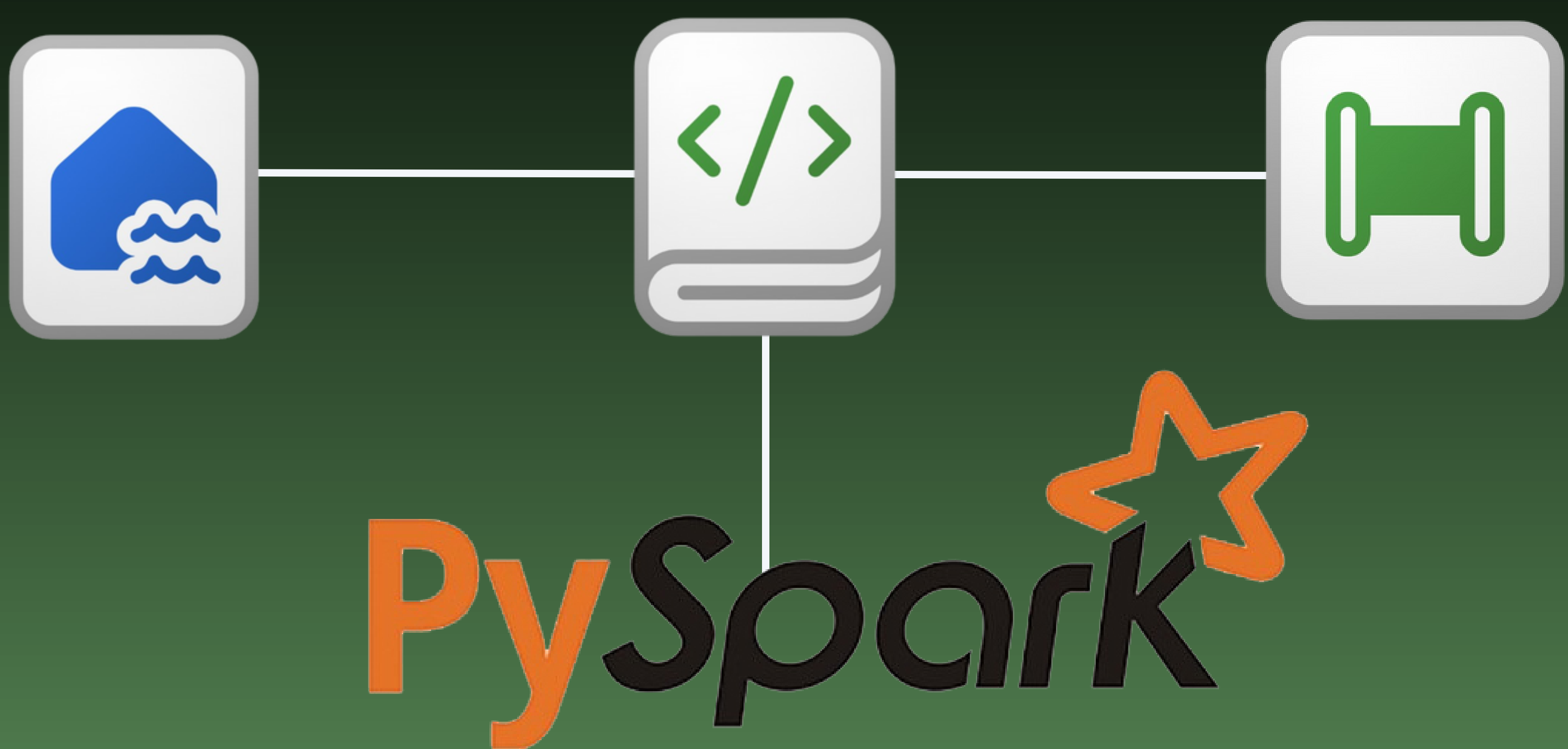
Retail Banking **Data Engineering** Project using **Microsoft Fabric!**

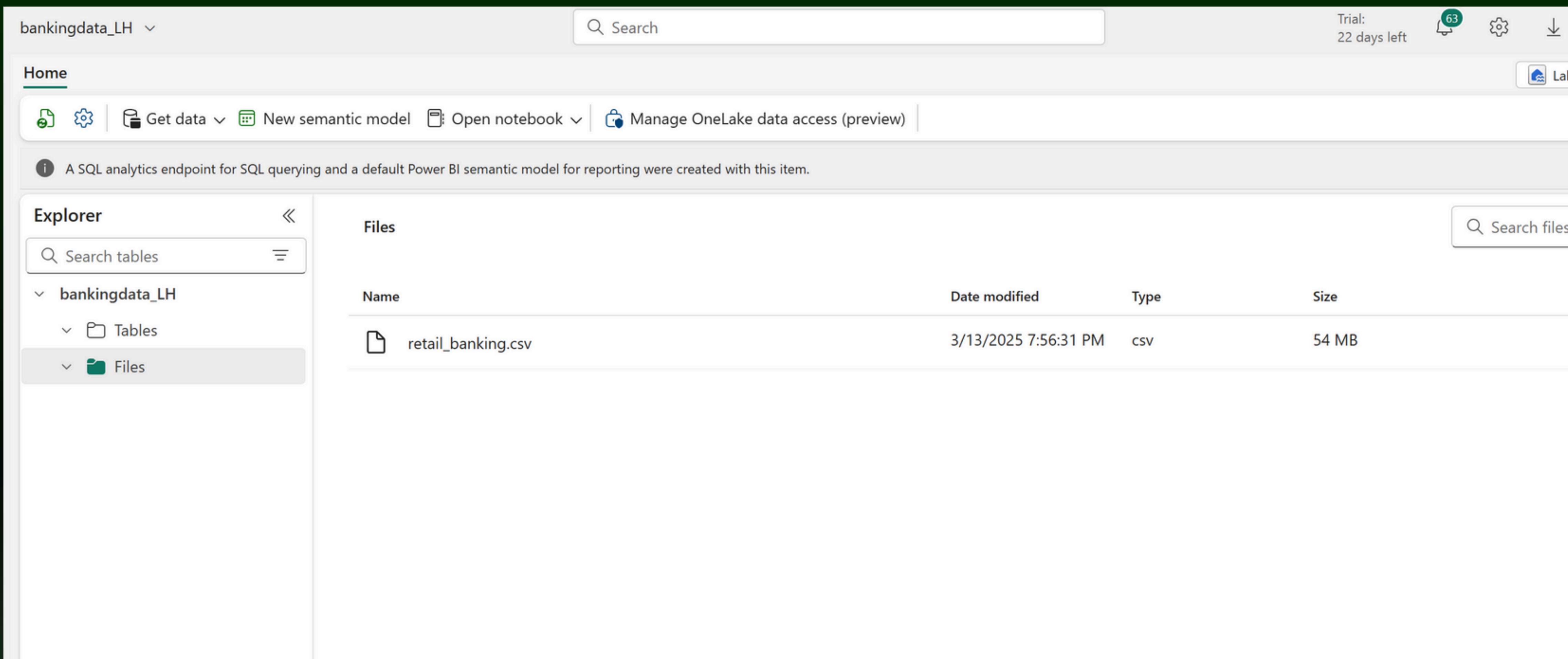


Inturi Suparna Babu

Retail Banking Project

PySpark

CREATED LAKEHOUSE AND UPLOADED THE INVESTMENT BANKING DATA INTO LAKEHOUSE

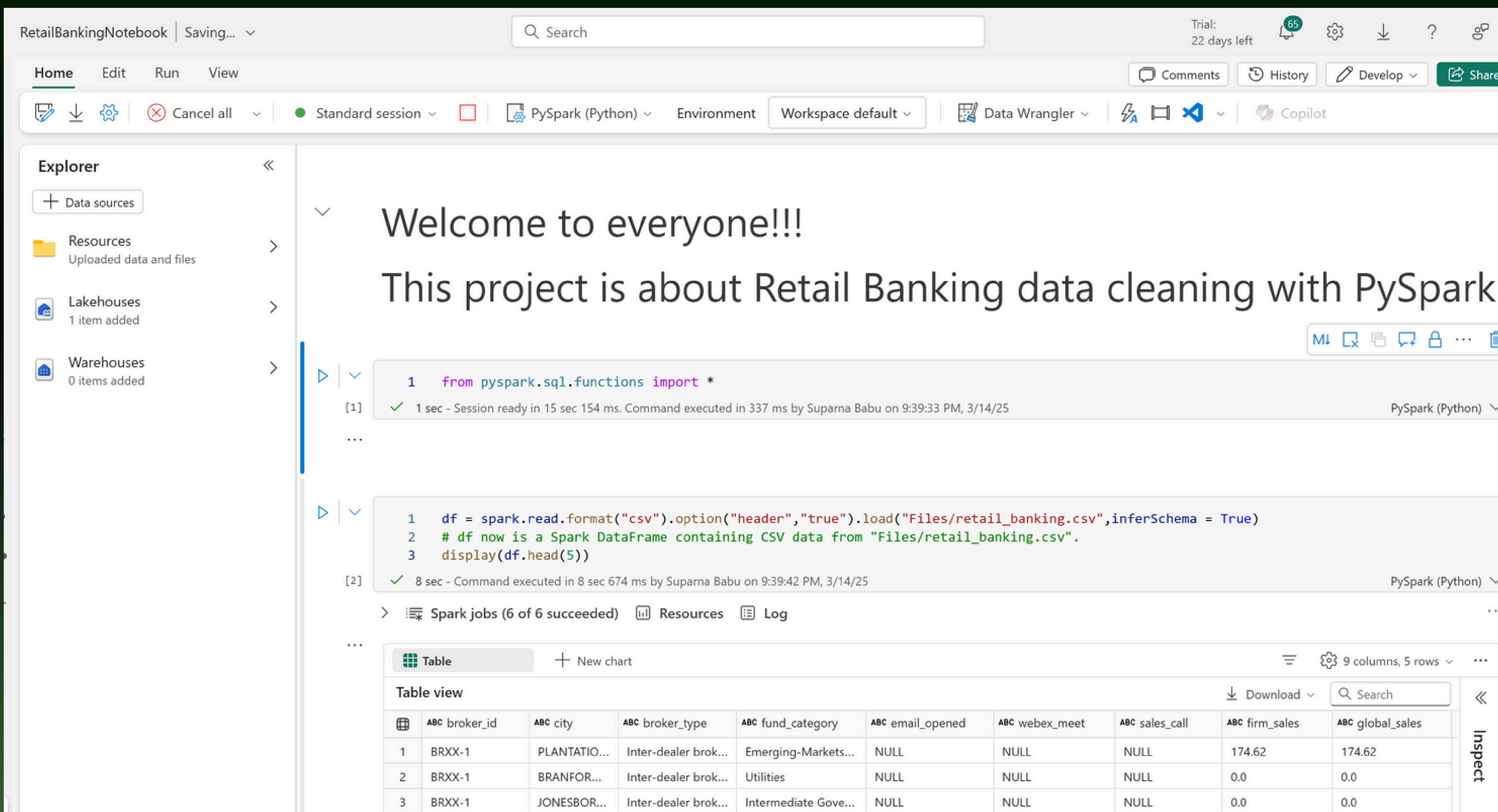


Inturi Suparna Babu

Retail Banking Project

PySpark

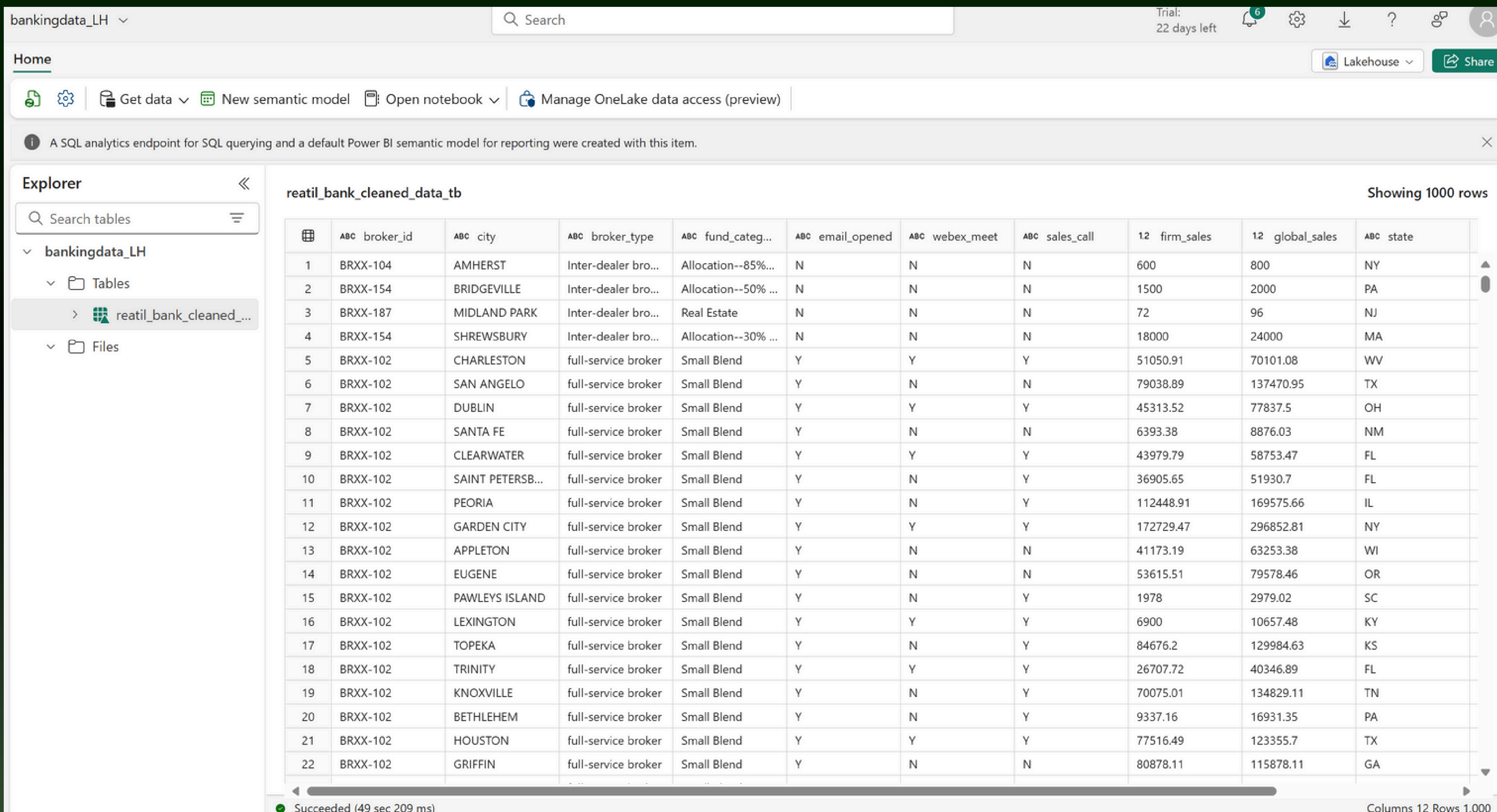
IN NOTEBOOK PERFORMED ALL DATA CLEANING AND PROCESSING ACTIVITY AND PUSHED TO LAKEHOUSE



Inturi Suparna Babu

Retail Banking Project

NOW CLEANED DATA LOADED SUCESSFULLY INTO LAKEHOUSE AS DELTA TABLES


bankingdata_LH

Search

Trial: 22 days left

Home

Lakehouse

Share

Get data | New semantic model | Open notebook | Manage OneLake data access (preview)

A SQL analytics endpoint for SQL querying and a default Power BI semantic model for reporting were created with this item.

Explorer

Search tables

- bankingdata_LH
  - Tables
    - reatil_bank_cleaned_...
  - Files

reatil_bank_cleaned_data_tb

Showing 1000 rows

	broker_id	city	broker_type	fund_categ...	email_opened	webex_meet	sales_call	firm_sales	global_sales	state
1	BRXX-104	AMHERST	Inter-dealer bro...	Allocation--85%...	N	N	N	600	800	NY
2	BRXX-154	BRIDGEVILLE	Inter-dealer bro...	Allocation--50% ...	N	N	N	1500	2000	PA
3	BRXX-187	MIDLAND PARK	Inter-dealer bro...	Real Estate	N	N	N	72	96	NJ
4	BRXX-154	SHREWSBURY	Inter-dealer bro...	Allocation--30% ...	N	N	N	18000	24000	MA
5	BRXX-102	CHARLESTON	full-service broker	Small Blend	Y	Y	Y	51050.91	70101.08	WV
6	BRXX-102	SAN ANGELO	full-service broker	Small Blend	Y	N	N	79038.89	137470.95	TX
7	BRXX-102	DUBLIN	full-service broker	Small Blend	Y	Y	Y	45313.52	77837.5	OH
8	BRXX-102	SANTA FE	full-service broker	Small Blend	Y	N	N	6393.38	8876.03	NM
9	BRXX-102	CLEARWATER	full-service broker	Small Blend	Y	Y	Y	43979.79	58753.47	FL
10	BRXX-102	SAINT PETERSB...	full-service broker	Small Blend	Y	N	Y	36905.65	51930.7	FL
11	BRXX-102	PEORIA	full-service broker	Small Blend	Y	N	Y	112448.91	169575.66	IL
12	BRXX-102	GARDEN CITY	full-service broker	Small Blend	Y	Y	Y	172729.47	296852.81	NY
13	BRXX-102	APPLETON	full-service broker	Small Blend	Y	N	N	41173.19	63253.38	WI
14	BRXX-102	EUGENE	full-service broker	Small Blend	Y	N	N	53615.51	79578.46	OR
15	BRXX-102	PAWLEYS ISLAND	full-service broker	Small Blend	Y	N	Y	1978	2979.02	SC
16	BRXX-102	LEXINGTON	full-service broker	Small Blend	Y	Y	Y	6900	10657.48	KY
17	BRXX-102	TOPEKA	full-service broker	Small Blend	Y	N	Y	84676.2	129984.63	KS
18	BRXX-102	TRINITY	full-service broker	Small Blend	Y	Y	Y	26707.72	40346.89	FL
19	BRXX-102	KNOXVILLE	full-service broker	Small Blend	Y	N	Y	70075.01	134829.11	TN
20	BRXX-102	BETHLEHEM	full-service broker	Small Blend	Y	N	Y	9337.16	16931.35	PA
21	BRXX-102	HOUSTON	full-service broker	Small Blend	Y	Y	Y	77516.49	123355.7	TX
22	BRXX-102	GRIFFIN	full-service broker	Small Blend	Y	N	N	80878.11	115878.11	GA

Succeeded (49 sec 209 ms)

Columns 12 Rows 1,000

Inturi Suparna Babu

Retail Banking Project

Key Numbers

- Rawdatset : Total columns - 12, Total Rows - 7,53,089
- Created 1 Lakehouse to perform activities
- Created *1 notebook* to clean the data.

Thank you!

Inturi Suparna Babu

retailbankingnotebook

March 14, 2025

1 Welcome to everyone!!!

2 This project is about Retail Banking data cleaning with PySpark

[1]:
```
from pyspark.sql.functions import *
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 3, Finished, Available,␣
↪Finished)
```

[2]:
```
df = spark.read.format("csv").option("header","true").load("Files/
↪retail_banking.csv",inferSchema = True)
# df now is a Spark DataFrame containing CSV data from "Files/retail_banking.
↪csv".
display(df.head(5))
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 4, Finished, Available,␣
↪Finished)
```

```
SynapseWidget(Synapse.DataFrame, 95a0a3e0-eb34-44cd-87ff-29785dcfb8f8)
```

[3]:
```
#df.select("*").where(col('city') == 'KNOXVILLE, TN, TN').show()
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 5, Finished, Available,␣
↪Finished)
```

[4]:
```
columns_count = len(df.columns)
rows_count = df.count()
print("This dataset has",columns_count, "columns")
print("This dataset has",rows_count, "rows")      #to find the columns and rows␣
↪count
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 6, Finished, Available,␣
↪Finished)
```

```
This dataset has 9 columns
This dataset has 753089 rows
```

```
[5]: df.printSchema() #to view the schema of dataframe
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 7, Finished, Available,
↪Finished)

root
 |-- broker_id: string (nullable = true)
 |-- city: string (nullable = true)
 |-- broker_type: string (nullable = true)
 |-- fund_category: string (nullable = true)
 |-- email_opened: string (nullable = true)
 |-- webex_meet: string (nullable = true)
 |-- sales_call: string (nullable = true)
 |-- firm_sales: double (nullable = true)
 |-- global_sales: double (nullable = true)
```

```
[6]: list(df.columns) #to view list of columns
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 8, Finished, Available,
↪Finished)
```

```
[6]: ['broker_id',
 'city',
 'broker_type',
 'fund_category',
 'email_opened',
 'webex_meet',
 'sales_call',
 'firm_sales',
 'global_sales']
```

```
[7]: brokerid_nulls_count = df.filter(col('broker_id').isNull()).count() #to find
↪the nulls count in respectieve column
print("broker_id column has ",brokerid_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 9, Finished, Available,
↪Finished)

broker_id column has  0 nulls
```

```
[8]: city_nulls_count = df.filter(col('city').isNull()).count() #to find the nulls
↪count in respectieve column
print("city column has",city_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 10, Finished, Available,
↪Finished)

city column has 0 nulls
```

[9]:
```python
brokertype_nulls_count = df.filter(col('broker_type').isNull()).count() #to␣
 ↪find the nulls count in respectieve column
print("broker_type column has",brokertype_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 11, Finished, Available,␣
 ↪Finished)
```

```
broker_type column has 0 nulls
```

[10]:
```python
fundcategory_nulls_count = df.filter(col('fund_category').isNull()).count() ␣
 ↪#to find the nulls count in respectieve column
print("fund_category column has",fundcategory_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 12, Finished, Available,␣
 ↪Finished)
```

```
fund_category column has 0 nulls
```

[11]:
```python
email_opened_nulls_count = df.filter(col('email_opened').isNull()).count() #to␣
 ↪find the nulls count in respectieve column
print("email_opened column has",email_opened_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 13, Finished, Available,␣
 ↪Finished)
```

```
email_opened column has 434545 nulls
```

[12]:
```python
webex_meet_nulls_count = df.filter(col('webex_meet').isNull()).count() #to find␣
 ↪the nulls count in respectieve column
print("webex_meet column has",webex_meet_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 14, Finished, Available,␣
 ↪Finished)
```

```
webex_meet column has 667429 nulls
```

[13]:
```python
sales_call_nulls_count = df.filter(col('sales_call').isNull()).count() #to find␣
 ↪the nulls count in respectieve column
print("sales_call column has",sales_call_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 15, Finished, Available,␣
 ↪Finished)
```

```
sales_call column has 573091 nulls
```

[14]:
```python
firm_sales_nulls_count = df.filter(col('firm_sales').isNull()).count() #to find␣
 ↪the nulls count in respectieve column
print("firm_sales column has",firm_sales_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 16, Finished, Available,
↪Finished)
```

```
firm_sales column has 0 nulls
```

[15]:
```python
global_sales_nulls_count = df.filter(col('global_sales').isNull()).count() #to
↪find the nulls count in respectieve column
print("global_sales column has",global_sales_nulls_count,"nulls")
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 17, Finished, Available,
↪Finished)
```

```
global_sales column has 0 nulls
```

3 Now I can calculate all the columns missing values at single instance with Functions

[16]:
```python
def check_miss_values_count (data, lst_cl):
    missing_values = {}
    for i in lst_cl:
        a = data.filter(col(i).isNull()).count()
        missing_values[i] = a
    return (missing_values)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 18, Finished, Available,
↪Finished)
```

[17]:
```python
check_miss_values_count(df,df.columns)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 19, Finished, Available,
↪Finished)
```

[17]:
```
{'broker_id': 0,
 'city': 0,
 'broker_type': 0,
 'fund_category': 0,
 'email_opened': 434545,
 'webex_meet': 667429,
 'sales_call': 573091,
 'firm_sales': 0,
 'global_sales': 0}
```

4 To get the missing value percentage with Functions

[18]:
```
def check_miss_values_pct(data,lst_cl):
    miss_value_pct = {}
    for i in lst_cl:
        a = data.filter(col(i).isNull()).count()
        b = data.count()
        c = (a/b) * 100
        miss_value_pct[i] = c
    return (miss_value_pct)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 20, Finished, Available,␣
↪Finished)
```

[19]:
```
check_miss_values_pct(df,df.columns)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 21, Finished, Available,␣
↪Finished)
```

[19]:
```
{'broker_id': 0.0,
 'city': 0.0,
 'broker_type': 0.0,
 'fund_category': 0.0,
 'email_opened': 57.70167934998387,
 'webex_meet': 88.62551438143433,
 'sales_call': 76.09870812081971,
 'firm_sales': 0.0,
 'global_sales': 0.0}
```

[20]:
```
df.select('broker_id').distinct().count() #to get disticnt count of broker_id
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 22, Finished, Available,␣
↪Finished)
```

[20]:
```
1178
```

5 To check distinct values of all columns by using function

[21]:
```
def check_dist_values(data,lst_cl):
    dist_values_count = {}
    for i in lst_cl:
        a = data.select(i).distinct().count()
        dist_values_count[i] = a
    return (dist_values_count)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 23, Finished, Available,␣
↪Finished)
```

[22]:
```
check_dist_values(df,df.columns)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 24, Finished, Available,
↪Finished)
```

[22]:
```
{'broker_id': 1178,
 'city': 9813,
 'broker_type': 2,
 'fund_category': 103,
 'email_opened': 2,
 'webex_meet': 2,
 'sales_call': 2,
 'firm_sales': 44958,
 'global_sales': 360477}
```

[23]:
```
df.select('broker_type').distinct().show() #to get distinct values of
↪broker_type column
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 25, Finished, Available,
↪Finished)
```

```
+--------------------+
|         broker_type|
+--------------------+
|full-service broker|
|Inter-dealer broker|
+--------------------+
```

[24]:
```
df.select('email_opened').distinct().show() #to get distinct values of
↪email_opened column
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 26, Finished, Available,
↪Finished)
```

```
+------------+
|email_opened|
+------------+
|           Y|
|        NULL|
+------------+
```

[25]:
```
df.select('webex_meet').distinct().show() #to get distinct values of
↪webex_meet column
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 27, Finished, Available,
↪Finished)
```

```
+-----------+
|webex_meet|
+-----------+
|          Y|
|       NULL|
+-----------+
```

[26]:
```
df.select('sales_call').distinct().show() #to get distinct values of␣
↪sales_call column
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 28, Finished, Available,␣
↪Finished)
```

```
+-----------+
|sales_call|
+-----------+
|          Y|
|       NULL|
+-----------+
```

6 Now I will fill all the nulls in email_opened, webex_meet and sales_call columns

[27]:
```
df = df.fillna('N', subset=['email_opened','webex_meet','sales_call']) #filling␣
↪nulls with N
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 29, Finished, Available,␣
↪Finished)
```

[28]:
```
df.select(['email_opened','webex_meet','sales_call']).distinct().show()
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 30, Finished, Available,␣
↪Finished)
```

```
+------------+----------+----------+
|email_opened|webex_meet|sales_call|
+------------+----------+----------+
|           N|         N|         N|
|           Y|         N|         N|
|           Y|         Y|         Y|
|           N|         N|         Y|
|           Y|         Y|         N|
|           N|         Y|         Y|
|           Y|         N|         Y|
|           N|         Y|         N|
```

```
+------------+---------+---------+
```

7 Now again I'm checking the nulls percentage of all columns

[29]:
```python
def check_miss_pctg(data,lst_cl):
    miss_pcntg = {}
    for i in lst_cl:
        a = data.filter(col(i).isNull()).count()
        b = data.count()
        c = (a/b) * 100
        miss_pcntg [i] = c
    return (miss_pcntg)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 31, Finished, Available,␣
↪Finished)
```

[30]:
```python
check_miss_pctg(df,df.columns)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 32, Finished, Available,␣
↪Finished)
```

[30]:
```
{'broker_id': 0.0,
 'city': 0.0,
 'broker_type': 0.0,
 'fund_category': 0.0,
 'email_opened': 0.0,
 'webex_meet': 0.0,
 'sales_call': 0.0,
 'firm_sales': 0.0,
 'global_sales': 0.0}
```

8 Now I will check broker_type, fund_category values unique count

[31]:
```python
df.groupBy('broker_type').agg(countDistinct('broker_id').alias("Unique Count")).
↪show()
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 33, Finished, Available,␣
↪Finished)
```

```
+------------------+------------+
|       broker_type|Unique Count|
+------------------+------------+
|full-service broker|           5|
|Inter-dealer broker|        1173|
```

```
+-------------------+------------+
```

[32]:
```python
df.groupBy('fund_category').agg(countDistinct('fund_category').alias("Unique␣
↪count")).show()
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 34, Finished, Available,␣
↪Finished)
```

[33]:
```python
df.select('city').show(5)
```

```
StatementMeta(, dae29050-9ec3-453f-880e-1428b9fdb12c, 35, Finished, Available,␣
↪Finished)
```

[ ]:
```python
df.groupBy('city').agg(countDistinct('city')).show(200,False)
```

```
StatementMeta(, , -1, Waiting, , Waiting)
```

[ ]:
```python
df.select('city').filter(col('city').contains(' TN, TN')).show()
```

9 Now I will extract state from city column

[36]:
```python
df = df.withColumn("state", split(df["city"], ", ")[1]) #extracting state
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 38, Finished, Available,␣
↪Finished)
```

[37]:
```python
df = df.withColumn('city', split(col('city'), ",")[0]) #extracting city
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 39, Finished, Available,␣
↪Finished)
```

[38]:
```python
df.show(5)
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 40, Finished, Available,␣
↪Finished)

+---------+-------------+-------------------+--------------------+------------+-
---------+----------+----------+------------+-----+
|broker_id|         city|        broker_type|
fund_category|email_opened|webex_meet|sales_call|firm_sales|global_sales|state|
+---------+-------------+-------------------+--------------------+------------+-
---------+----------+----------+------------+-----+
|   BRXX-1|   PLANTATION|Inter-dealer broker|Emerging-Markets …|           N|
N|         N|    174.62|      174.62|   FL|
|   BRXX-1|     BRANFORD|Inter-dealer broker|           Utilities|           N|
N|         N|       0.0|         0.0|   CT|
|   BRXX-1|    JONESBORO|Inter-dealer broker|Intermediate Gove…|           N|
```

```
N|          N|       0.0|          0.0|   GA|
|   BRXX-2|       VIENNA|Inter-dealer broker|Intermediate Gove…|           Y|
N|          N|       0.0|      30709.0|   VA|
|   BRXX-3|CHAGRIN FALLS|full-service broker|    Target-Date 2050|           Y|
N|          Y|       0.0|          0.0|   OH|
+---------+-------------+-------------------+--------------------+------------+-
---------+----------+----------+-----------+-----+
only showing top 5 rows
```

10 to check if there are any numerical values in city column

[39]:
```
df.select('city').filter(col('city').rlike("^[0-9]+$")).distinct().show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 41, Finished, Available,
Finished)
```

```
+------+
| city|
+------+
|15801|
|64150|
|95678|
+------+
```

11 I assume this as zip code of that city Now I'll replace this zip code with city names

12 64150 –> Riverside

13 15801 –> Du Bois

14 95678 –> Roseville

[40]:
```
df = df.withColumn("city", regexp_replace('city','64150','Riverside'))
#Replacing the zip code with city name
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 42, Finished, Available,
Finished)
```

[43]:
```
df = df.withColumn('city', regexp_replace('city','15801','Du Bois')) #Replacing
the zip code with city name
df = df.withColumn('city', regexp_replace('city','95678','Roseville'))
#Replacing the zip code with city name
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 43, Finished, Available,␣
↪Finished)
```

[44]:
```
df.select('city').filter(col('city').rlike("^[0-9]+$")).distinct().show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 44, Finished, Available,␣
↪Finished)
```

```
+----+
|city|
+----+
+----+
```

[46]:
```
df.select('state').filter(length('state') > 2).show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 45, Finished, Available,␣
↪Finished)
```

```
+-----+
|state|
+-----+
+-----+
```

[47]:
```
df.filter(col('firm_sales').isNull()).count()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 46, Finished, Available,␣
↪Finished)
```

[47]: 0

[49]:
```
df.filter(col('global_sales').isNull()).count()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 47, Finished, Available,␣
↪Finished)
```

[49]: 0

[50]:
```
df.printSchema()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 48, Finished, Available,␣
↪Finished)
```

```
root
 |-- broker_id: string (nullable = true)
 |-- city: string (nullable = true)
 |-- broker_type: string (nullable = true)
 |-- fund_category: string (nullable = true)
 |-- email_opened: string (nullable = false)
```

```
 |-- webex_meet: string (nullable = false)
 |-- sales_call: string (nullable = false)
 |-- firm_sales: double (nullable = true)
 |-- global_sales: double (nullable = true)
 |-- state: string (nullable = true)
```

15 Getting the broker_id wise firm_sales

[54]:
```
df.groupBy('broker_id').agg(sum('firm_sales').alias('Total_firm_sales')).
 ↪orderBy(desc('Total_firm_sales')).show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 49, Finished, Available,␣
 ↪Finished)

+---------+--------------------+
|broker_id|    Total_firm_sales|
+---------+--------------------+
| BRXX-298| 5.748833387300012E8|
|  BRXX-53|2.1652823814999998E8|
| BRXX-154|       2.1559129687E8|
| BRXX-262|1.9989880654000002E8|
|   BRXX-3|1.4541805568999997E8|
| BRXX-102|       1.3920014522E8|
|  BRXX-92|       1.2718398564E8|
| BRXX-253|       1.0950492304E8|
| BRXX-124| 8.806711471000002E7|
|  BRXX-93| 7.784665832999998E7|
|  BRXX-94| 6.184790281999999E7|
|  BRXX-70|3.2582956979999997E7|
| BRXX-299| 2.816686303000001E7|
| BRXX-263|2.7194103220000003E7|
| BRXX-291|2.1790086169999998E7|
| BRXX-136|        2.035211896E7|
| BRXX-247|2.0185874740000002E7|
| BRXX-171|2.0175036339999996E7|
| BRXX-261|2.0074834919999994E7|
| BRXX-106|1.9562382400000002E7|
+---------+--------------------+
only showing top 20 rows
```

16 Getting the broker_id wise global_sales

[55]:
```
df.groupBy('broker_id').agg(sum('global_sales').alias('ttl_global_sales')).
↪orderBy(desc('ttl_global_sales')).show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 50, Finished, Available,␣
↪Finished)

+---------+--------------------+
|broker_id|    ttl_global_sales|
+---------+--------------------+
|  BRXX-53|  8.36834248278002E9|
| BRXX-298| 8.225108341179993E9|
| BRXX-154| 5.907926378420001E9|
|   BRXX-3| 5.880296586749996E9|
| BRXX-262| 4.671995777180005E9|
| BRXX-102|3.8967221447000017E9|
|  BRXX-92| 3.858099052869999E9|
| BRXX-253|     2.41769305174E9|
| BRXX-124|1.9496785312600005E9|
|  BRXX-55|     1.40345230576E9|
|  BRXX-93|1.2780107563400004E9|
|  BRXX-94|     1.21571801222E9|
| BRXX-172| 8.027236808899999E8|
| BRXX-247| 6.464248926900003E8|
|  BRXX-70| 6.334079940899998E8|
| BRXX-291| 6.156686732499999E8|
| BRXX-299| 5.385067264999998E8|
|  BRXX-76|5.0848096750000006E8|
| BRXX-179|4.7917767275999993E8|
| BRXX-301|4.5536769792999977E8|
+---------+--------------------+
only showing top 20 rows
```

17 Getting the broker_id wise firm_sales and global_sales

[58]:
```
df.groupBy('broker_id').agg(sum('firm_sales').
↪alias("ttl_firm_sales"),sum('global_sales').alias('ttl_global_sales'))\
 .orderBy(desc('ttl_firm_sales'),desc('ttl_global_sales')).show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 51, Finished, Available,␣
↪Finished)

+---------+--------------------+--------------------+
|broker_id|      ttl_firm_sales|    ttl_global_sales|
+---------+--------------------+--------------------+
| BRXX-298| 5.748833387300012E8| 8.225108341179993E9|
```

```
|  BRXX-53|2.1652823814999998E8|  8.36834248278002E9|
| BRXX-154|       2.1559129687E8| 5.907926378420001E9|
| BRXX-262|1.9989880654000002E8| 4.671995777180005E9|
|   BRXX-3|1.4541805568999997E8| 5.880296586749996E9|
| BRXX-102|       1.3920014522E8|3.8967221447000017E9|
|  BRXX-92|       1.2718398564E8| 3.858099052869999E9|
| BRXX-253|       1.0950492304E8|     2.41769305174E9|
| BRXX-124| 8.806711471000002E7|1.9496785312600005E9|
|  BRXX-93| 7.784665832999998E7|1.2780107563400004E9|
|  BRXX-94| 6.184790281999999E7|     1.21571801222E9|
|  BRXX-70|3.2582956979999997E7| 6.334079940899998E8|
| BRXX-299| 2.816686303000001E7| 5.385067264999998E8|
| BRXX-263|2.7194103220000003E7| 2.254376408700001E8|
| BRXX-291|2.1790086169999998E7| 6.156686732499999E8|
| BRXX-136|        2.035211896E7|       2.7739742015E8|
| BRXX-247|2.0185874740000002E7| 6.464248926900003E8|
| BRXX-171|2.0175036339999996E7|2.5824924037000006E8|
| BRXX-261|2.0074834919999994E7| 4.121742455899999E8|
| BRXX-106|1.9562382400000002E7|       2.8011750861E8|
+---------+--------------------+--------------------+
only showing top 20 rows
```

18 calculating customer firm sales percentage

[59]:
```
df = df.withColumn('cust_firm_sales_percen', (col('firm_sales') /␣
↪col('global_sales')) * 100)
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 52, Finished, Available,␣
↪Finished)
```

[62]:
```
display(df.head(5))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 53, Finished, Available,␣
↪Finished)

SynapseWidget(Synapse.DataFrame, 196f763c-1327-4658-838e-e02ac59d9319)
```

[64]:
```
df = df.fillna(0, subset = 'cust_firm_sales_percen') #filling null vales with 0
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 54, Finished, Available,␣
↪Finished)
```

[66]:
```
display(df.head(5))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 55, Finished, Available,␣
↪Finished)
```

SynapseWidget(Synapse.DataFrame, 2f441b8c-7641-4ccb-a487-48721f8e4e32)

19 calculating a function for loya customer category

[67]:
```
def loyalcustcate(cust_firm_sales_percen):
    if cust_firm_sales_percen == 100:
        return "loyal customer"
    elif cust_firm_sales_percen > 75 and cust_firm_sales_percen < 100:
        return "top-tier loyal customer"
    elif cust_firm_sales_percen > 50 and cust_firm_sales_percen <=75:
        return "mid-tier loyal customer"
    elif cust_firm_sales_percen > 25 and cust_firm_sales_percen <=50:
        return "low-tier loyal customer"
    elif cust_firm_sales_percen  > 0 and cust_firm_sales_percen <=25:
        return "basic loyal customer"
    else:
        return "stangant loyal customer"
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 56, Finished, Available,␣
↪Finished)
```

[71]:
```
loyalcustcate(0)
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 57, Finished, Available,␣
↪Finished)
```

[71]: 'stangant loyal customer'

20 Now Applying this function to cust_firm_sales_percen and creating new column called custoemr_status

[72]:
```
cust_loyal_udf = udf(loyalcustcate,StringType())
df = df.
↪withColumn("customer_category",cust_loyal_udf(df['cust_firm_sales_percen']))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 58, Finished, Available,␣
↪Finished)
```

[ ]:
```
display(df.head(10))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 59, Finished, Available,␣
↪Finished)
```

SynapseWidget(Synapse.DataFrame, ec9e3eab-5801-490e-8701-f05804cd0e2b)

21 Cautomer category wise brokers count

[ ]:
```
df.groupBy('customer_category').agg(count_distinct(col('broker_id')).
↪alias("count_of_brokers"))\
                    .orderBy(desc('count_of_brokers')).show()
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 60, Finished, Available,␣
↪Finished)
```

```
+--------------------+----------------+
|   customer_category|count_of_brokers|
+--------------------+----------------+
|stangant loyal cu…|            1176|
|      loyal customer|             278|
|basic loyal customer|             232|
|top-tier loyal cu…|             205|
|low-tier loyal cu…|             176|
|mid-tier loyal cu…|             163|
+--------------------+----------------+
```

[ ]:
```
display(df.groupBy('customer_category').agg(count_distinct(col('broker_id')).
↪alias("count_of_brokers"))\
                    .orderBy(desc('count_of_brokers')))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 61, Finished, Available,␣
↪Finished)
```

```
SynapseWidget(Synapse.DataFrame, 5d991ba7-0973-4e46-9bf1-80c0f396bbd8)
```

[ ]:
```
display(df.groupBy(['broker_type','customer_category']).
↪agg(count_distinct(col('broker_id')).alias("broker_count"))\
 .orderBy(desc('broker_count')))
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 62, Finished, Available,␣
↪Finished)
```

```
SynapseWidget(Synapse.DataFrame, bb7372ad-f038-43ed-a20c-fe2ca9f105b2)
```

[61]:
```
df.write.format("delta").saveAsTable("Reatil_bank_cleaned_data_TB")
```

```
StatementMeta(, 51ae7205-cfae-427b-88c7-e248d7e647a4, 63, Finished, Available,␣
↪Finished)
```

22 Completed this Retail Banking PySpark Project in Microsoft Fabric Environment and final table psuhed to Tables section in Delat format

23 I hope you all will replciate this project. Thank you all. Keep Learning!! Keep Growing

24 *Inturi Suparna Babu*