

Coding Challenge – 4

Creating a Cluster, Notebook, Transformations and Visualizations

Name: Aathirainathan P

Date: 10-12-2024

a) Create a cluster & Attach the notebook to the cluster and run all commands in the notebook & creates a DataFrame from a Databricks dataset & Create a Visualizations in Databricks notebooks.

Step 1: Create a cluster:

To create a cluster:

- *Log in to Azure Databricks.*
- *Go to the Compute section and click on Create Cluster.*
- *Fill out the cluster details:*
 - **Cluster Name:** azuser2356_mml.local's Cluster
 - **Databricks Runtime Version:** 15.4 LTS(Scala 2.12, Spark 3.5.0)
 - **Node Type:** Single Node, Standard_D4ads_v5
- *Click Create Cluster and wait until the status is "Running."*

The screenshot shows the 'New compute' page in the Azure Databricks interface. The cluster is named 'azuser2356_mml.local's Cluster'. The configuration is set to 'Single node' with 'Single user access'. The 'Access mode' is 'Single user' and the 'Cluster name' is 'azuser2356_mml.local'. The 'Performance' section shows 'Databricks runtime version' as 'Runtime: 15.4 LTS (Scala 2.12, Spark 3.5.0)'. The 'Node type' is 'Standard_D4ads_v5' with '16 GB Memory, 4 Cores'. The 'Terminate after' checkbox is checked, set to '120 minutes of inactivity'. The 'Tags' section is empty. A 'Summary' panel on the right shows: 1 Driver, 16 GB Memory, 4 Cores; Runtime: 15.4.x-scala2.12; Standard_D4ads_v5, 1 DBU/h. The 'Create compute' button is visible at the bottom.

Microsoft Azure | databricks

Search data, notebooks, recents, and more... CTRL + P

hexaworkspace

+ New

Workspace

Recents

Catalog

Workflows

Compute

Data Engineering

Job Runs

Machine Learning

Playground

Experiments

Features

Models

Serving

Partner Connect

Compute > New compute

azuser2356_mml.local's Cluster

☐ Multi node ☒ Single node

Access mode ☐ Single user access

Single user azuser2356_mml.local

Performance

Databricks runtime version

Runtime: 15.4 LTS (Scala 2.12, Spark 3.5.0)

☐ Use Photon Acceleration

Node type

Standard_D4ads_v5 16 GB Memory, 4 Cores

☒ Terminate after 120 minutes of inactivity

Tags

Add tags

Key Value Add

Create compute Cancel

Summary

1 Driver 16 GB Memory, 4 Cores

Runtime 15.4.x-scala2.12

Standard_D4ads_v5 1 DBU/h

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P

hexaworkspace

+ New

Workspace

Recents

Catalog

Workflows

Compute

Data Engineering

Job Runs

Machine Learning

Playground

Experiments

Features

Models

Serving

Partner Connect

Compute

All-purpose compute Job compute Pools

Filter compute you have access to Created by Only pinned Create compute

State	Name	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator	Notebooks
Running	azuser2356_mmlLocal's Cluster	15.4	16 GB	4 cores	1	UI	azuser2356_mmlLocal	1

< Previous Next > 20 / page

Step 2: Create a Notebook

To create a notebook:

- Create a Notebook
- Go to the Workspace section.
- Click Create → Notebook.
- Name your notebook.
- Select the language as Python

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P

hexaworkspace

+ New

Workspace

Recents

Catalog

Workflows

Compute

Data Engineering

Job Runs

Machine Learning

Playground

Experiments

Features

Models

Serving

Partner Connect

Coding Challenge - 4 Aathirainathan P

Python

File Edit View Run Help Last edit was now

Run all azuser2356_mmlLocal's... Schedule Share

Just now (<1s)

Python

```
df_lineitem: pyspark.sql.dataframe.DataFrame = [(l_orderkey: long, l_partkey: long ... 14 more fields)]
```

2

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts

88°F 16:26

Step 3: Attach the cluster to the notebook:

To attach the cluster to the notebook:

- *Open the notebook.*
- *In the top toolbar, use the Cluster dropdown menu to attach the notebook to the cluster created.*

The screenshot displays the Databricks notebook interface. At the top, the header shows 'Microsoft Azure' and 'databricks' logos, a search bar, and a 'hexaworkspace' dropdown. The notebook title is 'Coding Challenge - 4 Aathirainathan P' with a 'Python' language selector. The top toolbar includes a 'Run all' button, a cluster dropdown menu currently showing 'azuser2356_mml.local's...', and 'Schedule' and 'Share' buttons. The left sidebar contains navigation options: 'New', 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'Data Engineering', 'Job Runs', 'Machine Learning', 'Playground', 'Experiments', 'Features', 'Models', 'Serving', and 'Partner Connect'. The main workspace area shows two code cells. The first cell, labeled '1', contains a PySpark DataFrame definition and has a status bar indicating it was executed 'Just now (<1s)'. The second cell, labeled '2', is currently empty. A dropdown menu is open from the cluster selector, showing 'Connected' status and a 'Go to last run cell' link. Under 'Recent resources', it lists 'azuser2356_mml.local's Cluster' with details: 'DBR 15.4 LTS • Spark 3.5.0 • Scala 2.12' and 'Standard_D4ads_v5 • 16 GB • 4 Cores'. Below this, it shows 'azuser2356_mml.local's Cluster' with 'DBR 15.4 LTS'. At the bottom of the menu are 'More...' and 'Create new resource...' options. Keyboard shortcuts are visible at the bottom of the notebook area: '[Shift+Enter] to run and move to next cell', '[Ctrl+Shift+P] to open the command palette', and '[Esc H] to see all keyboard shortcuts'.

Step 4: Create a Dataframe from a Databricks Dataset (line item):

▶

✓

Just now (2s)

1

Python

✦

⌵

⋮

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Coding Challenge - 4").getOrCreate()

# Load lineitem table as a DataFrame
df_lineitem = spark.read.format("delta").load("/databricks-datasets/tpch/delta-001/lineitem")

# Display the DataFrame
df_lineitem.show(5)
```

▶ (1) Spark Jobs

▶ (1) Spark Jobs

▶ df_lineitem: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenum	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode	l_comment
15997987	295335	20346	4	50.00	66516.00	0.01	0.08	A	F	1992-02-12	1992-04-22	1992-03-10	TAKE BACK RETURN	REG AIR	even pinto beans...
15997988	332059	19578	1	49.00	53460.96	0.08	0.03	A	F	1994-05-31	1994-06-20	1994-06-22	NONE	FOB	detect carefully...
15997988	904286	4287	2	37.00	47738.88	0.07	0.05	A	F	1994-05-24	1994-05-28	1994-06-03	COLLECT COD	RAIL	the slyly e
15997988	327898	15417	3	13.00	25036.44	0.09	0.08	A	F	1994-06-09	1994-06-28	1994-06-29	NONE	FOB	ss courts. fina
15997988	973125	48164	4	29.00	34744.32	0.10	0.02	A	F	1994-08-05	1994-06-26	1994-08-17	COLLECT COD	RAIL	sly final packages

only showing top 5 rows

Step 5: Run all the transformation Queries:

1. Select Specific Columns:

▶

✓

04:34 PM (1s)

2

Python

✦

⌵

⋮

```
#Select Specific Columns
df_selected = df_lineitem.select("l_orderkey", "l_quantity", "l_extendedprice")
df_selected.show(5)
```

▶ (1) Spark Jobs

▶ df_selected: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_quantity: decimal(18,2) ... 1 more field]

l_orderkey	l_quantity	l_extendedprice
15997987	50.00	66516.00
15997988	49.00	53460.96
15997988	37.00	47738.88
15997988	13.00	25036.44
15997988	29.00	34744.32

only showing top 5 rows

2. Filter rows where the quantity is greater than 10:

▶

04:35 PM (<1s)

3

Python

```
#Filter rows where the quantity is greater than 10:
df_filtered = df_lineitem.filter(df_lineitem.l_quantity > 10)
df_filtered.show(5)
```

▶ (1) Spark Jobs

df_filtered: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenum	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode	l_comment
15997987	295335	20346	4	50.00	66516.00	0.01	0.08	A	F	1992-02-12	1992-04-22	1992-03-10	TAKE BACK RETURN	REG AIR	even pinto beans...
15997988	332059	19578	1	49.00	53460.96	0.08	0.03	A	F	1994-05-31	1994-06-20	1994-06-22	NONE	FOB	detect carefully...
15997988	904286	4287	2	37.00	47738.88	0.07	0.05	A	F	1994-05-24	1994-05-28	1994-06-03	COLLECT COD	RAIL	the slyly e
15997988	327898	15417	3	13.00	25036.44	0.09	0.08	A	F	1994-06-09	1994-06-28	1994-06-29	NONE	FOB	ss courts. fina
15997988	973125	48164	4	29.00	34744.32	0.10	0.02	A	F	1994-08-05	1994-06-26	1994-08-17	COLLECT COD	RAIL	sly final packages

only showing top 5 rows

3. Group By and Aggregate (Calculate total revenue for each l_shipmode):

▶

04:36 PM (5s)

4

Python

```
from pyspark.sql.functions import sum
#Group By and Aggregate
#Calculate total revenue for each l_shipmode:

df_grouped = df_lineitem.groupBy("l_shipmode") \
    .agg((sum(df_lineitem.l_extendedprice * (1 - df_lineitem.l_discount))).alias("total_revenue"))
df_grouped.show()
```

▶ (2) Spark Jobs

df_grouped: pyspark.sql.dataframe.DataFrame = [l_shipmode: string, total_revenue: decimal(38,4)]

l_shipmode	total_revenue
AIR	155756263830.2007
MAIL	155626832178.4431
RAIL	155650102804.4504
SHIP	155659562319.9524
TRUCK	155703299295.9749
REG AIR	155782357803.3855
FOB	155656761014.8085

4. Add a New Column:

04:37 PM (1s)

5

Python

#Add a New Column

df_with_total_price = df_lineitem.withColumn("total_price", df_lineitem.l_extendedprice * (1 - df_lineitem.l_discount))

df_with_total_price.show(5)

(1) Spark Jobs

df_with_total_price: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 15 more fields]

l_orderkey|l_partkey|l_suppkey|l_linenum|l_quantity|l_extendedprice|l_discount|l_tax|l_returnflag|l_linestatus|l_shipdate|l_commitdate|l_receiptdate|l_shipinstruct|l_shipmode|l_comment|total_price|

|15997987|295335|20346|4|50.00|66516.00|0.01|0.08|A|F|1992-02-12|1992-04-22|1992-03-10|TAKE BACK RETURN|REG AIR|even pinto beans...|65850.8400|

|15997988|332059|19578|1|49.00|53460.96|0.08|0.03|A|F|1994-05-31|1994-06-20|1994-06-22|NONE|FOB|detect carefully...|49184.0832|

|15997988|904286|4287|2|37.00|47738.88|0.07|0.05|A|F|1994-05-24|1994-05-28|1994-06-03|COLLECT COD|RAIL|the slyly e|44397.1584|

|15997988|327898|15417|3|13.00|25036.44|0.09|0.08|A|F|1994-06-09|1994-06-28|1994-06-29|NONE|FOB|ss courts. fina|22783.1604|

|15997988|973125|48164|4|29.00|34744.32|0.10|0.02|A|F|1994-08-05|1994-06-26|1994-08-17|COLLECT COD|RAIL|sly final packages |31269.8880|

only showing top 5 rows

5. Sort Data:

04:37 PM (13s)

6

Python

#Sort Data

df_sorted = df_lineitem.orderBy("l_quantity", ascending=False)

df_sorted.show(5)

(1) Spark Jobs

df_sorted: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

l_orderkey|l_partkey|l_suppkey|l_linenum|l_quantity|l_extendedprice|l_discount|l_tax|l_returnflag|l_linestatus|l_shipdate|l_commitdate|l_receiptdate|l_shipinstruct|l_shipmode|l_comment|

|15998113|943289|5808|2|50.00|66612.00|0.08|0.02|N|O|1998-09-24|1998-08-18|1998-10-02|TAKE BACK RETURN|AIR|al packages. blit...|

|25245639|336791|36792|3|50.00|91389.00|0.10|0.07|N|O|1995-12-03|1996-01-09|1995-12-22|NONE|MAIL|ly final pinto be...|

|13070276|434679|9696|2|50.00|80682.50|0.03|0.07|N|O|1997-02-18|1997-02-14|1997-03-11|COLLECT COD|MAIL|fully regul|

|25245922|153777|3778|1|50.00|91538.50|0.00|0.00|N|O|1996-07-07|1996-08-29|1996-07-12|NONE|REG AIR|s. even id|

|263|714455|1998|3|50.00|73471.00|0.06|0.04|R|F|1994-08-18|1994-07-31|1994-08-22|NONE|TRUCK|re the packages. ...|

only showing top 5 rows

6. Drop a Column:

04:38 PM (<1s)

7

Python

#Drop a Column

df_dropped = df_lineitem.drop("l_comment")

df_dropped.show(5)

(1) Spark Jobs

df_dropped: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 13 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenumber	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode
15997987	295335	20346	4	50.00	66516.00	0.01	0.08	A	F	1992-02-12	1992-04-22	1992-03-10	TAKE BACK RETURN	REG AIR
15997988	332059	19578	1	49.00	53460.96	0.08	0.03	A	F	1994-05-31	1994-06-20	1994-06-22	NONE	FOB
15997988	904286	4287	2	37.00	47738.88	0.07	0.05	A	F	1994-05-24	1994-05-28	1994-06-03	COLLECT COD	RAIL
15997988	327898	15417	3	13.00	25036.44	0.09	0.08	A	F	1994-06-09	1994-06-28	1994-06-29	NONE	FOB
15997988	973125	48164	4	29.00	34744.32	0.10	0.02	A	F	1994-08-05	1994-06-26	1994-08-17	COLLECT COD	RAIL

only showing top 5 rows

7. Rename a Column (Rename the l_shipmode column to shipping_mode):

04:38 PM (<1s)

8

Python

#Rename a Column

#Rename the l_shipmode column to shipping_mode:

df_renamed = df_lineitem.withColumnRenamed("l_shipmode", "shipping_mode")

df_renamed.show(5)

(1) Spark Jobs

df_renamed: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenumber	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	shipping_mode	l_comment
15997987	295335	20346	4	50.00	66516.00	0.01	0.08	A	F	1992-02-12	1992-04-22	1992-03-10	TAKE BACK RETURN	REG AIR	even pinto beans...
15997988	332059	19578	1	49.00	53460.96	0.08	0.03	A	F	1994-05-31	1994-06-20	1994-06-22	NONE	FOB	detect carefully...
15997988	904286	4287	2	37.00	47738.88	0.07	0.05	A	F	1994-05-24	1994-05-28	1994-06-03	COLLECT COD	RAIL	the slyly e
15997988	327898	15417	3	13.00	25036.44	0.09	0.08	A	F	1994-06-09	1994-06-28	1994-06-29	NONE	FOB	ss courts. fina
15997988	973125	48164	4	29.00	34744.32	0.10	0.02	A	F	1994-08-05	1994-06-26	1994-08-17	COLLECT COD	RAIL	sly final packages

only showing top 5 rows

8. Filter Rows Using Multiple Conditions:

04:39 PM (1s)

9

Python

```
#Filter Rows Using Multiple Conditions

df_filtered_multi = df_lineitem.filter((df_lineitem.l_shipmode == "AIR") & (df_lineitem.l_quantity > 30))
df_filtered_multi.show(5)
```

(1) Spark Jobs

df_filtered_multi: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenum	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode	l_comment
15997989 02-06	150305	306	2	49.00	66409.70	0.01	0.08	R	F	1993-01-18	1993-02-05	1993-02-06	NONE	AIR	ometimes iron
15998018 10-28	650130	12644	1	50.00	54005.00	0.06	0.07	N	O	1997-10-20	1997-10-09	1997-10-28	COLLECT COD	AIR	the slyly special...
15998018 12-18	455260	30279	4	41.00	49824.84	0.03	0.07	N	O	1997-11-21	1997-09-23	1997-12-18	TAKE BACK RETURN	AIR	d accounts aga
15998022 12-31	137122	24629	1	42.00	48683.04	0.01	0.06	A	F	1992-12-26	1992-10-21	1992-12-31	DELIVER IN PERSON	AIR	instructions boos...
15998022 12-18	600590	591	2	41.00	61112.96	0.08	0.02	A	F	1992-12-06	1992-11-16	1992-12-18	DELIVER IN PERSON	AIR	osits detect even...

only showing top 5 rows

9. Calculate Discounted Price:

04:40 PM (1s)

10

Python

```
#Calculate Discounted Price
df_discounted_price = df_lineitem.withColumn(
    "discounted_price",
    df_lineitem.l_extendedprice * (1 - df_lineitem.l_discount)
)
df_discounted_price.show(5)
```

(1) Spark Jobs

df_discounted_price: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 15 more fields]

l_orderkey	l_partkey	l_suppkey	l_linenum	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode	l_comment	discounted_price
15997987 03-10	295335	20346	4	50.00	66516.00	0.01	0.08	A	F	1992-02-12	1992-04-22	1992-03-10	TAKE BACK RETURN	REG AIR	even pinto beans...	65850.8400
15997988 06-22	332059	19578	1	49.00	53460.96	0.08	0.03	A	F	1994-05-31	1994-06-20	1994-06-22	NONE	FOB	detect carefully....	49184.0832
15997988 06-03	904286	4287	2	37.00	47738.88	0.07	0.05	A	F	1994-05-24	1994-05-28	1994-06-03	COLLECT COD	RAIL	the slyly e	44397.1584
15997988 06-29	327898	15417	3	13.00	25036.44	0.09	0.08	A	F	1994-06-09	1994-06-28	1994-06-29	NONE	FOB	ss courts. fina	22783.1604
15997988 08-17	973125	48164	4	29.00	34744.32	0.10	0.02	A	F	1994-08-05	1994-06-26	1994-08-17	COLLECT COD	RAIL	sly final packages	31269.8880

10. Filter Based on Date Range:

04:40 PM (1s)

11

Python

```
#Filter Based on Date Range
from pyspark.sql.functions import col

df_date_filtered = df_lineitem.filter(
    (col("l_shipdate") >= "1995-01-01") & (col("l_shipdate") <= "1995-12-31")
)
df_date_filtered.show(5)
```

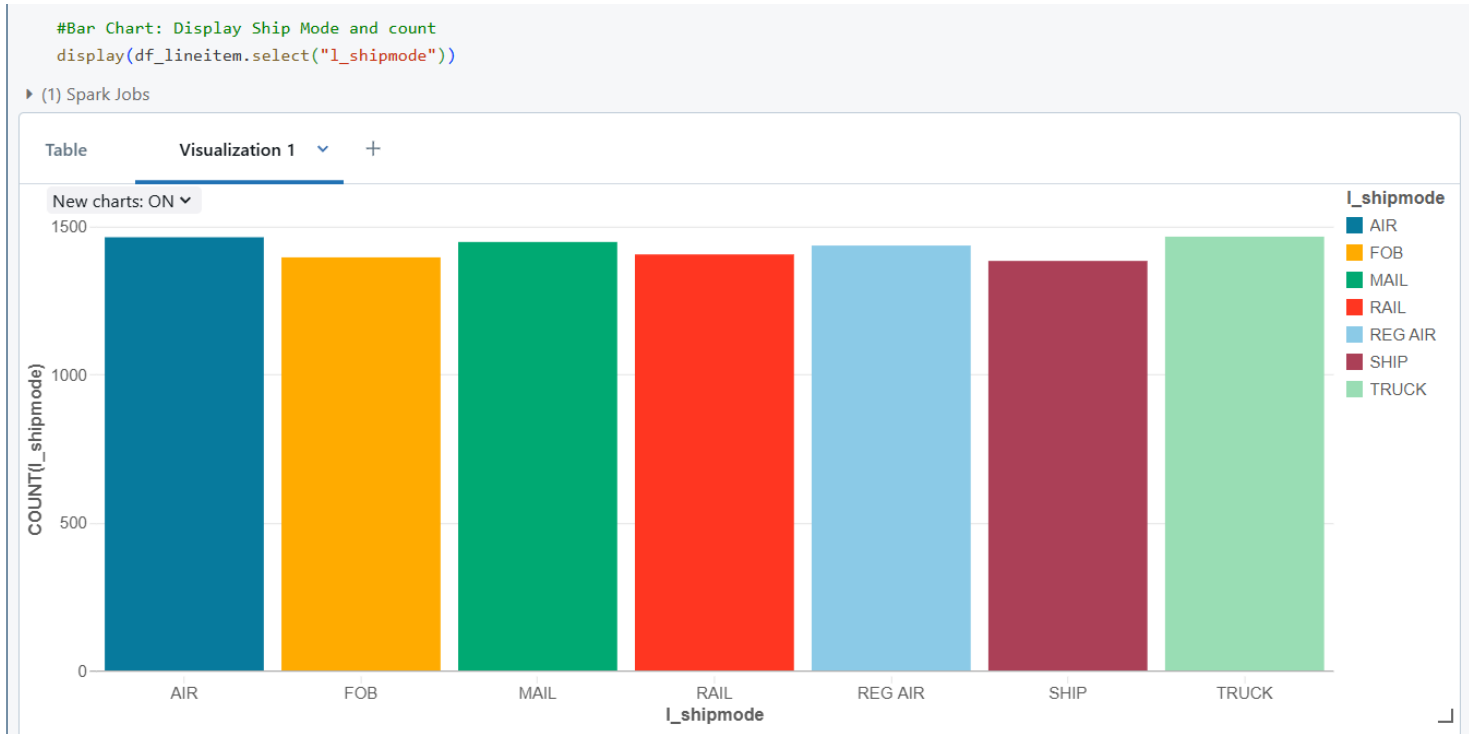
(1) Spark Jobs

df_date_filtered: pyspark.sql.dataframe.DataFrame = [l_orderkey: long, l_partkey: long ... 14 more fields]

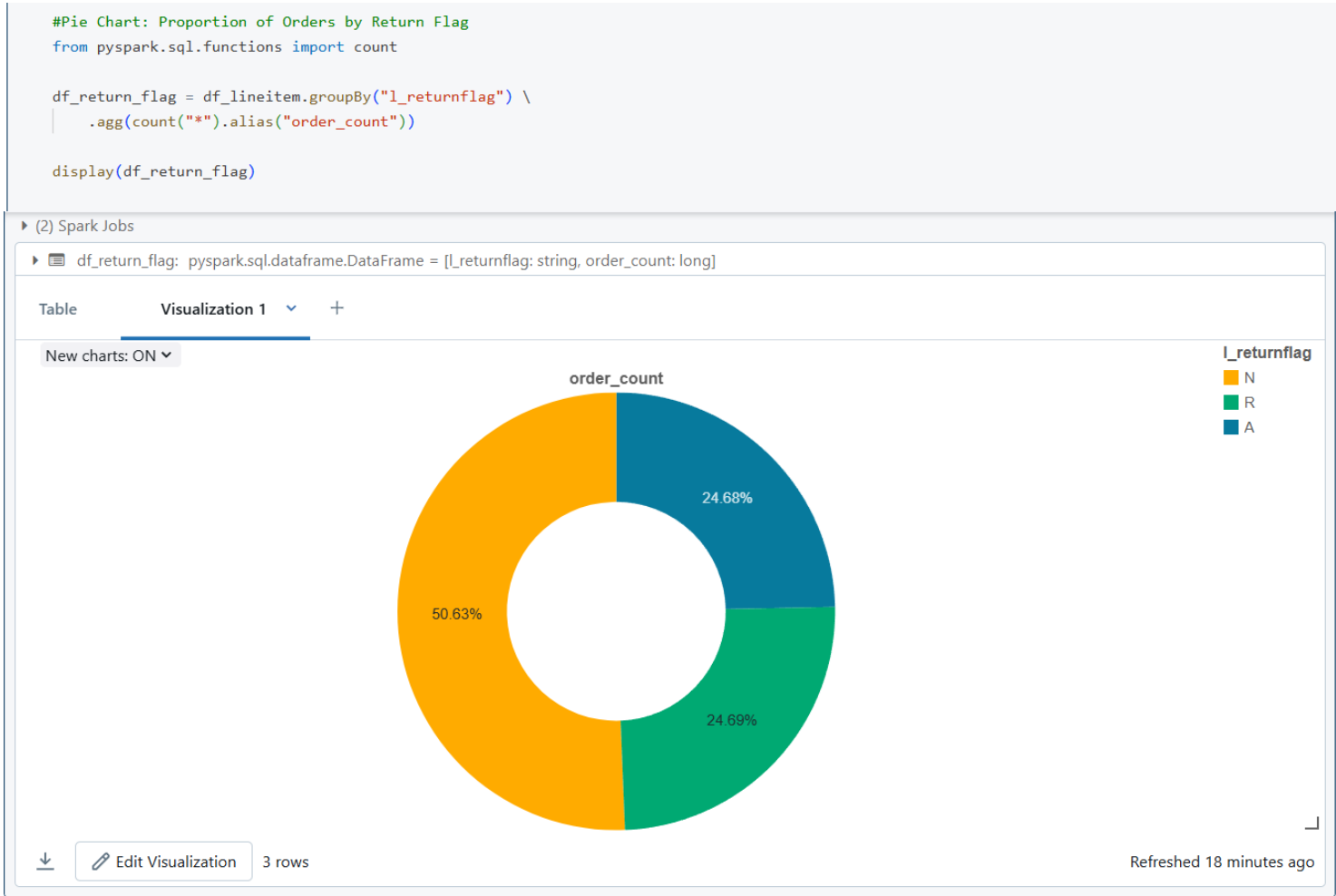
l_orderkey	l_partkey	l_supplykey	l_linenum	l_quantity	l_extendedprice	l_discount	l_tax	l_returnflag	l_linestatus	l_shipdate	l_commitdate	l_receiptdate	l_shipinstruct	l_shipmode	l_comment
15997991	466072	3600	1	26.00	26989.30	0.04	0.00	N	F	1995-05-31	1995-07-11	1995-06-30	COLLECT COD	TRUCK	es. regula
15997991	271922	34428	2	26.00	49241.66	0.01	0.05	N	O	1995-06-19	1995-06-23	1995-07-06	COLLECT COD	AIR	ages are fluffily...
15997991	183864	8871	3	6.00	11687.16	0.08	0.06	N	O	1995-07-06	1995-07-12	1995-08-03	COLLECT COD	MAIL	cajole furiously. s
15997991	199077	11581	4	17.00	19993.19	0.04	0.00	N	O	1995-07-12	1995-07-17	1995-07-27	COLLECT COD	TRUCK	ve silent request...
15997991	873553	23554	5	43.00	65639.93	0.10	0.01	N	O	1995-06-21	1995-07-12	1995-06-25	COLLECT COD	MAIL	e pending, express

Step 6: Visualization:

1. Bar Chart: Display Ship Mode and count:



2. Pie Chart: Proportion of Orders by Return Flag:



3. Line Chart: Average Quantity Over Time:

```
#Line Chart: Average Quantity Over Time
from pyspark.sql.functions import avg

df_avg_quantity = df_lineitem.groupBy("l_shipdate") \
    .agg(avg("l_quantity").alias("avg_quantity")) \
    .orderBy("l_shipdate")

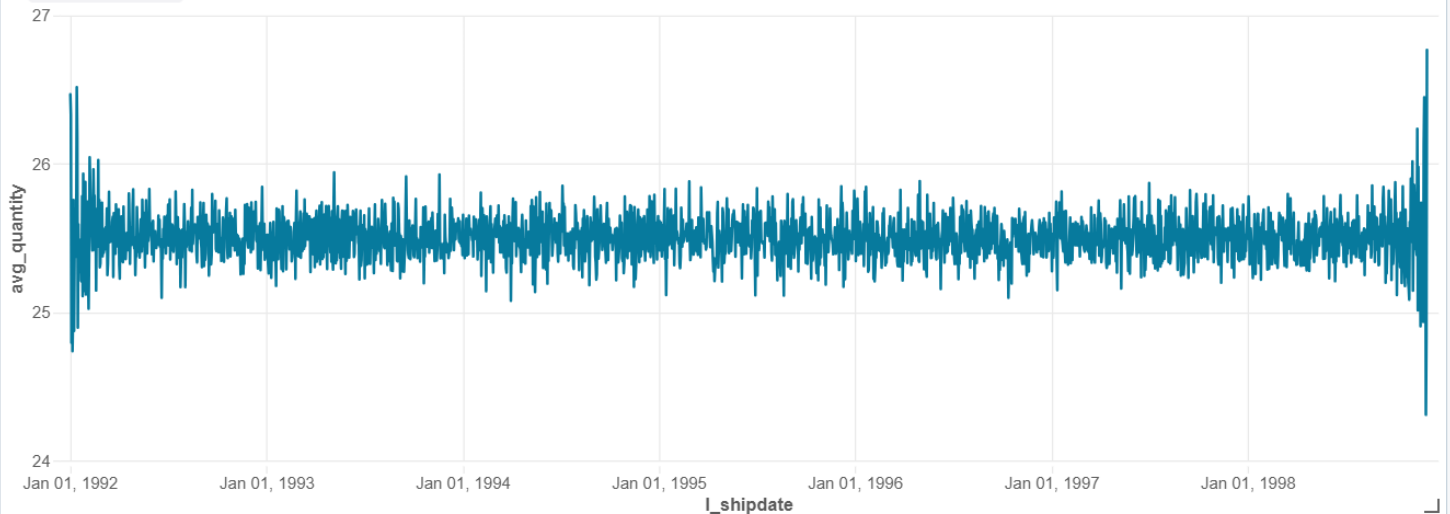
display(df_avg_quantity)
```

▶ (2) Spark Jobs

▶ df_avg_quantity: pyspark.sql.dataframe.DataFrame = [l_shipdate: date, avg_quantity: decimal(22,6)]

Table Visualization 1 ▾ +

New charts: ON ▾



↓ Edit Visualization 2,526 rows

Refreshed 19 minutes ago

4. Histogram: Distribution of Quantity:

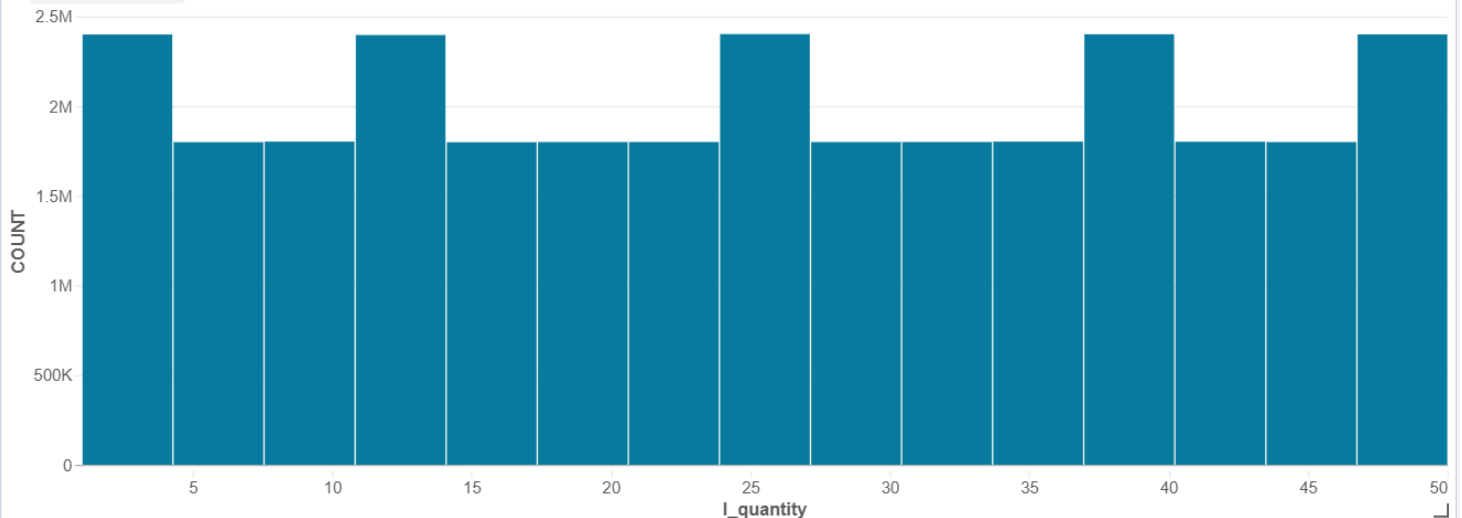
```
#Histogram: Distribution of Quantity
display(df_lineitem.select("l_quantity"))
```

▶ (1) Spark Jobs

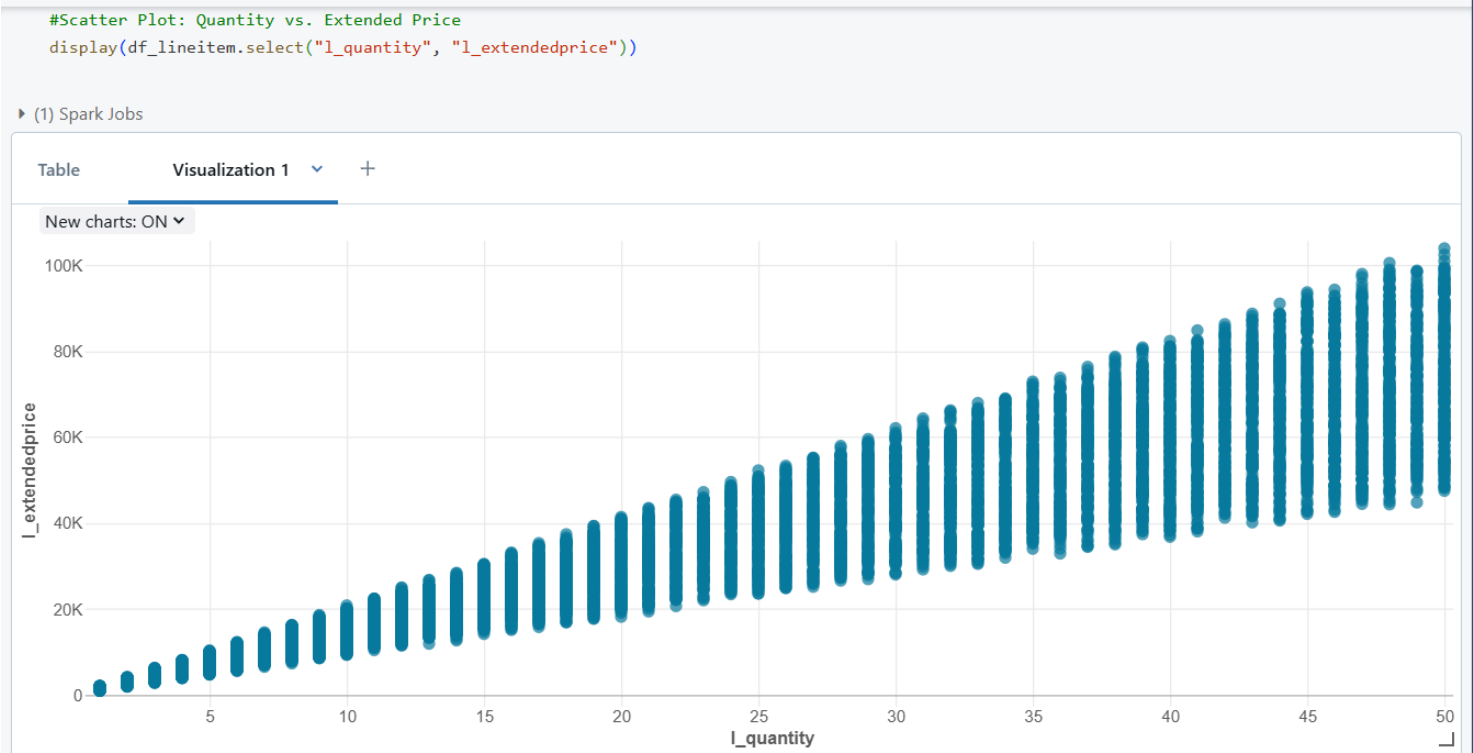
Table Visualization 1 ▾ +

▶ (4) Spark Jobs

New charts: ON ▾



5. Scatter Plot: Quantity vs. Extended Price:



Submitted By:
Aathirainathan P