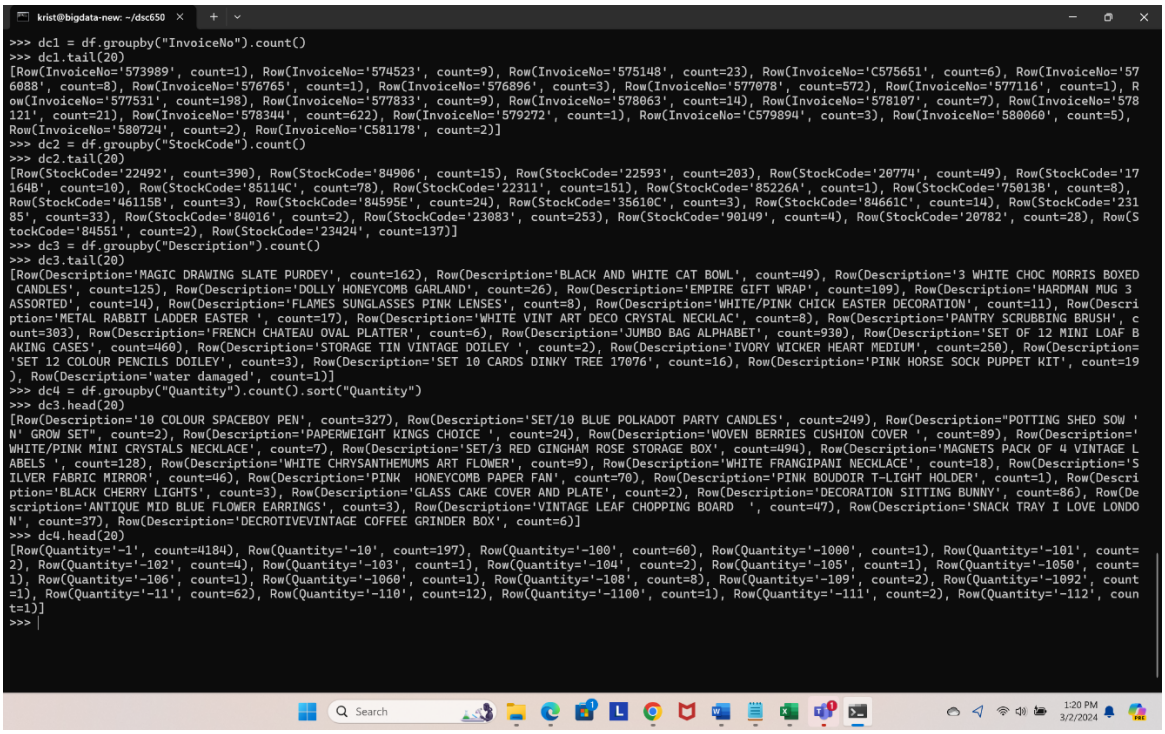
#6:


CPU Utilization
100%
0
UTC-8
7:30 PM
7:40 PM
7:50 PM
8:00 PM
8:10 PM

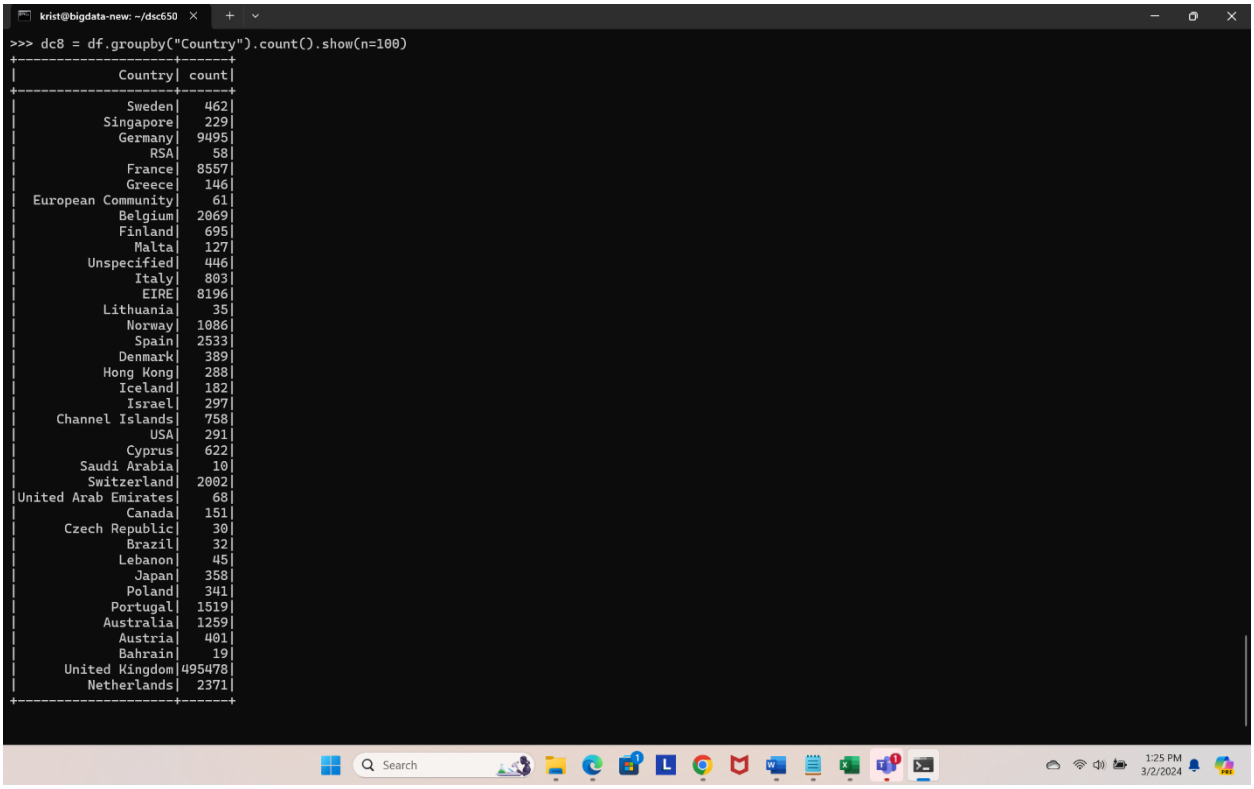
#7:


krist@bigdata-new: ~/dsc650
662 [Thread-4] WARN org.apache.hadoop.hive.conf.HiveConf - HiveConf of name hive.create.as.insert.only does not exist
Welcome to
version 3.0.0
Using Python version 3.7.10 (default, Mar 2 2021 09:06:08)
SparkSession available as 'spark'.
>>> df = spark.read.format("csv").option("header", "true").load("/data/data.csv")
>>> df.printSchema()
root
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: string (nullable = true)
|-- InvoiceDate: string (nullable = true)
|-- UnitPrice: string (nullable = true)
|-- CustomerID: string (nullable = true)
|-- Country: string (nullable = true)
>>> from pyspark.sql.types import IntegerType
>>> from pyspark.sql.types import DecimalType
>>> df = df.withColumn('NumQuant', df['Quantity'].cast(IntegerType()))
>>> df = df.withColumn('InvoiceNum', df['InvoiceNo'].cast(IntegerType()))
>>> df = df.withColumn('UnitPriceN', df['UnitPrice'].cast(DecimalType(12,2)))
>>> df = df.withColumn('CustomerIDN', df['CustomerID'].cast(IntegerType()))
>>> df.printSchema()
root
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: string (nullable = true)
|-- InvoiceDate: string (nullable = true)
|-- UnitPrice: string (nullable = true)
|-- CustomerID: string (nullable = true)
|-- Country: string (nullable = true)
|-- NumQuant: integer (nullable = true)
|-- InvoiceNum: integer (nullable = true)
|-- UnitPriceN: decimal(12,2) (nullable = true)
|-- CustomerIDN: integer (nullable = true)
>>>
Search
1:04 PM
3/2/2024

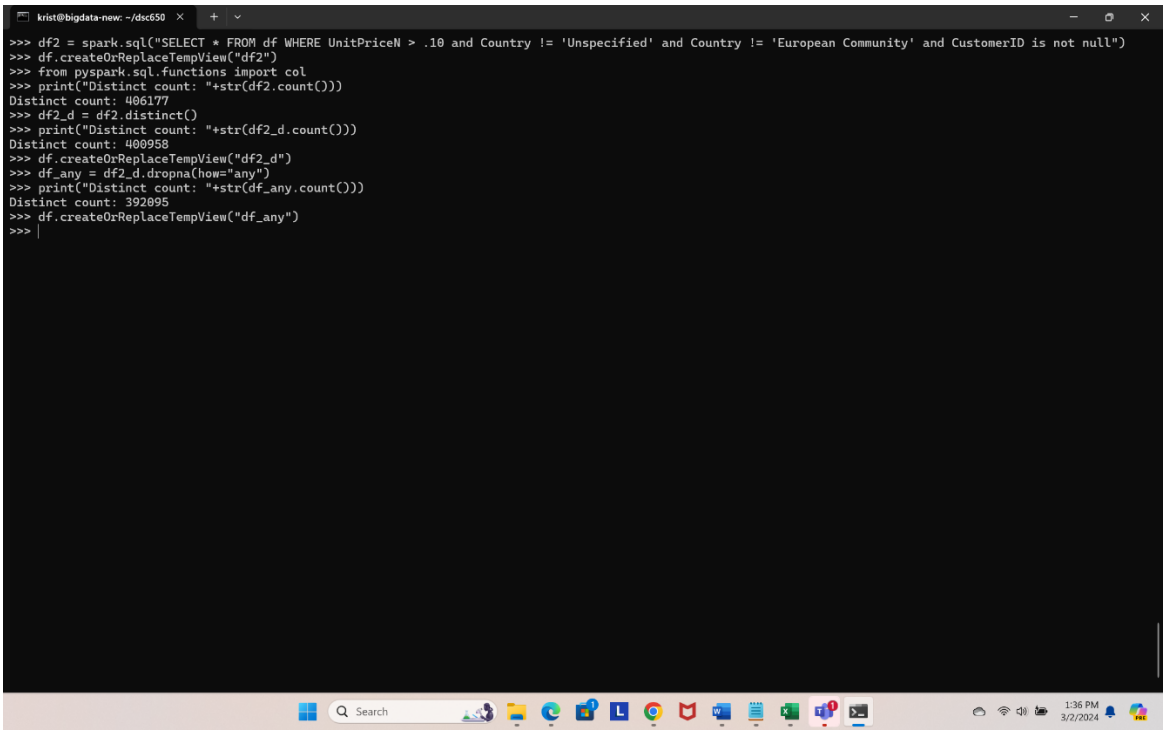
#8a:


krist@bigdata-new: ~/dsc650
>>> dc1 = df.groupby("InvoiceNo").count()
>>> dc1.tail(20)
[Row(InvoiceNo='573989', count=1), Row(InvoiceNo='574523', count=9), Row(InvoiceNo='575148', count=23), Row(InvoiceNo='C575651', count=6), Row(InvoiceNo='576088', count=8), Row(InvoiceNo='576765', count=1), Row(InvoiceNo='576896', count=3), Row(InvoiceNo='577078', count=572), Row(InvoiceNo='577116', count=1), Row(InvoiceNo='577531', count=198), Row(InvoiceNo='577833', count=9), Row(InvoiceNo='578063', count=14), Row(InvoiceNo='578107', count=7), Row(InvoiceNo='578121', count=21), Row(InvoiceNo='578344', count=622), Row(InvoiceNo='579272', count=1), Row(InvoiceNo='C579894', count=3), Row(InvoiceNo='580060', count=5), Row(InvoiceNo='580724', count=2), Row(InvoiceNo='C581178', count=2)]
>>> dc2 = df.groupby("StockCode").count()
>>> dc2.tail(20)
[Row(StockCode='22492', count=390), Row(StockCode='84906', count=15), Row(StockCode='22593', count=203), Row(StockCode='20774', count=49), Row(StockCode='17164B', count=10), Row(StockCode='85114C', count=78), Row(StockCode='22311', count=151), Row(StockCode='85226A', count=1), Row(StockCode='75013B', count=8), Row(StockCode='46115B', count=3), Row(StockCode='84595E', count=24), Row(StockCode='35610C', count=3), Row(StockCode='84661C', count=14), Row(StockCode='23185', count=33), Row(StockCode='84016', count=2), Row(StockCode='23083', count=253), Row(StockCode='90149', count=4), Row(StockCode='20782', count=28), Row(StockCode='84551', count=2), Row(StockCode='23424', count=137)]
>>> dc3 = df.groupby("Description").count()
>>> dc3.tail(20)
[Row(Description='MAGIC DRAWING SLATE PURDEY', count=162), Row(Description='BLACK AND WHITE CAT BOWL', count=49), Row(Description='3 WHITE CHOC MORRIS BOXED CANDLES', count=125), Row(Description='DOLLY HONEYCOMB GARLAND', count=26), Row(Description='EMPIRE GIFT WRAP', count=109), Row(Description='HARDMAN MUG 3 ASSORTED', count=14), Row(Description='FLAMES SUNGLASSES PINK LENSES', count=8), Row(Description='WHITE/PINK CHICK EASTER DECORATION', count=11), Row(Description='METAL RABBIT LADDER EASTER ', count=17), Row(Description='WHITE VINT ART DECO CRYSTAL NECKLAC', count=8), Row(Description='PANTRY SCRUBBING BRUSH', count=303), Row(Description='FRENCH CHATEAU OVAL PLATTER', count=6), Row(Description='JUMBO BAG ALPHABET', count=930), Row(Description='SET OF 12 MINI LOAF BAKING CASES', count=460), Row(Description='STORAGE TIN VINTAGE DOILEY ', count=2), Row(Description='IVORY WICKER HEART MEDIUM', count=250), Row(Description='SET 12 COLOUR PENCILS DOILEY', count=3), Row(Description='SET 10 CARDS DINKY TREE 17076', count=16), Row(Description='PINK HORSE SOCK PUPPET KIT', count=19), Row(Description='water damaged', count=1)]
>>> dc4 = df.groupby("Quantity").count().sort("Quantity")
>>> dc3.head(20)
[Row(Description='10 COLOUR SPACEBOY PEN', count=327), Row(Description='SET/10 BLUE POLKADOT PARTY CANDLES', count=249), Row(Description="POTTING SHED SOW 'N' GROW SET", count=2), Row(Description='PAPERWEIGHT KINGS CHOICE ', count=24), Row(Description='WOVEN BERRIES CUSHION COVER ', count=89), Row(Description='WHITE/PINK MINI CRYSTALS NECKLACE', count=7), Row(Description='SET/3 RED GINGHAM ROSE STORAGE BOX', count=494), Row(Description='MAGNETS PACK OF 4 VINTAGE LABELS ', count=128), Row(Description='WHITE CHRYSANTHEMUMS ART FLOWER', count=9), Row(Description='WHITE FRANGIPANI NECKLACE', count=18), Row(Description='SILVER FABRIC MIRROR', count=46), Row(Description='PINK HONEYCOMB PAPER FAN', count=70), Row(Description='PINK BOUDOIR T-LIGHT HOLDER', count=1), Row(Description='BLACK CHERRY LIGHTS', count=3), Row(Description='GLASS CAKE COVER AND PLATE', count=2), Row(Description='DECORATION SITTING BUNNY', count=86), Row(Description='ANTIQUE MID BLUE FLOWER EARRINGS', count=3), Row(Description='VINTAGE LEAF CHOPPING BOARD ', count=47), Row(Description='SNACK TRAY I LOVE LONDON', count=37), Row(Description='DECROTIVEVINTAGE COFFEE GRINDER BOX', count=6)]
>>> dc4.head(20)
[Row(Quantity='-1', count=4184), Row(Quantity='-10', count=197), Row(Quantity='-100', count=60), Row(Quantity='-1000', count=1), Row(Quantity='-101', count=2), Row(Quantity='-102', count=4), Row(Quantity='-103', count=1), Row(Quantity='-104', count=2), Row(Quantity='-105', count=1), Row(Quantity='-1050', count=1), Row(Quantity='-106', count=1), Row(Quantity='-1060', count=1), Row(Quantity='-108', count=8), Row(Quantity='-109', count=2), Row(Quantity='-1092', count=1), Row(Quantity='-11', count=62), Row(Quantity='-110', count=12), Row(Quantity='-1100', count=1), Row(Quantity='-111', count=2), Row(Quantity='-112', count=1)]
>>>
Search
1:20 PM
3/2/2024

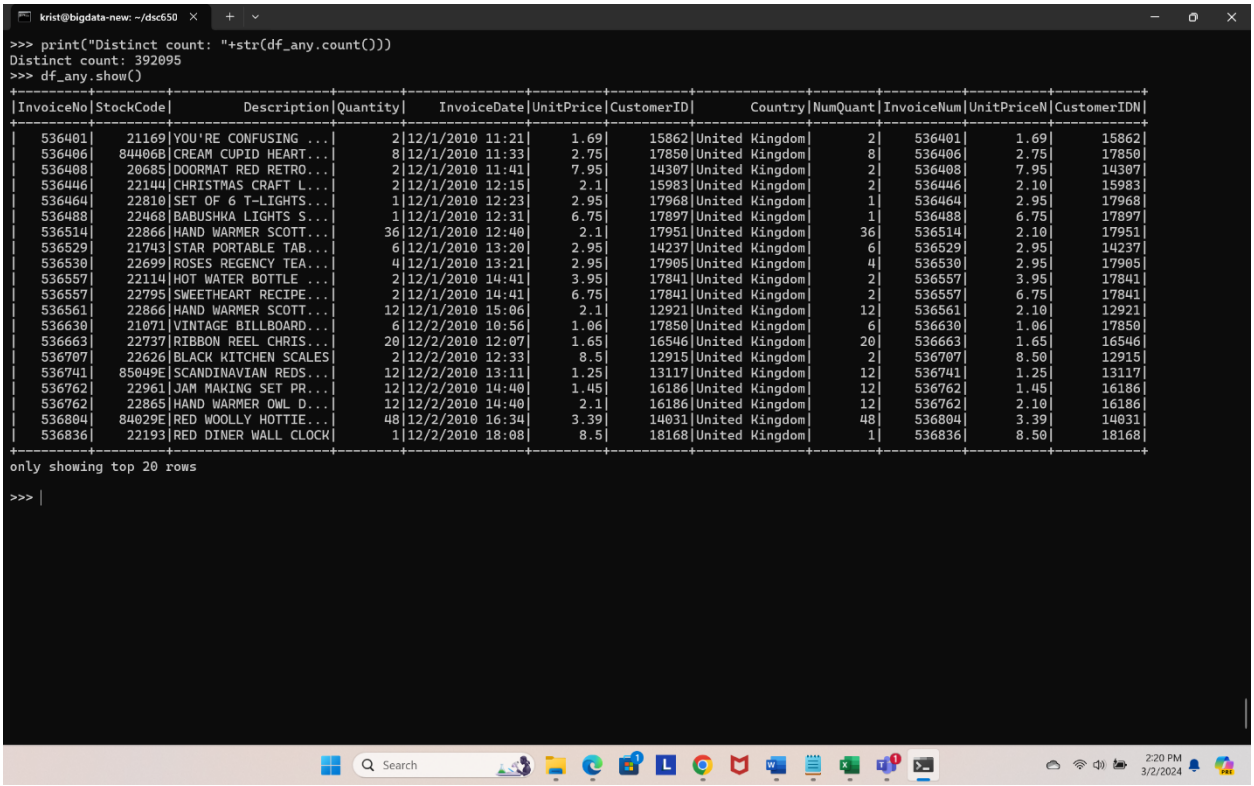
#8b:


krist@bigdata-new: ~/dsc650
>>> dc8 = df.groupby("Country").count().show(n=100)
+--------------------+------+
| Country| count|
+--------------------+------+
Sweden	462
Singapore	229
Germany	9495
RSA	58
France	8557
Greece	146
European Community	61
Belgium	2069
Finland	695
Malta	127
Unspecified	446
Italy	803
EIRE	8196
Lithuania	35
Norway	1086
Spain	2533
Denmark	389
Hong Kong	288
Iceland	182
Israel	297
Channel Islands	758
USA	291
Cyprus	622
Saudi Arabia	10
Switzerland	2002
United Arab Emirates	68
Canada	151
Czech Republic	30
Brazil	32
Lebanon	45
Japan	358
Poland	341
Portugal	1519
Australia	1259
Austria	401
Bahrain	19
United Kingdom	495478
Netherlands	2371
+--------------------+------+
Search
1:25 PM
3/2/2024

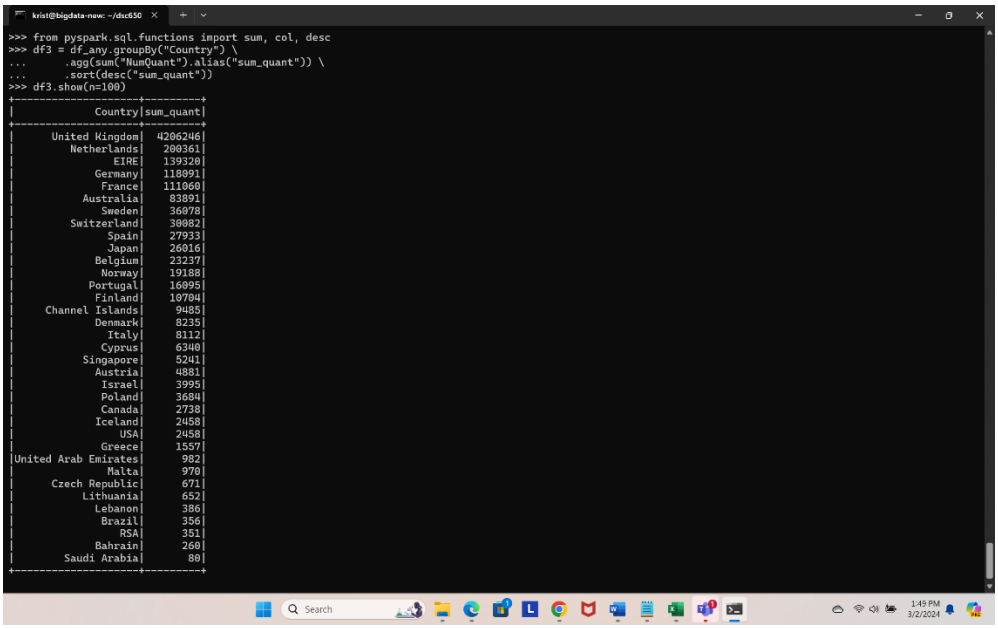
#8c:


krist@bigdata-new: ~/dsc650
>>> df2 = spark.sql("SELECT * FROM df WHERE UnitPriceN > .10 and Country != 'Unspecified' and Country != 'European Community' and CustomerID is not null")
>>> df.createOrReplaceTempView("df2")
>>> from pyspark.sql.functions import col
>>> print("Distinct count: "+str(df2.count()))
Distinct count: 406177
>>> df2_d = df2.distinct()
>>> print("Distinct count: "+str(df2_d.count()))
Distinct count: 400958
>>> df.createOrReplaceTempView("df2_d")
>>> df_any = df2_d.dropna(how="any")
>>> print("Distinct count: "+str(df_any.count()))
Distinct count: 392095
>>> df.createOrReplaceTempView("df_any")
>>>
Search
1:36 PM
3/2/2024

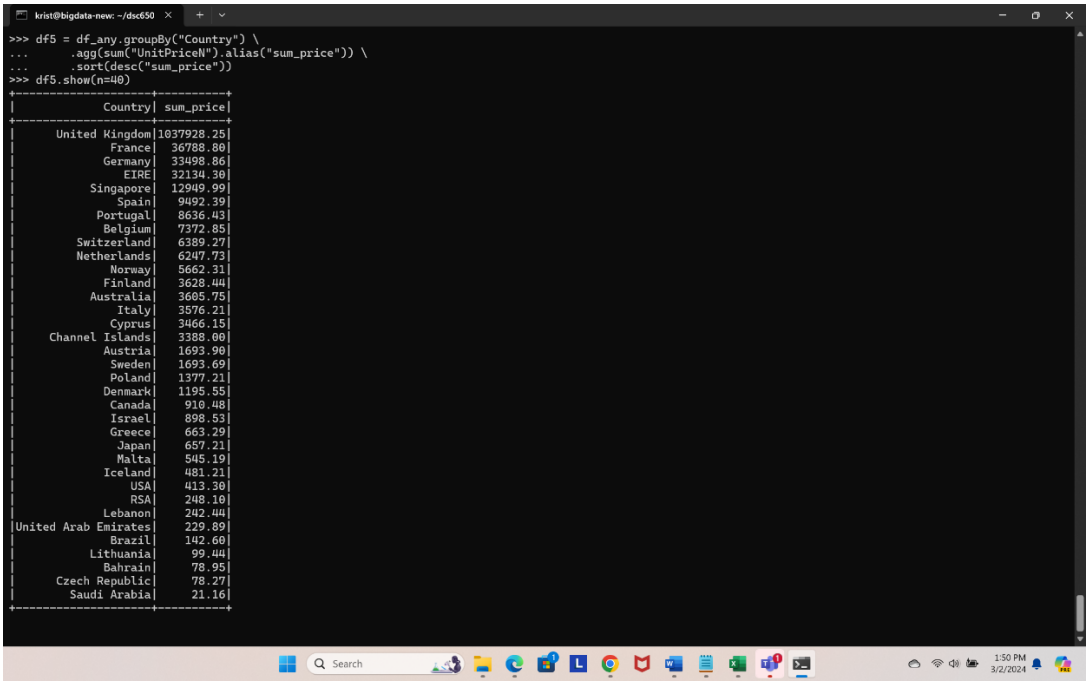
#9:


krist@bigdata-new: ~/dsc650
>>> print("Distinct count: "+str(df_any.count()))
Distinct count: 392095
>>> df_any.show()
InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	NumQuant	InvoiceNum	UnitPriceN	CustomerIDN
536401	21169	YOU'RE CONFUSING ...	2	12/1/2010 11:21	1.69	15862	United Kingdom	2	536401	1.69	15862
536406	84406B	CREAM CUPID HEART...	8	12/1/2010 11:33	2.75	17850	United Kingdom	8	536406	2.75	17850
536408	20685	DOORMAT RED RETRO...	2	12/1/2010 11:41	7.95	14307	United Kingdom	2	536408	7.95	14307
536446	22144	CHRISTMAS CRAFT L...	2	12/1/2010 12:15	2.1	15983	United Kingdom	2	536446	2.10	15983
536464	22810	SET OF 6 T-LIGHTS...	1	12/1/2010 12:23	2.95	17968	United Kingdom	1	536464	2.95	17968
536488	22468	BABUSHKA LIGHTS S...	1	12/1/2010 12:31	6.75	17897	United Kingdom	1	536488	6.75	17897
536514	22866	HAND WARMER SCOTT...	36	12/1/2010 12:40	2.1	17951	United Kingdom	36	536514	2.10	17951
536529	21743	STAR PORTABLE TAB...	6	12/1/2010 13:20	2.95	14237	United Kingdom	6	536529	2.95	14237
536530	22699	ROSES REGENCY TEA...	4	12/1/2010 13:21	2.95	17905	United Kingdom	4	536530	2.95	17905
536557	22114	HOT WATER BOTTLE ...	2	12/1/2010 14:41	3.95	17841	United Kingdom	2	536557	3.95	17841
536557	22795	SWEETHEART RECIPE...	2	12/1/2010 14:41	6.75	17841	United Kingdom	2	536557	6.75	17841
536561	22866	HAND WARMER SCOTT...	12	12/1/2010 15:06	2.1	12921	United Kingdom	12	536561	2.10	12921
536630	21071	VINTAGE BILLBOARD...	6	12/2/2010 10:56	1.06	17850	United Kingdom	6	536630	1.06	17850
536663	22737	RIBBON REEL CHRIS...	20	12/2/2010 12:07	1.65	16546	United Kingdom	20	536663	1.65	16546
536707	22626	BLACK KITCHEN SCALES	2	12/2/2010 12:33	8.5	12915	United Kingdom	2	536707	8.50	12915
536741	85049E	SCANDINAVIAN REDS...	12	12/2/2010 13:11	1.25	13117	United Kingdom	12	536741	1.25	13117
536762	22961	JAM MAKING SET PR...	12	12/2/2010 14:40	1.45	16186	United Kingdom	12	536762	1.45	16186
536762	22865	HAND WARMER OWL D...	12	12/2/2010 14:40	2.1	16186	United Kingdom	12	536762	2.10	16186
536804	84029E	RED WOOLLY HOTTIE...	48	12/2/2010 16:34	3.39	14031	United Kingdom	48	536804	3.39	14031
536836	22193	RED DINER WALL CLOCK	1	12/2/2010 18:08	8.5	18168	United Kingdom	1	536836	8.50	18168
only showing top 20 rows
>>>
Search
2:20 PM
3/2/2024

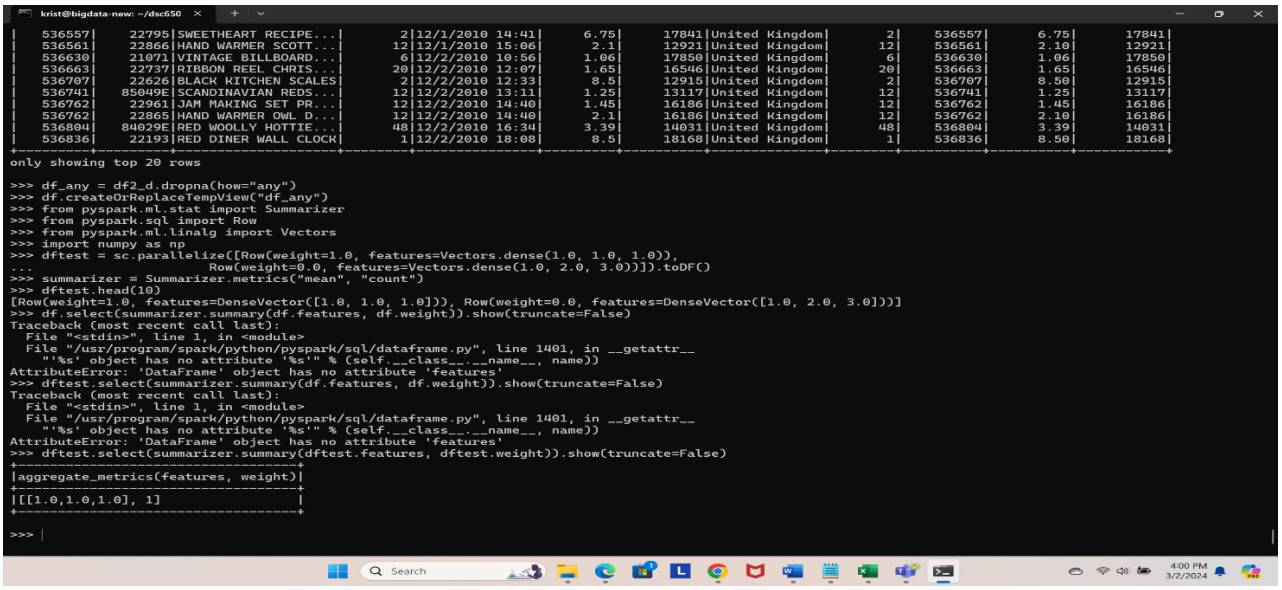
#10:


krist@bigdata-new: ~/dsc650
>>> from pyspark.sql.functions import sum, col, desc
>>> df3 = df_any.groupBy("Country") \
... .agg(sum("NumQuant").alias("sum_quant")) \
... .sort(desc("sum_quant"))
>>> df3.show(n=100)
+--------------------+---------+
| Country|sum_quant|
+--------------------+---------+
United Kingdom	42062246
Netherlands	200361
EIRE	139320
Germany	118091
France	111060
Australia	83891
Sweden	36078
Switzerland	30082
Spain	27933
Japan	26016
Belgium	23237
Norway	19188
Portugal	16095
Finland	10704
Channel Islands	9485
Denmark	8235
Italy	8112
Cyprus	6340
Singapore	5241
Austria	4881
Israel	3995
Poland	3684
Canada	2738
Iceland	2458
USA	2458
Greece	1557
United Arab Emirates	982
Malta	970
Czech Republic	671
Lithuania	652
Lebanon	386
Brazil	356
RSA	351
Bahrain	260
Saudi Arabia	80
+--------------------+---------+
Search
1:49 PM
3/2/2024

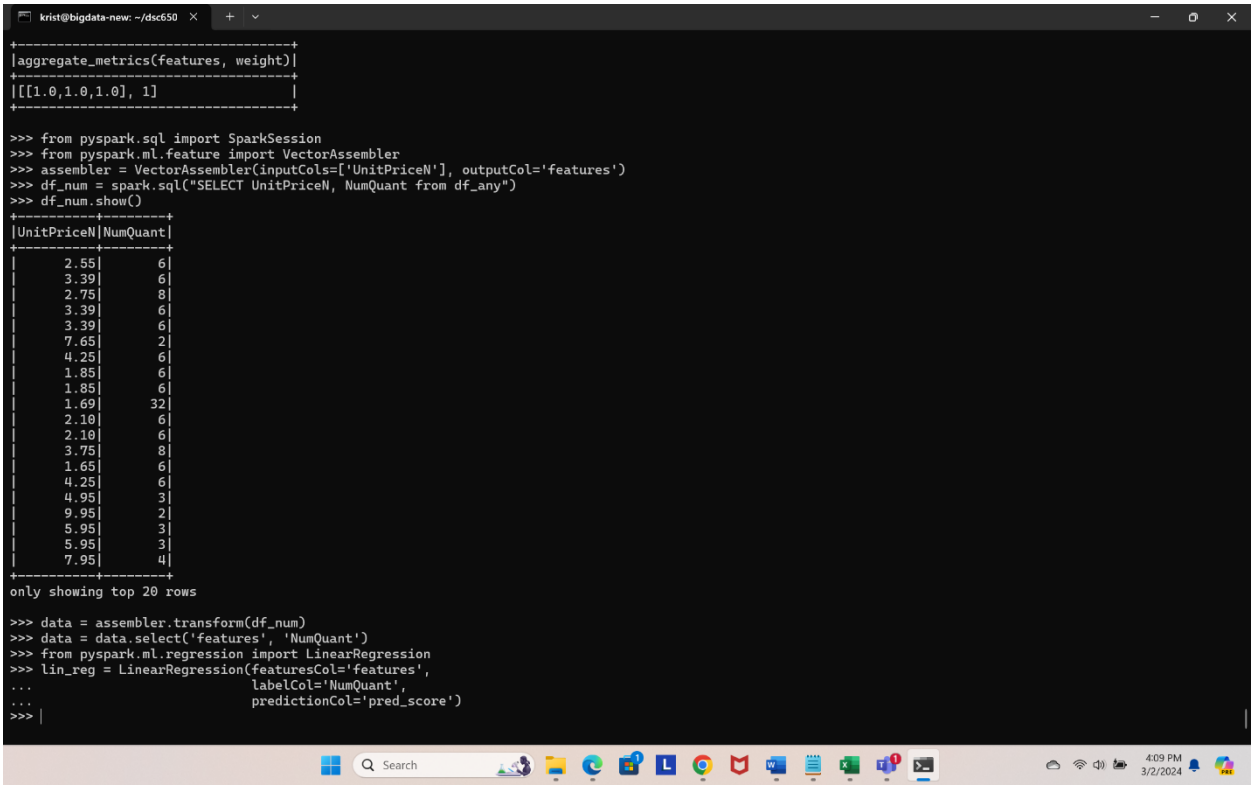
#11:


krist@bigdata-new: ~/dsc650
>>> df5 = df_any.groupBy("Country") \
... .agg(sum("UnitPriceN").alias("sum_price")) \
... .sort(desc("sum_price"))
>>> df5.show(n=40)
+--------------------+----------+
| Country| sum_price|
+--------------------+----------+
United Kingdom	1037928.25
France	36788.80
Germany	33498.86
EIRE	32134.30
Singapore	12949.99
Spain	9492.39
Portugal	8636.43
Belgium	7372.85
Switzerland	6389.27
Netherlands	6247.73
Norway	5662.31
Finland	3628.44
Australia	3605.75
Italy	3576.21
Cyprus	3466.15
Channel Islands	3388.00
Austria	1693.90
Sweden	1693.69
Poland	1377.21
Denmark	1195.55
Canada	910.48
Israel	898.53
Greece	663.29
Japan	657.21
Malta	545.19
Iceland	481.21
USA	413.30
RSA	248.10
Lebanon	242.44
United Arab Emirates	229.89
Brazil	142.60
Lithuania	99.44
Bahrain	78.95
Czech Republic	78.27
Saudi Arabia	21.16
+--------------------+----------+
Search
1:50 PM
3/2/2024

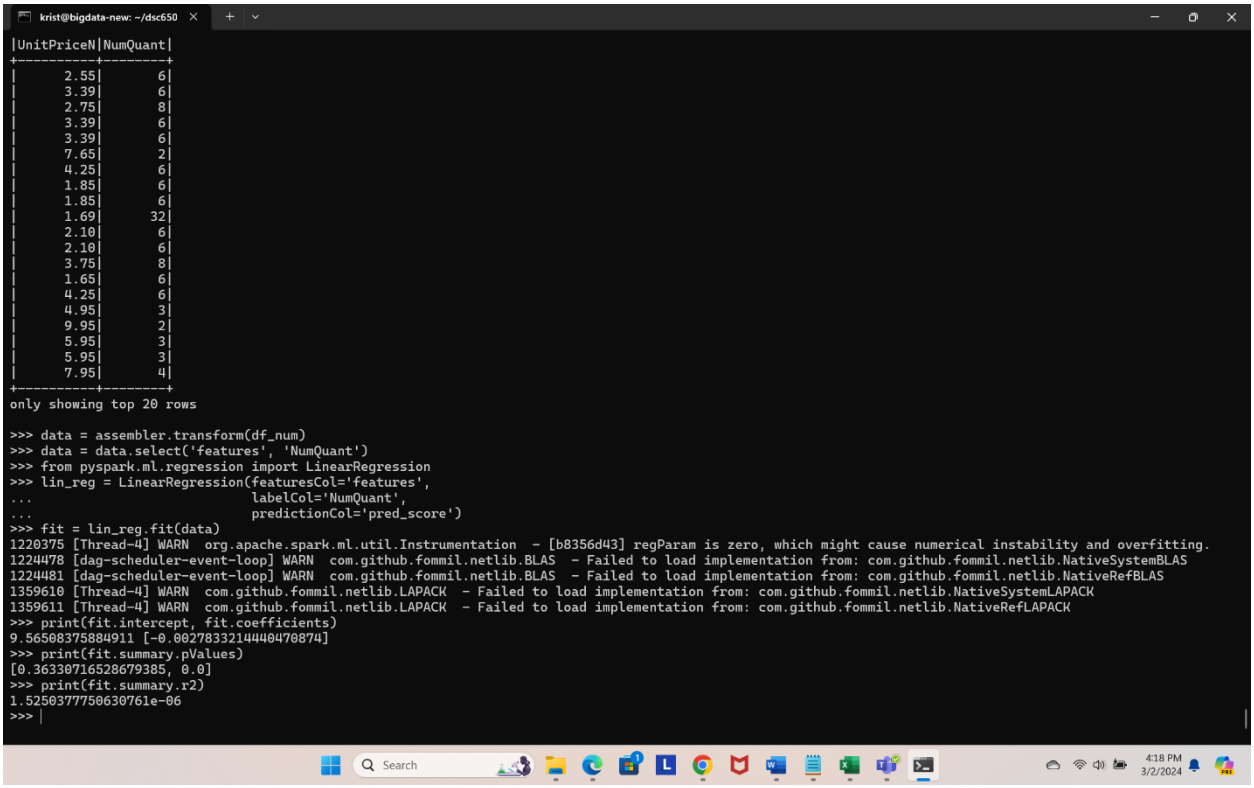
#12:


krist@bigdata-new: ~/dsc650
536557	22795	SWEETHEART RECIPE...	2	12/1/2010 14:41	6.75	17841	United Kingdom	2	536557	6.75	17841
536561	22866	HAND WARMER SCOTT...	12	12/1/2010 15:06	2.1	12921	United Kingdom	12	536561	2.10	12921
536630	21071	VINTAGE BILLBOARD...	6	12/2/2010 10:56	1.06	17850	United Kingdom	6	536630	1.06	17850
536663	22737	RIBBON REEL CHRIS...	20	12/2/2010 12:07	1.65	16546	United Kingdom	20	536663	1.65	16546
536707	22626	BLACK KITCHEN SCALES	2	12/2/2010 12:33	8.5	12915	United Kingdom	2	536707	8.50	12915
536741	85049E	SCANDINAVIAN REDS...	12	12/2/2010 13:11	1.25	13117	United Kingdom	12	536741	1.25	13117
536762	22961	JAM MAKING SET PR...	12	12/2/2010 14:40	1.45	16186	United Kingdom	12	536762	1.45	16186
536762	22865	HAND WARMER OWL D...	12	12/2/2010 14:40	2.1	16186	United Kingdom	12	536762	2.10	16186
536804	84029E	RED WOOLLY HOTTIE...	48	12/2/2010 16:34	3.39	14031	United Kingdom	48	536804	3.39	14031
536836	22193	RED DINER WALL CLOCK	1	12/2/2010 18:08	8.5	18168	United Kingdom	1	536836	8.50	18168
only showing top 20 rows
>>> df_any = df2_d.dropna(how="any")
>>> df.createOrReplaceTempView("df_any")
>>> from pyspark.ml.stat import Summarizer
>>> from pyspark.sql import Row
>>> from pyspark.ml.linalg import Vectors
>>> import numpy as np
>>> dftest = sc.parallelize([Row(weight=1.0, features=Vectors.dense(1.0, 1.0, 1.0)),
...   Row(weight=0.0, features=Vectors.dense(1.0, 2.0, 3.0))]).toDF()
>>> summarizer = Summarizer.metrics("mean", "count")
>>> dftest.head(10)
[Row(weight=1.0, features=DenseVector([1.0, 1.0, 1.0])), Row(weight=0.0, features=DenseVector([1.0, 2.0, 3.0]))]
>>> df.select(summarizer.summary(df.features, df.weight)).show(truncate=False)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/program/spark/python/pyspark/sql/dataframe.py", line 1401, in __getattr__
    "'%s' object has no attribute '%s'" % (self.__class__.__name__, name))
AttributeError: 'DataFrame' object has no attribute 'features'
>>> dftest.select(summarizer.summary(df.features, df.weight)).show(truncate=False)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/program/spark/python/pyspark/sql/dataframe.py", line 1401, in __getattr__
    "'%s' object has no attribute '%s'" % (self.__class__.__name__, name))
AttributeError: 'DataFrame' object has no attribute 'features'
>>> dftest.select(summarizer.summary(dftest.features, dftest.weight)).show(truncate=False)
+-----------------------------------+
|aggregate_metrics(features, weight)|
+-----------------------------------+
|[[1.0,1.0,1.0], 1] |
+-----------------------------------+
>>>
Search
4:00 PM
3/2/2024

#13:


krist@bigdata-new: ~/dsc650
+-----------------------------------+
|aggregate_metrics(features, weight)|
+-----------------------------------+
|[[1.0,1.0,1.0], 1] |
+-----------------------------------+
>>> from pyspark.sql import SparkSession
>>> from pyspark.ml.feature import VectorAssembler
>>> assembler = VectorAssembler(inputCols=['UnitPriceN'], outputCol='features')
>>> df_num = spark.sql("SELECT UnitPriceN, NumQuant from df_any")
>>> df_num.show()
+----------+--------+
|UnitPriceN|NumQuant|
+----------+--------+
2.55	6
3.39	6
2.75	8
3.39	6
3.39	6
7.65	2
4.25	6
1.85	6
1.85	6
1.69	32
2.10	6
2.10	6
3.75	8
1.65	6
4.25	6
4.95	3
9.95	2
5.95	3
5.95	3
7.95	4
+----------+--------+
only showing top 20 rows
>>> data = assembler.transform(df_num)
>>> data = data.select('features', 'NumQuant')
>>> from pyspark.ml.regression import LinearRegression
>>> lin_reg = LinearRegression(featuresCol='features',
...   labelCol='NumQuant',
...   predictionCol='pred_score')
>>>
Search
4:09 PM
3/2/2024

#14:


krist@bigdata-new: ~/dsc650
|UnitPriceN|NumQuant|
+----------+--------+
2.55	6
3.39	6
2.75	8
3.39	6
3.39	6
7.65	2
4.25	6
1.85	6
1.85	6
1.69	32
2.10	6
2.10	6
3.75	8
1.65	6
4.25	6
4.95	3
9.95	2
5.95	3
5.95	3
7.95	4
+----------+--------+
only showing top 20 rows
>>> data = assembler.transform(df_num)
>>> data = data.select('features', 'NumQuant')
>>> from pyspark.ml.regression import LinearRegression
>>> lin_reg = LinearRegression(featuresCol='features',
... labelCol='NumQuant',
... predictionCol='pred_score')
>>> fit = lin_reg.fit(data)
1220375 [Thread-4] WARN org.apache.spark.ml.util.Instrumentation - [b8356d43] regParam is zero, which might cause numerical instability and overfitting.
1224478 [dag-scheduler-event-loop] WARN com.github.fommil.netlib.BLAS - Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
1224481 [dag-scheduler-event-loop] WARN com.github.fommil.netlib.BLAS - Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
1359610 [Thread-4] WARN com.github.fommil.netlib.LAPACK - Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
1359611 [Thread-4] WARN com.github.fommil.netlib.LAPACK - Failed to load implementation from: com.github.fommil.netlib.NativeRefLAPACK
>>> print(fit.intercept, fit.coefficients)
9.56508375884911 [-0.0027833214440470874]
>>> print(fit.summary.pValues)
[0.36330716528679385, 0.0]
>>> print(fit.summary.r2)
1.5250377750630761e-06
>>>
Search
4:18 PM
3/2/2024

References

Basic Statistics (accessed 2024, March). Basic Statistics - Spark 3.5.0 Documentation (apache.org)

Convert CSV to JSON, Avro, XML using ConvertRecord (Apache NiFi 1.2+) (Lim, A., 2017, July). My Cloudera. https://community.cloudera.com/t5/Community-Articles/Convert-CSV-to-JSON-Avro-XML-using-ConvertRecord-Apache-NiFi/ta-p/246607.

E-Commerce Data (accessed 2024, March). Actual transactions from UK retailer. https://www.kaggle.com/datasets/carrie1/ecommerce-data.

How to Interpret P-Values in Linear Regression (With Example) (accessed 2024, March). Statology. How to Interpret P-Values in Linear Regression (With Example) - Statology.

How to Perform Linear Regression in PySpark (With Example) (accessed 2024, March). How to Perform Linear Regression in PySpark (With Example) - Statology.

Key differences between JSON and CSV (accessed 2024, March). JSON Editor Online. https://jsoneditoronline.org/indepth/compare/json-vs-csv/#:~:text=A%20note%20about%20the%20data,keys%20in%20a%20JSON%20file.