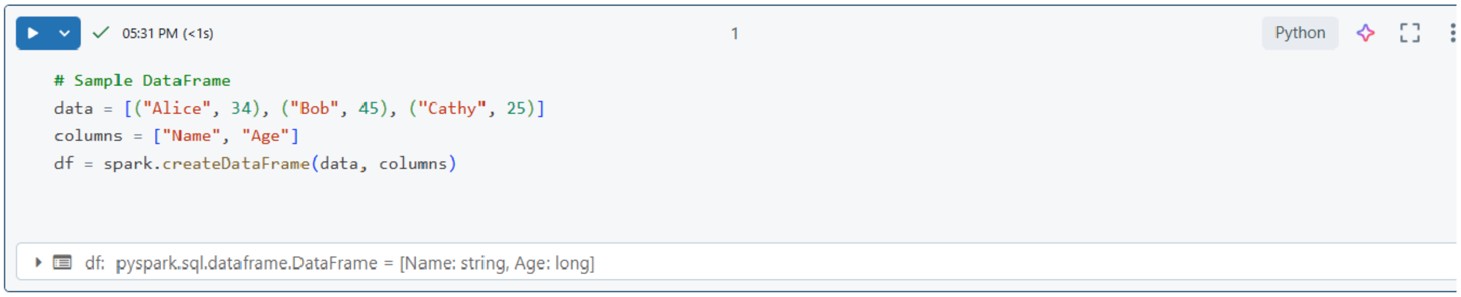
Azure Databricks Assignment

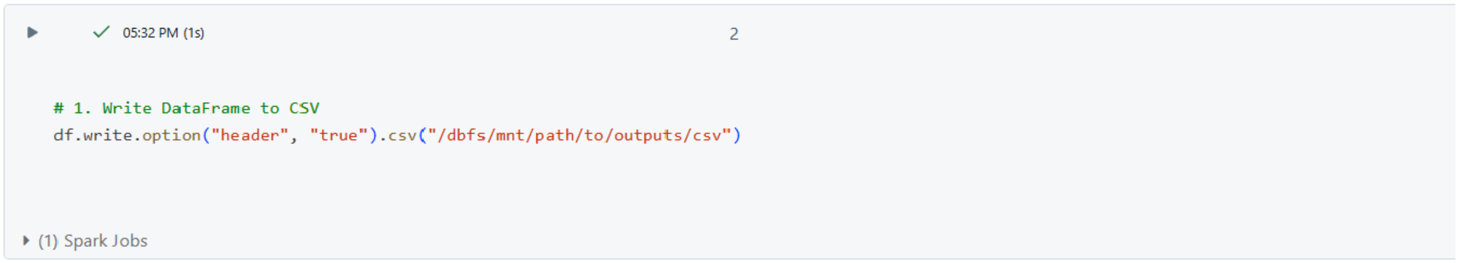
Submitted By: Aniroop Gupta
DE Batch 1

1. Write Data to CSV, JSON, Parquet, delta:



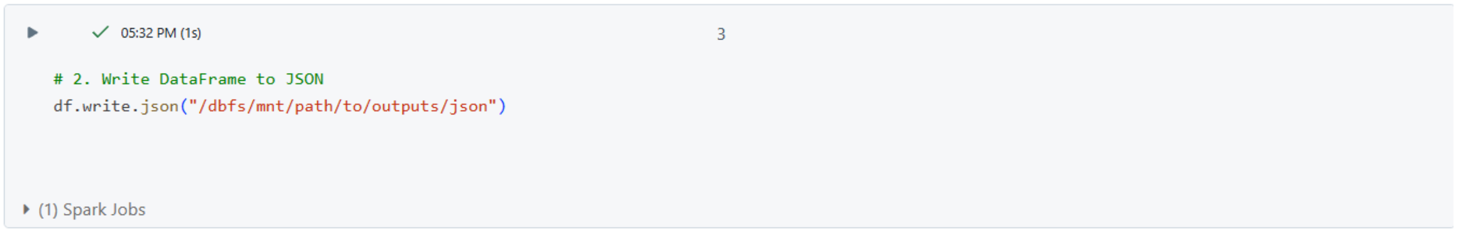
```
# Sample DataFrame
data = [("Alice", 34), ("Bob", 45), ("Cathy", 25)]
columns = ["Name", "Age"]
df = spark.createDataFrame(data, columns)
```

```
df: pyspark.sql.dataframe.DataFrame = [Name: string, Age: long]
```



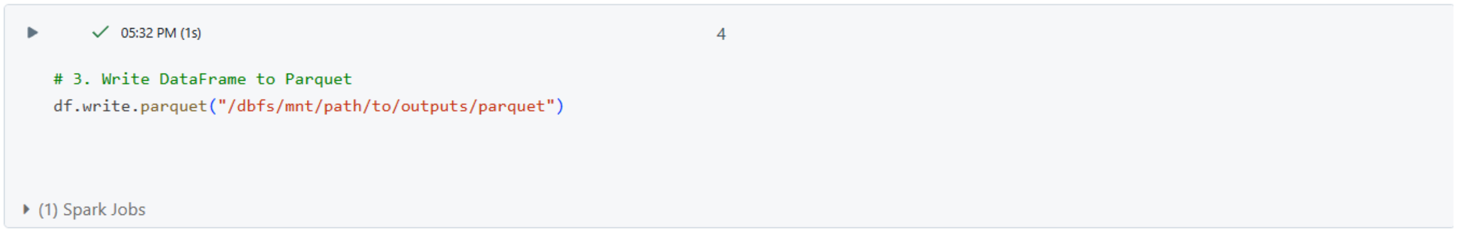
```
# 1. Write DataFrame to CSV
df.write.option("header", "true").csv("/dbfs/mnt/path/to/outputs/csv")
```

```
(1) Spark Jobs
```



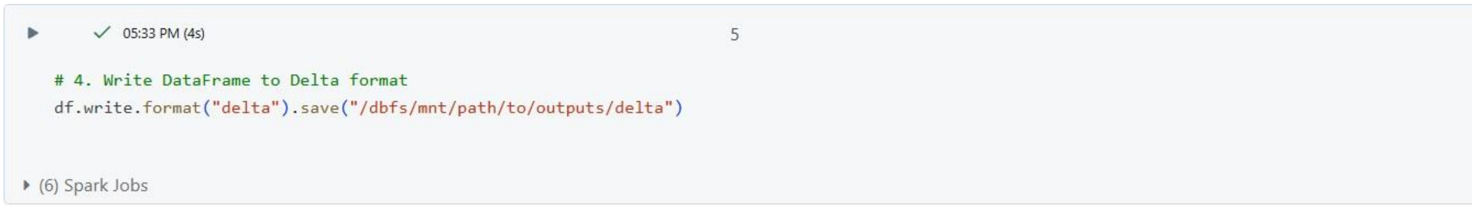
```
# 2. Write DataFrame to JSON
df.write.json("/dbfs/mnt/path/to/outputs/json")
```

```
(1) Spark Jobs
```



```
# 3. Write DataFrame to Parquet
df.write.parquet("/dbfs/mnt/path/to/outputs/parquet")
```

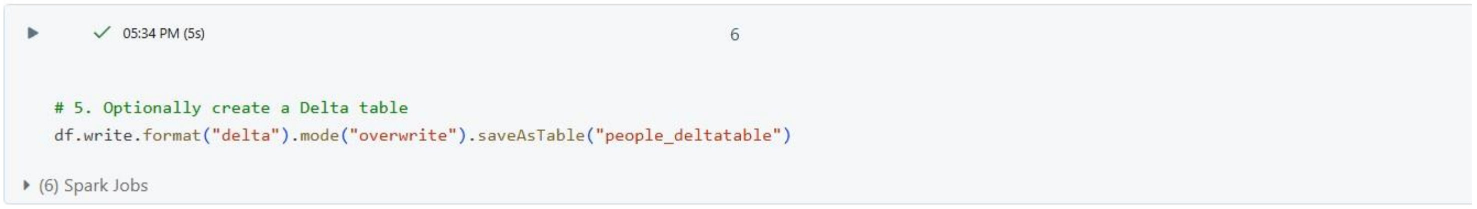
```
(1) Spark Jobs
```



05:33 PM (4s) 5

```
# 4. Write DataFrame to Delta format
df.write.format("delta").save("/dbfs/mnt/path/to/outputs/delta")
```

(6) Spark Jobs

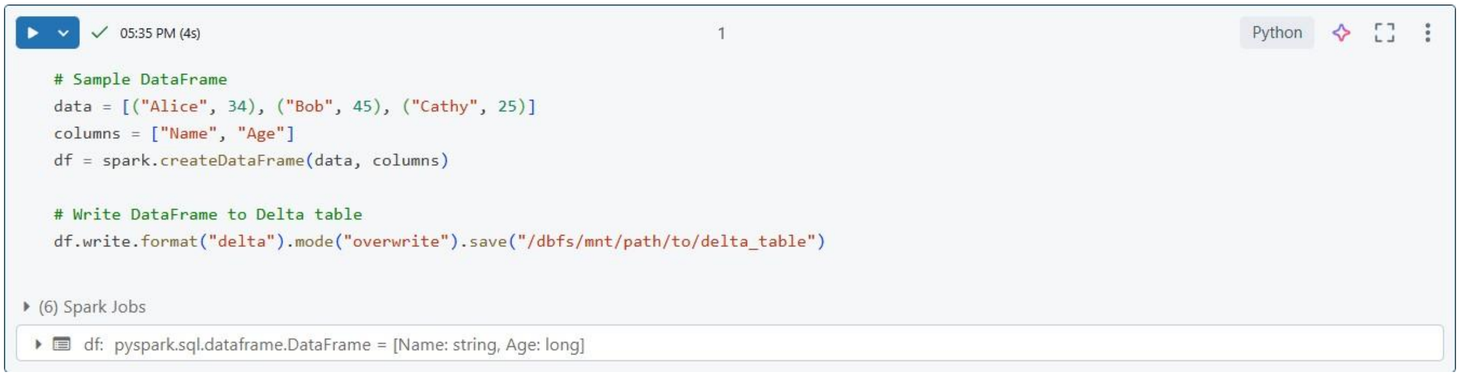


05:34 PM (5s) 6

```
# 5. Optionally create a Delta table
df.write.format("delta").mode("overwrite").saveAsTable("people_deltatable")
```

(6) Spark Jobs

2. Writing dataframe to Delta Table:

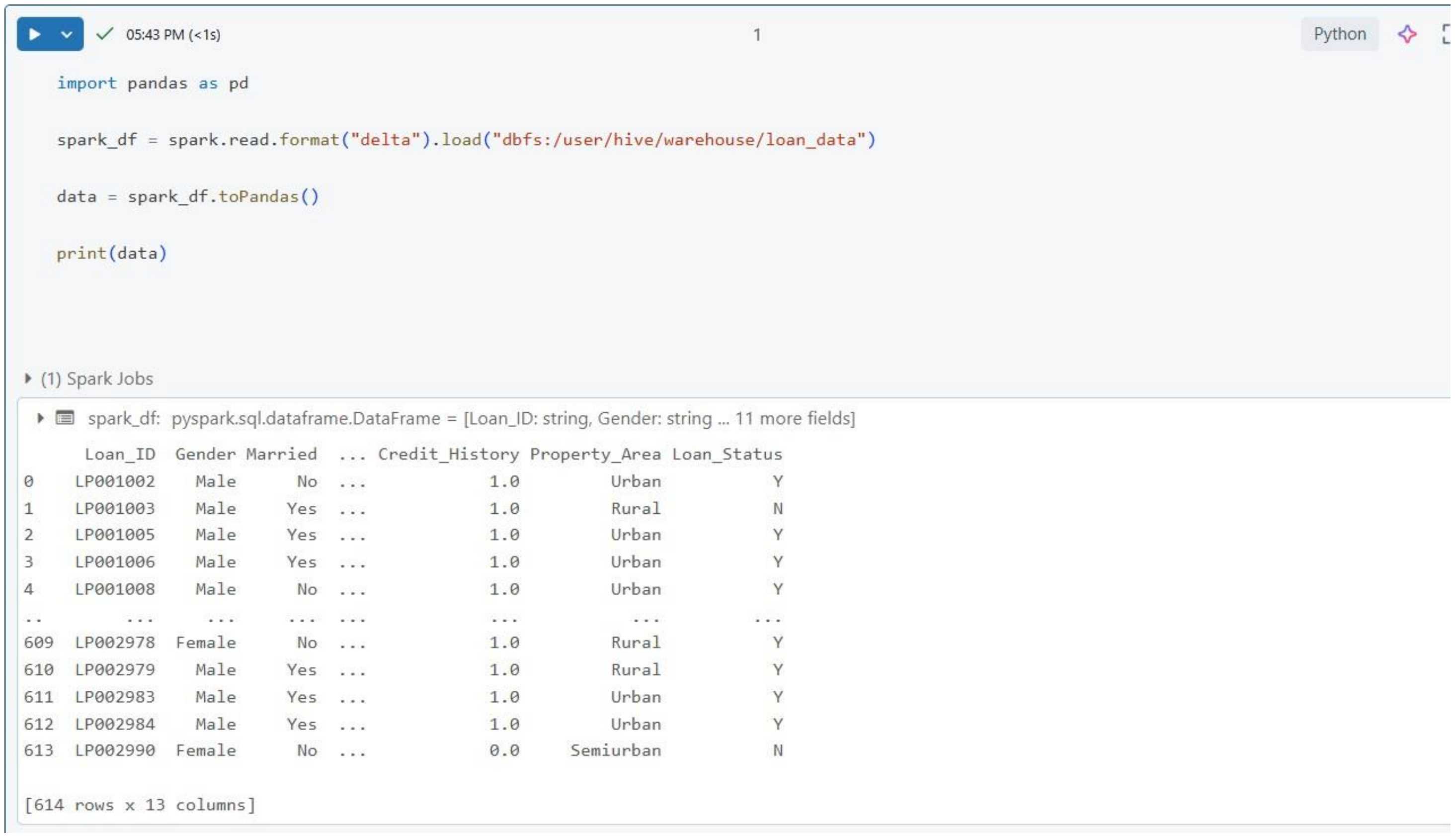


```
# Sample DataFrame
data = [("Alice", 34), ("Bob", 45), ("Cathy", 25)]
columns = ["Name", "Age"]
df = spark.createDataFrame(data, columns)

# Write DataFrame to Delta table
df.write.format("delta").mode("overwrite").save("/dbfs/mnt/path/to/delta_table")
```

(6) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Name: string, Age: long]

3. Exploratory data analysis (EDA) in Databricks:

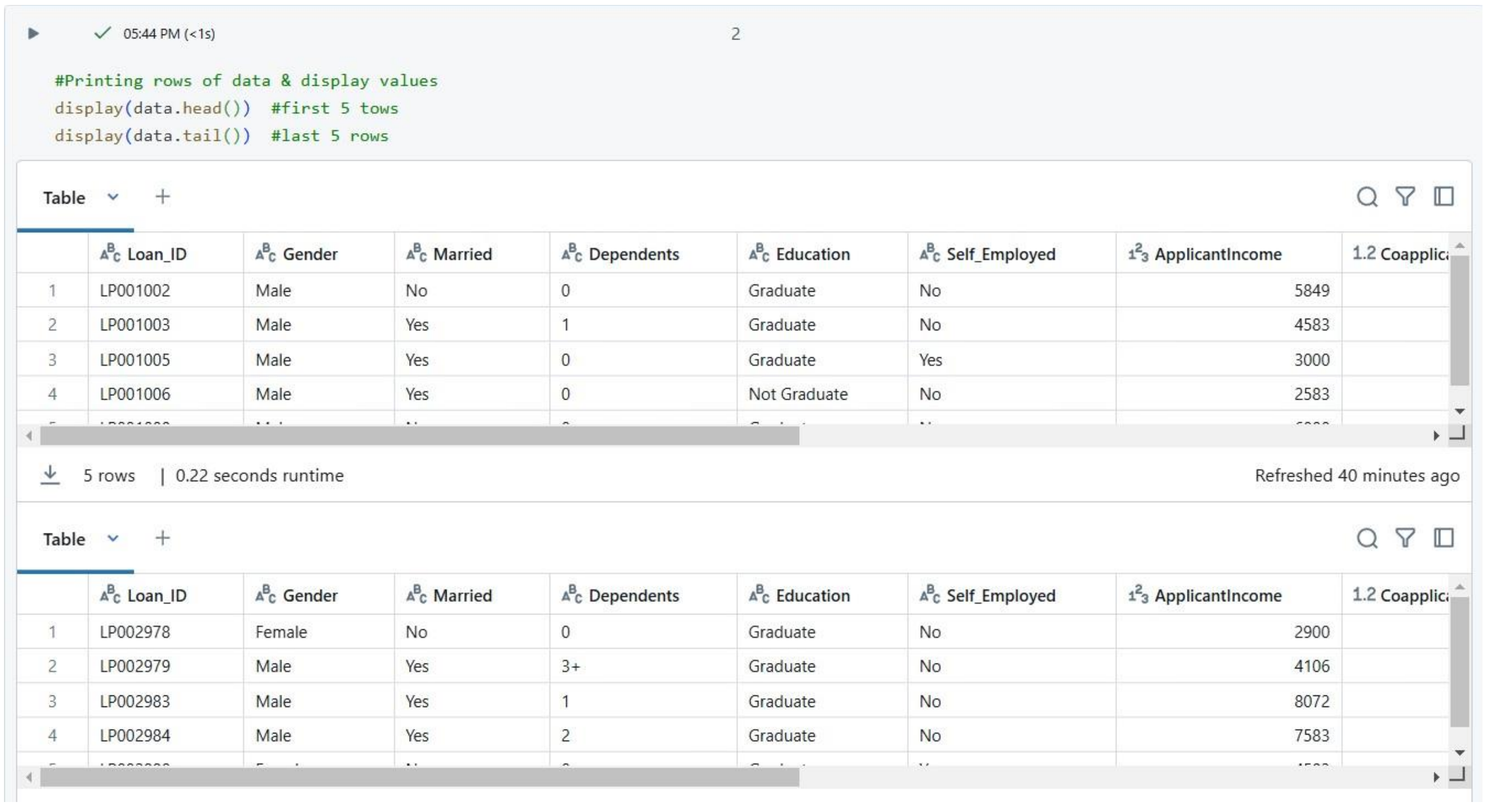


```
import pandas as pd

spark_df = spark.read.format("delta").load("dbfs:/user/hive/warehouse/loan_data")

data = spark_df.toPandas()

print(data)
```

(1) Spark Jobs

spark_df: pyspark.sql.dataframe.DataFrame = [Loan_ID: string, Gender: string ... 11 more fields]

```
      Loan_ID  Gender Married  ... Credit_History Property_Area Loan_Status
0    LP001002    Male      No  ...            1.0         Urban           Y
1    LP001003    Male     Yes  ...            1.0         Rural           N
2    LP001005    Male     Yes  ...            1.0         Urban           Y
3    LP001006    Male     Yes  ...            1.0         Urban           Y
4    LP001008    Male      No  ...            1.0         Urban           Y
..        ...     ...     ...  ...            ...           ...         ...
609  LP002978  Female      No  ...            1.0         Rural           Y
610  LP002979    Male     Yes  ...            1.0         Rural           Y
611  LP002983    Male     Yes  ...            1.0         Urban           Y
612  LP002984    Male     Yes  ...            1.0         Urban           Y
613  LP002990  Female      No  ...            0.0     Semiurban           N

[614 rows x 13 columns]
```



05:44 PM (<1s) 2

```
#Printing rows of data & display values
display(data.head())  #first 5 tows
display(data.tail())  #last 5 rows
```

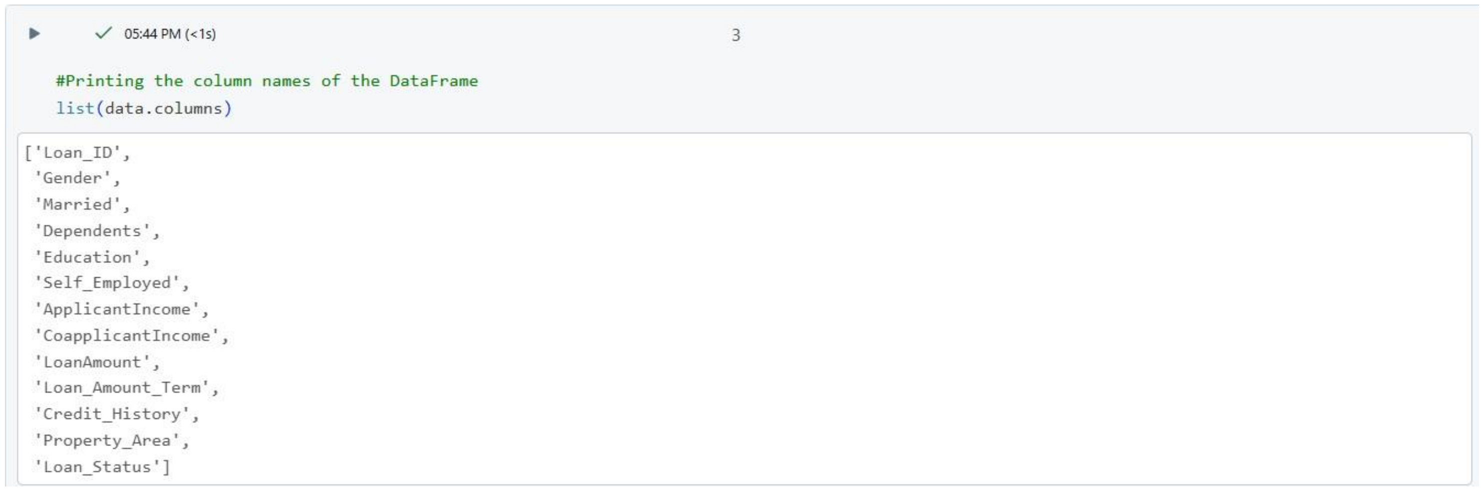
Table

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	1.2 Coapplic
1	LP001002	Male	No	0	Graduate	No	5849	
2	LP001003	Male	Yes	1	Graduate	No	4583	
3	LP001005	Male	Yes	0	Graduate	Yes	3000	
4	LP001006	Male	Yes	0	Not Graduate	No	2583	

5 rows | 0.22 seconds runtime Refreshed 40 minutes ago

Table

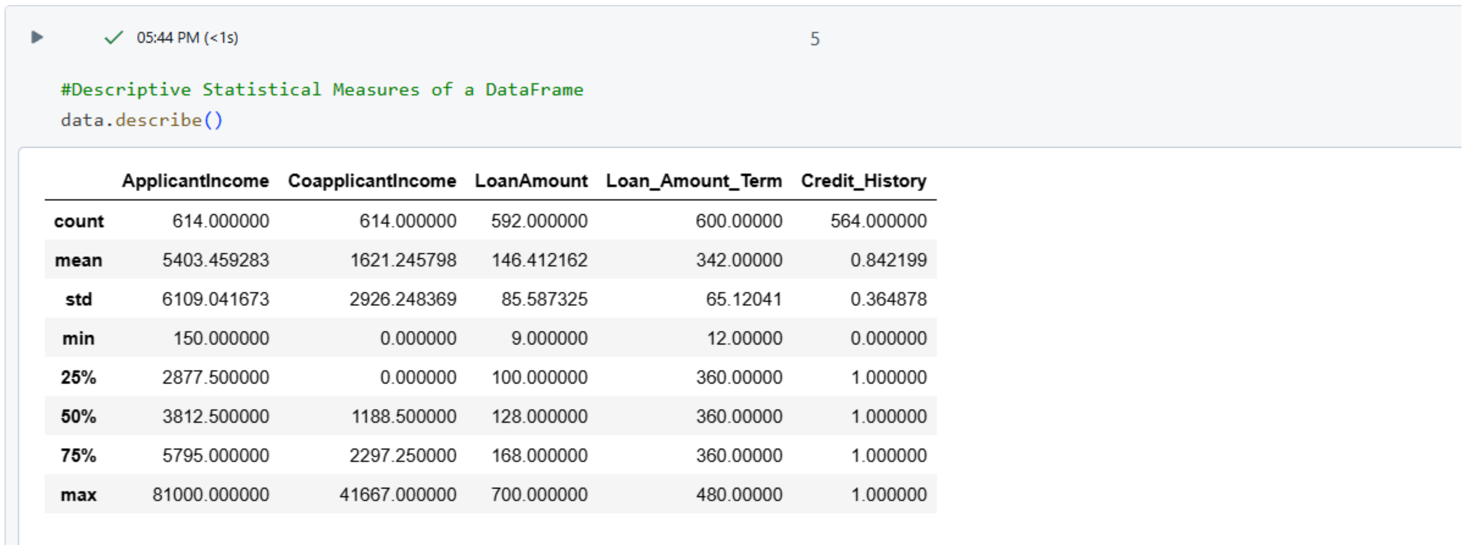
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	1.2 Coapplic
1	LP002978	Female	No	0	Graduate	No	2900	
2	LP002979	Male	Yes	3+	Graduate	No	4106	
3	LP002983	Male	Yes	1	Graduate	No	8072	
4	LP002984	Male	Yes	2	Graduate	No	7583	



05:44 PM (<1s) 3

```
#Printing the column names of the DataFrame
list(data.columns)
```

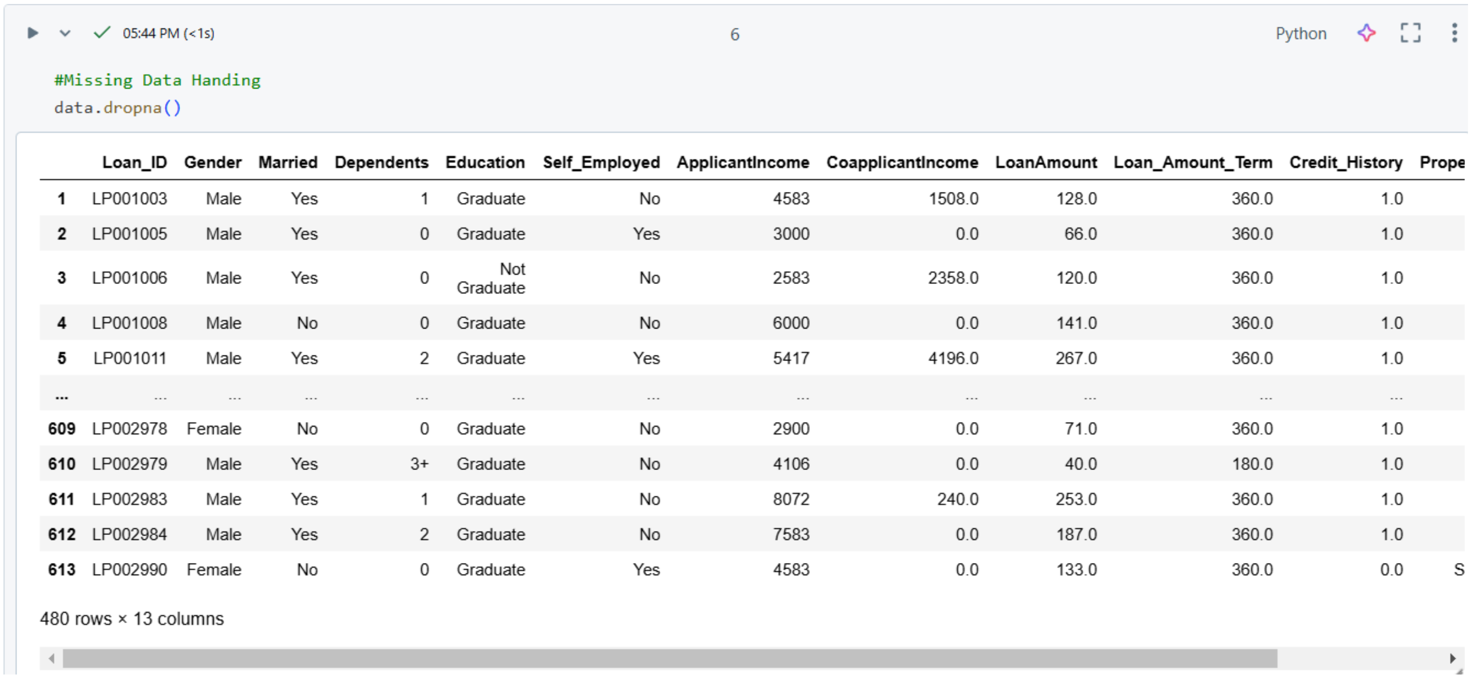
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status']
```



05:44 PM (<1s) 5

```
#Descriptive Statistical Measures of a DataFrame
data.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

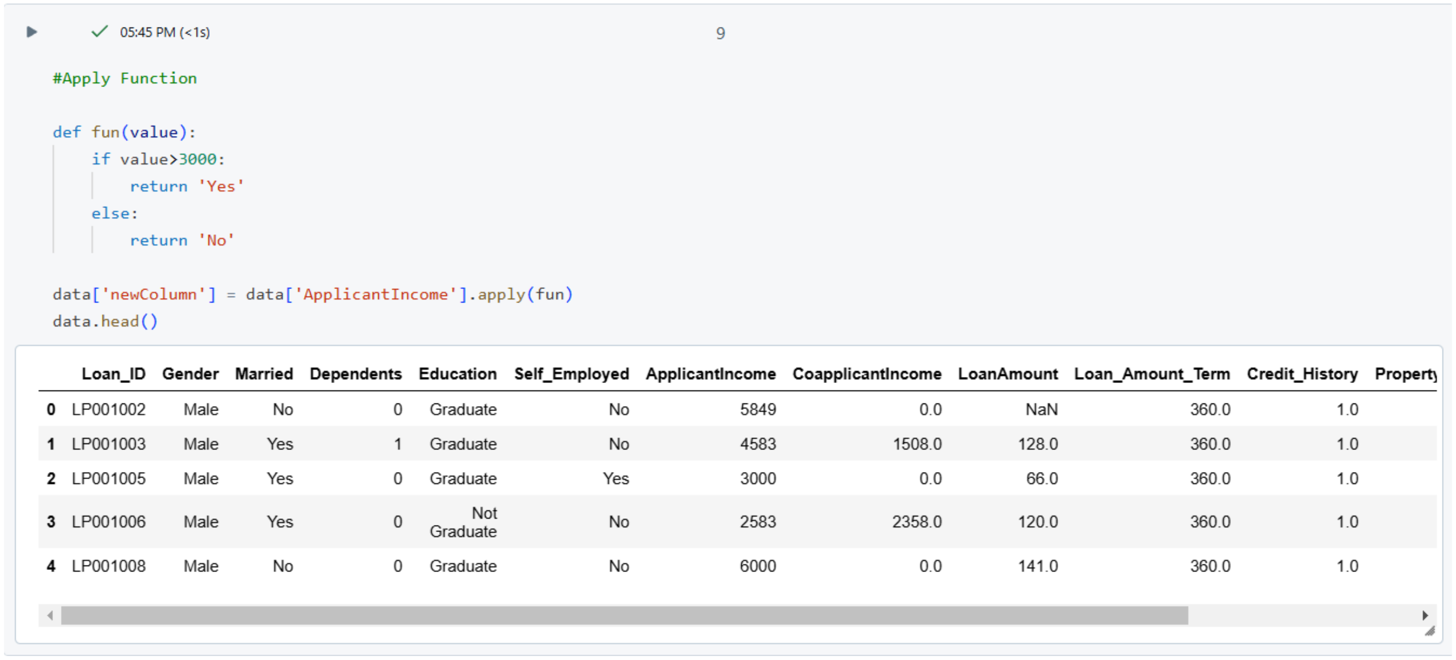


05:44 PM (<1s) 6 Python

```python
#Missing Data Handing
data.dropna()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Prope
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	S

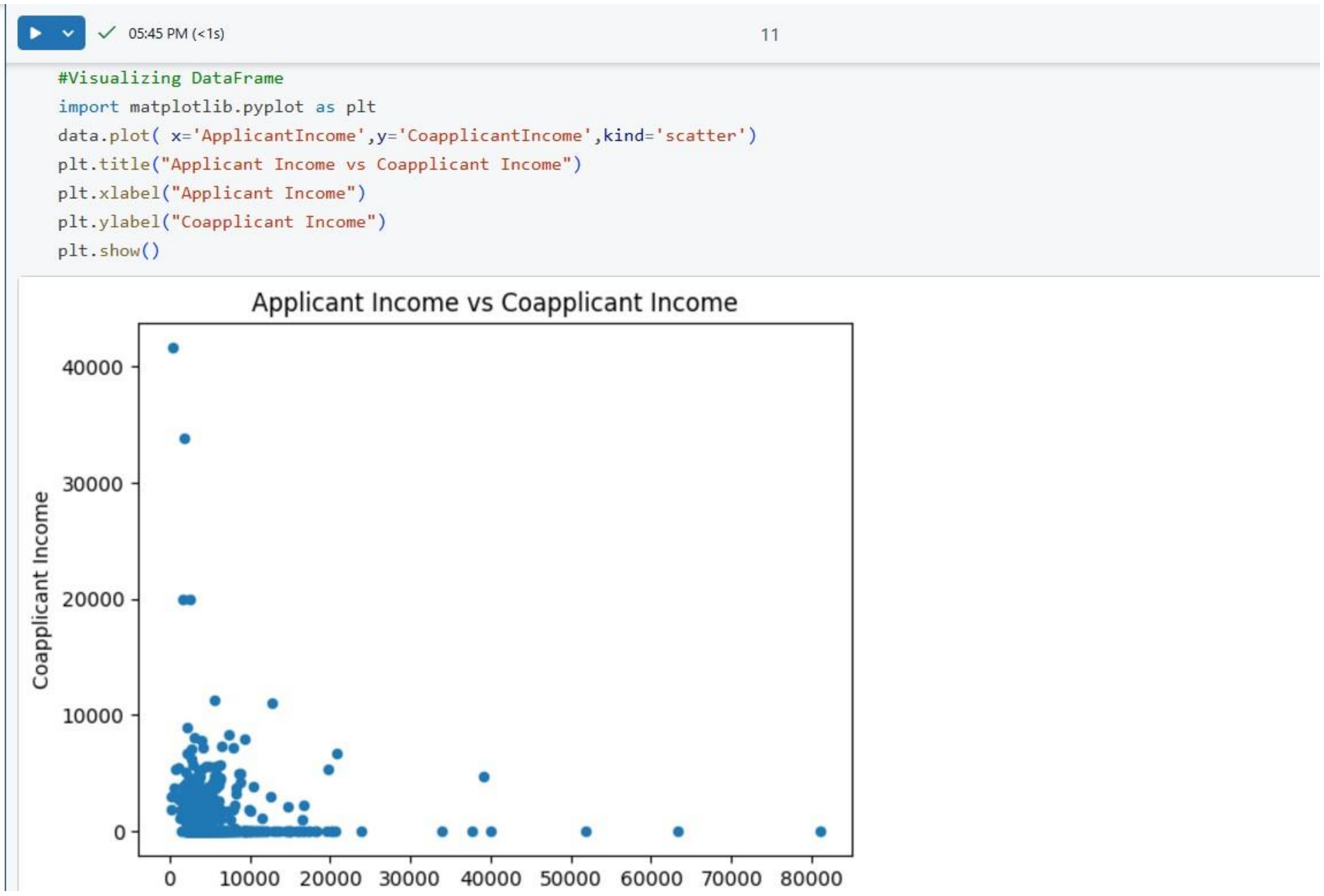
480 rows × 13 columns



05:45 PM (<1s) 9

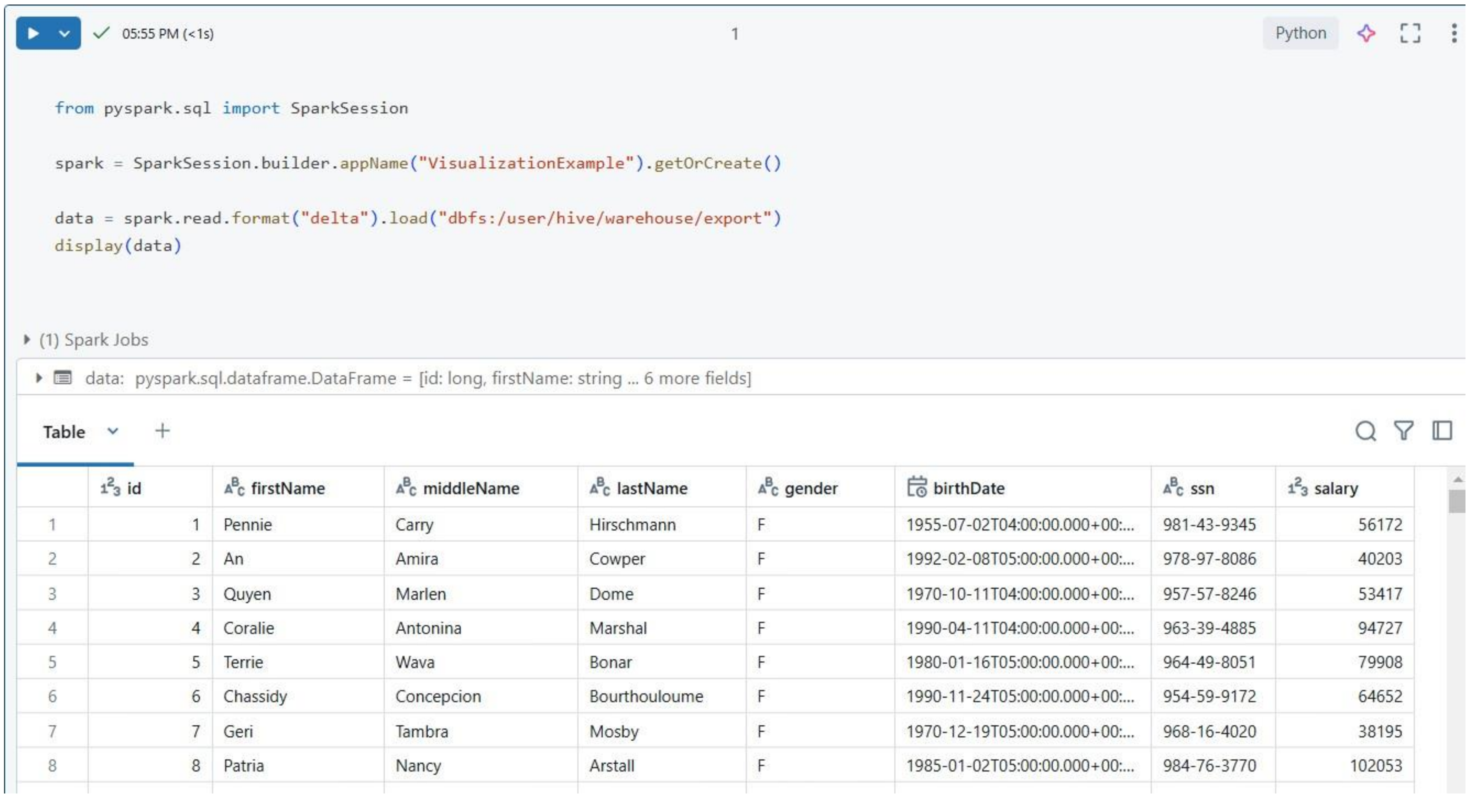
```python
#Apply Function

def fun(value):
    if value>3000:
        return 'Yes'
    else:
        return 'No'

data['newColumn'] = data['ApplicantIncome'].apply(fun)
data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	



```
#Visualizing DataFrame
import matplotlib.pyplot as plt
data.plot( x='ApplicantIncome',y='CoapplicantIncome',kind='scatter')
plt.title("Applicant Income vs Coapplicant Income")
plt.xlabel("Applicant Income")
plt.ylabel("Coapplicant Income")
plt.show()
```

4. Data Exploration and Visualization in Databricks:



```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("VisualizationExample").getOrCreate()

data = spark.read.format("delta").load("dbfs:/user/hive/warehouse/export")
display(data)
```

data: pyspark.sql.dataframe.DataFrame = [id: long, firstName: string ... 6 more fields]

	id	firstName	middleName	lastName	gender	birthDate	ssn	salary
1	1	Pennie	Carry	Hirschmann	F	1955-07-02T04:00:00.000+00:...	981-43-9345	56172
2	2	An	Amira	Cowper	F	1992-02-08T05:00:00.000+00:...	978-97-8086	40203
3	3	Quyen	Marlen	Dome	F	1970-10-11T04:00:00.000+00:...	957-57-8246	53417
4	4	Coralie	Antonina	Marshal	F	1990-04-11T04:00:00.000+00:...	963-39-4885	94727
5	5	Terrie	Wava	Bonar	F	1980-01-16T05:00:00.000+00:...	964-49-8051	79908
6	6	Chassidy	Concepcion	Bourthouloume	F	1990-11-24T05:00:00.000+00:...	954-59-9172	64652
7	7	Geri	Tambra	Mosby	F	1970-12-19T05:00:00.000+00:...	968-16-4020	38195
8	8	Patria	Nancy	Arstall	F	1985-01-02T05:00:00.000+00:...	984-76-3770	102053

05:52 PM (<1s) 2

```
# Print the schema to understand data types and column structure
data.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- firstName: string (nullable = true)
 |-- middleName: string (nullable = true)
 |-- lastName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- birthDate: timestamp (nullable = true)
 |-- ssn: string (nullable = true)
 |-- salary: long (nullable = true)
```

05:52 PM (1s) 3

```
# Show summary statistics for numeric columns
data.describe().show()
```

(2) Spark Jobs

```
+-------+-----------------+---------+----------+--------+------+-----------+------------------+
|summary|               id|firstName|middleName|lastName|gender|        ssn|            salary|
+-------+-----------------+---------+----------+--------+------+-----------+------------------+
|  count|             1000|     1000|      1000|    1000|  1000|       1000|              1000|
|   mean|            500.5|     NULL|      NULL|    NULL|  NULL|       NULL|          72763.54|
| stddev|288.8194360957494|     NULL|      NULL|    NULL|  NULL|       NULL|20670.644326853664|
|    min|                1|    Abbie|    Adella| Abrahim|     F|666-15-8671|             -6523|
|    max|             1000|     Zita|     Zulma|Zannutti|     F|999-93-2044|            134393|
+-------+-----------------+---------+----------+--------+------+-----------+------------------+
```

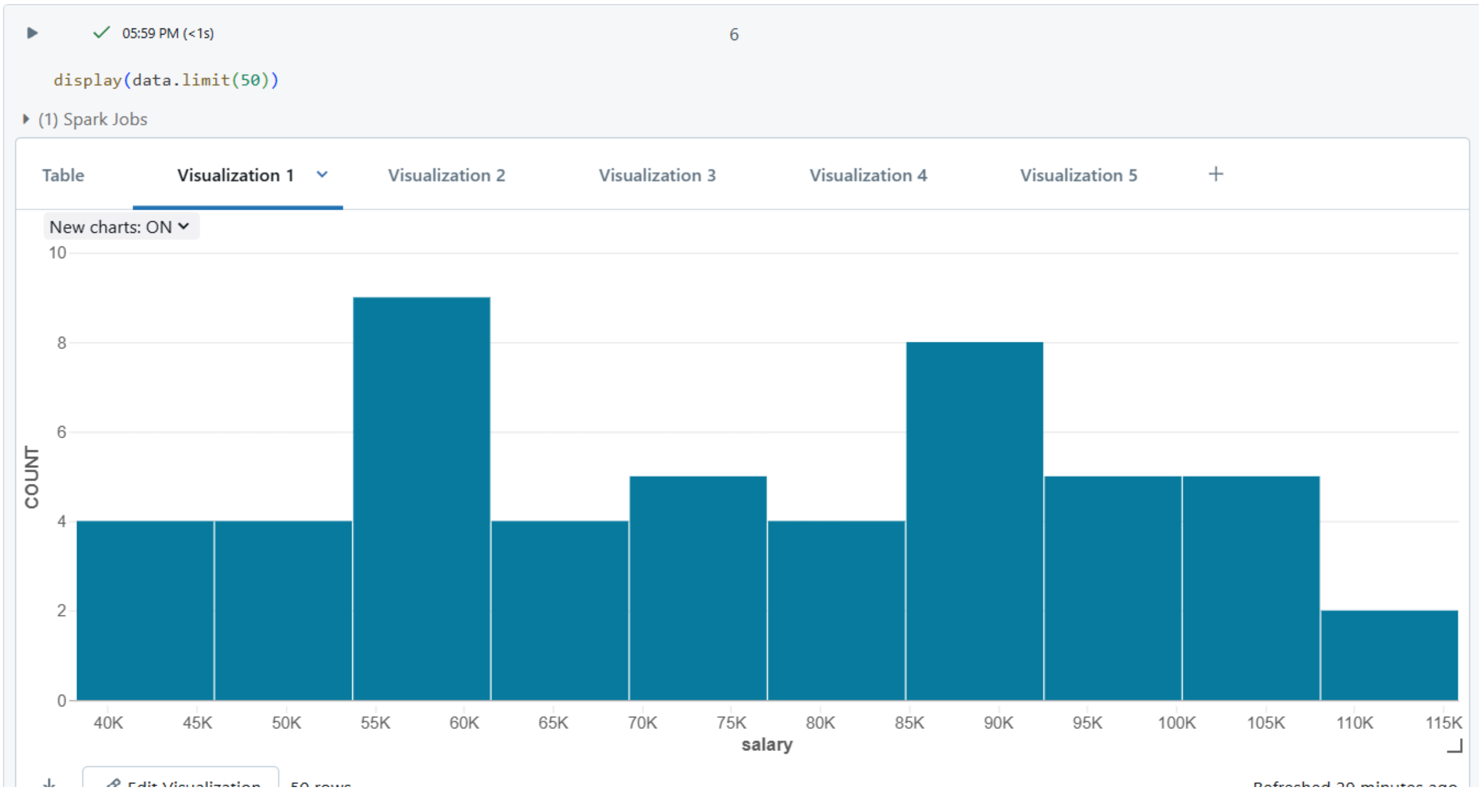
05:52 PM (<1s) 4

```
# Count the number of rows and columns
num_rows = data.count()
num_cols = len(data.columns)
print(f"Rows: {num_rows}, Columns: {num_cols}")
```

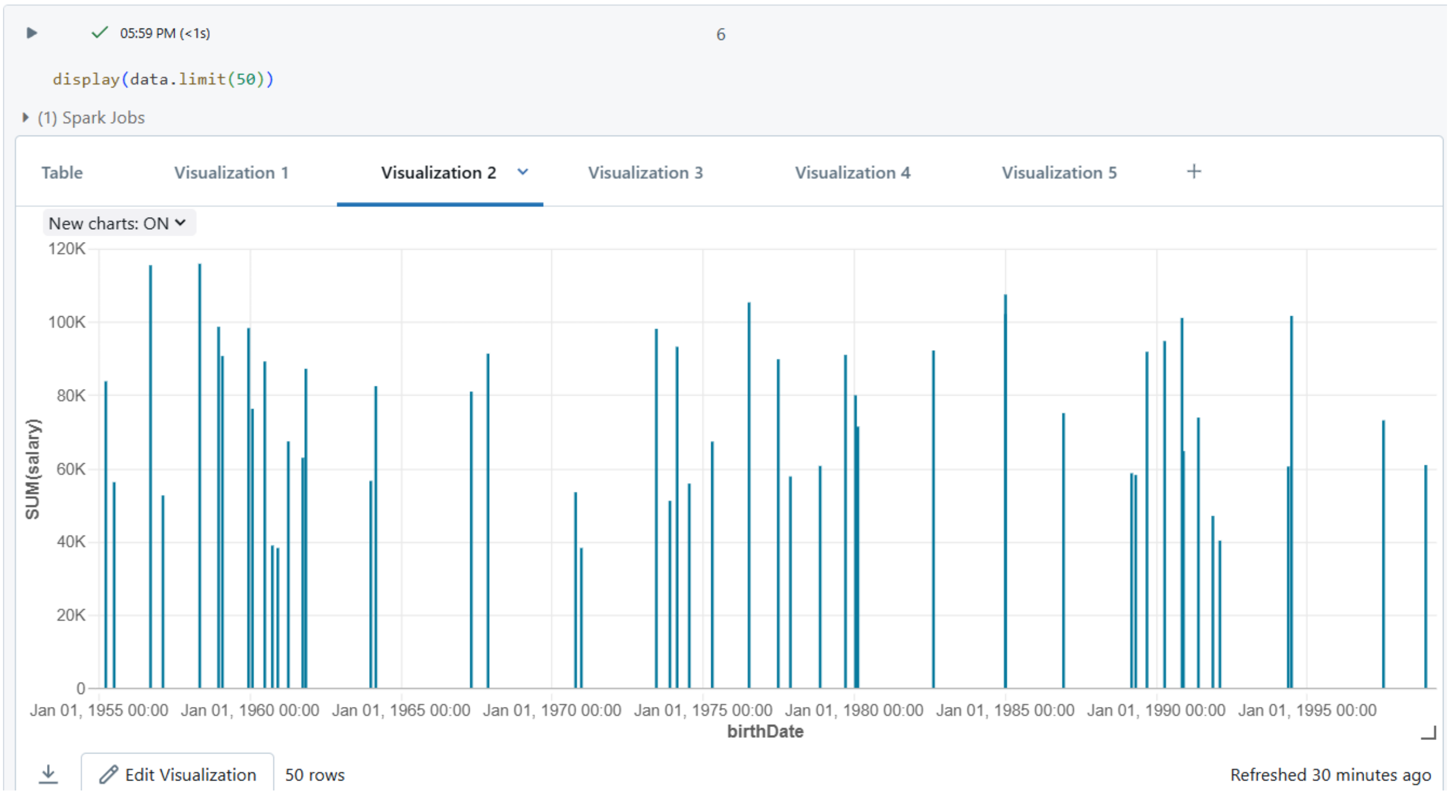
(2) Spark Jobs

```
Rows: 1000, Columns: 8
```

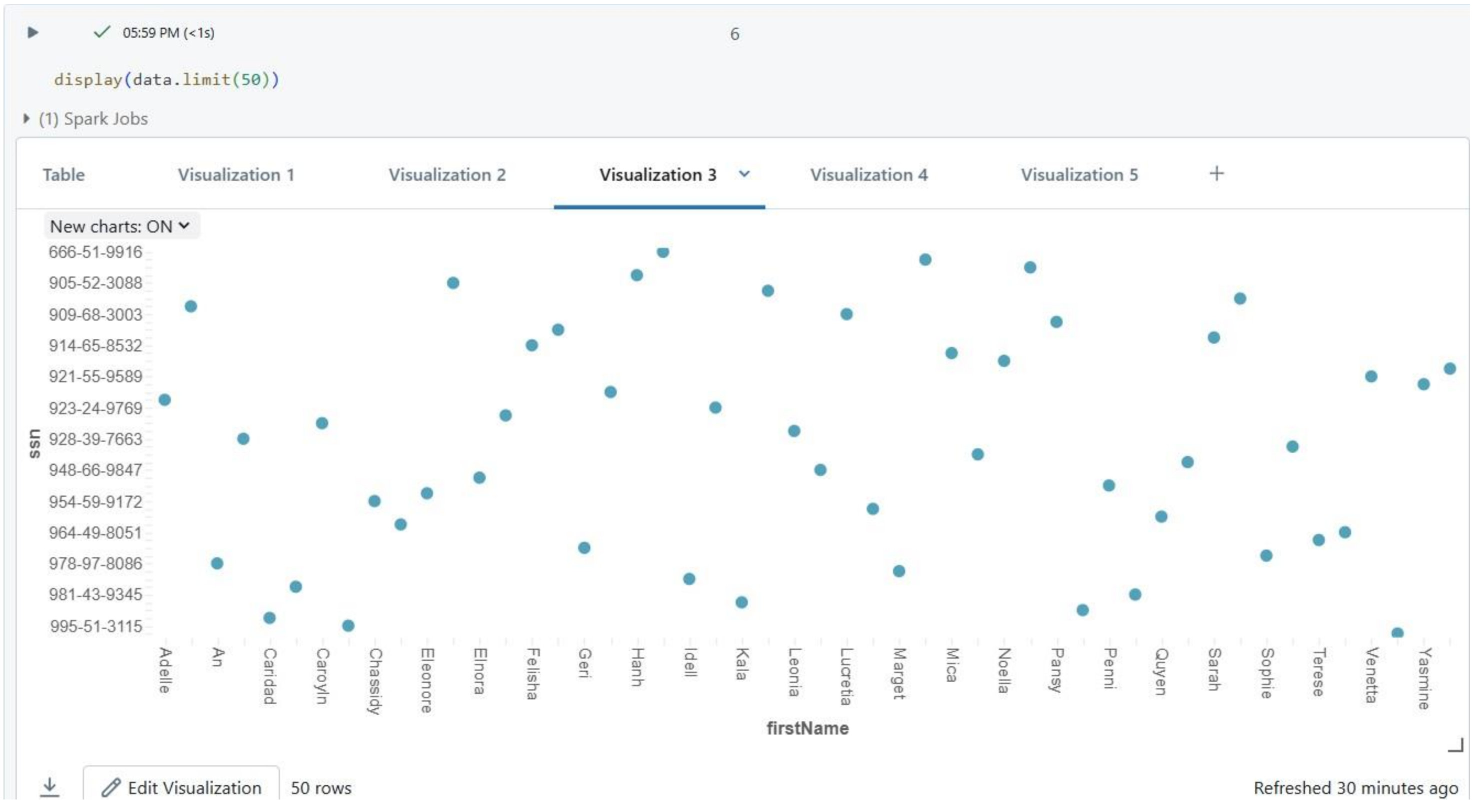
Visualization:



```
display(data.limit(50))
```



```
display(data.limit(50))
```

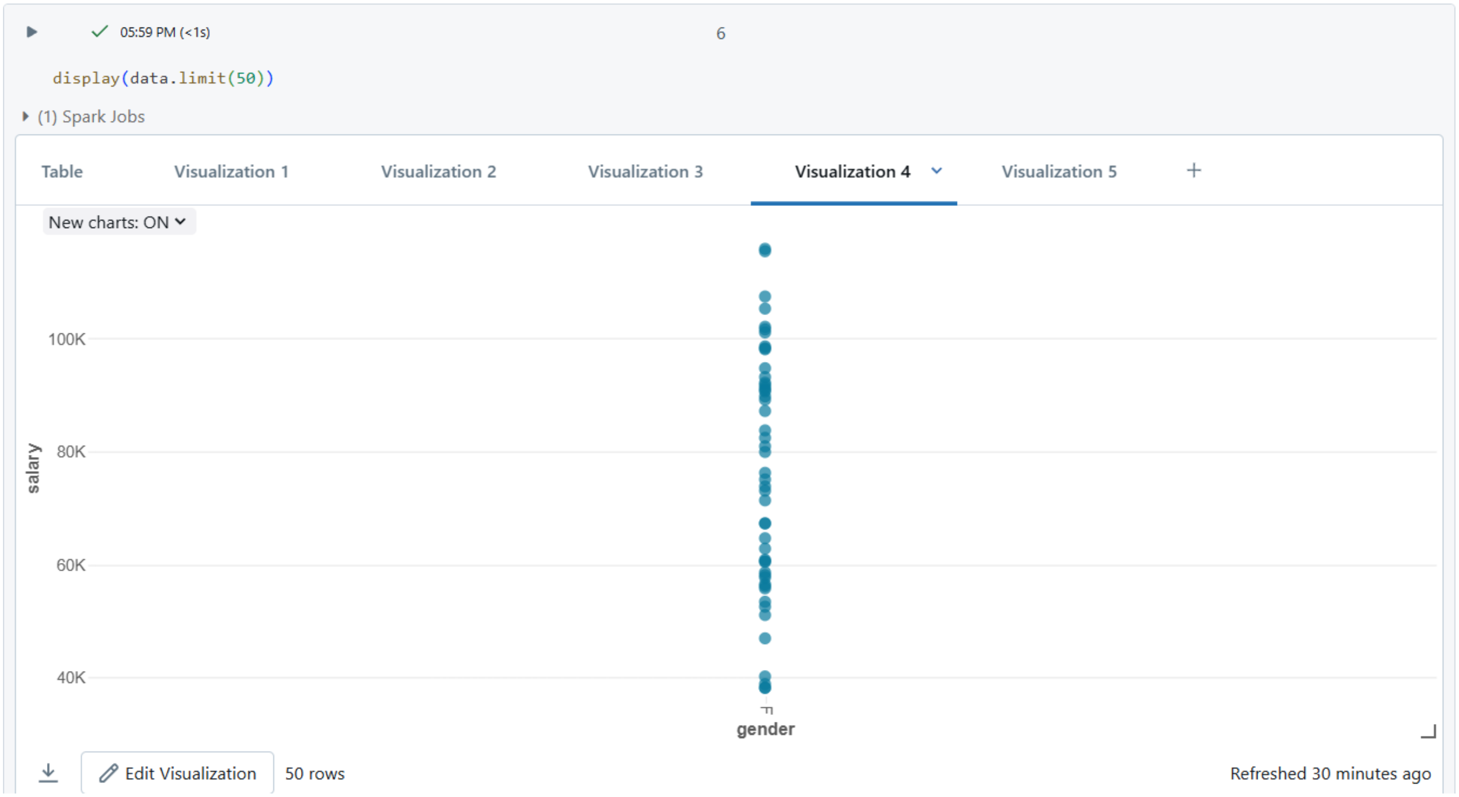


```
05:59 PM (<1s)                                    6

display(data.limit(50))
```

(1) Spark Jobs

50 rows

Refreshed 30 minutes ago

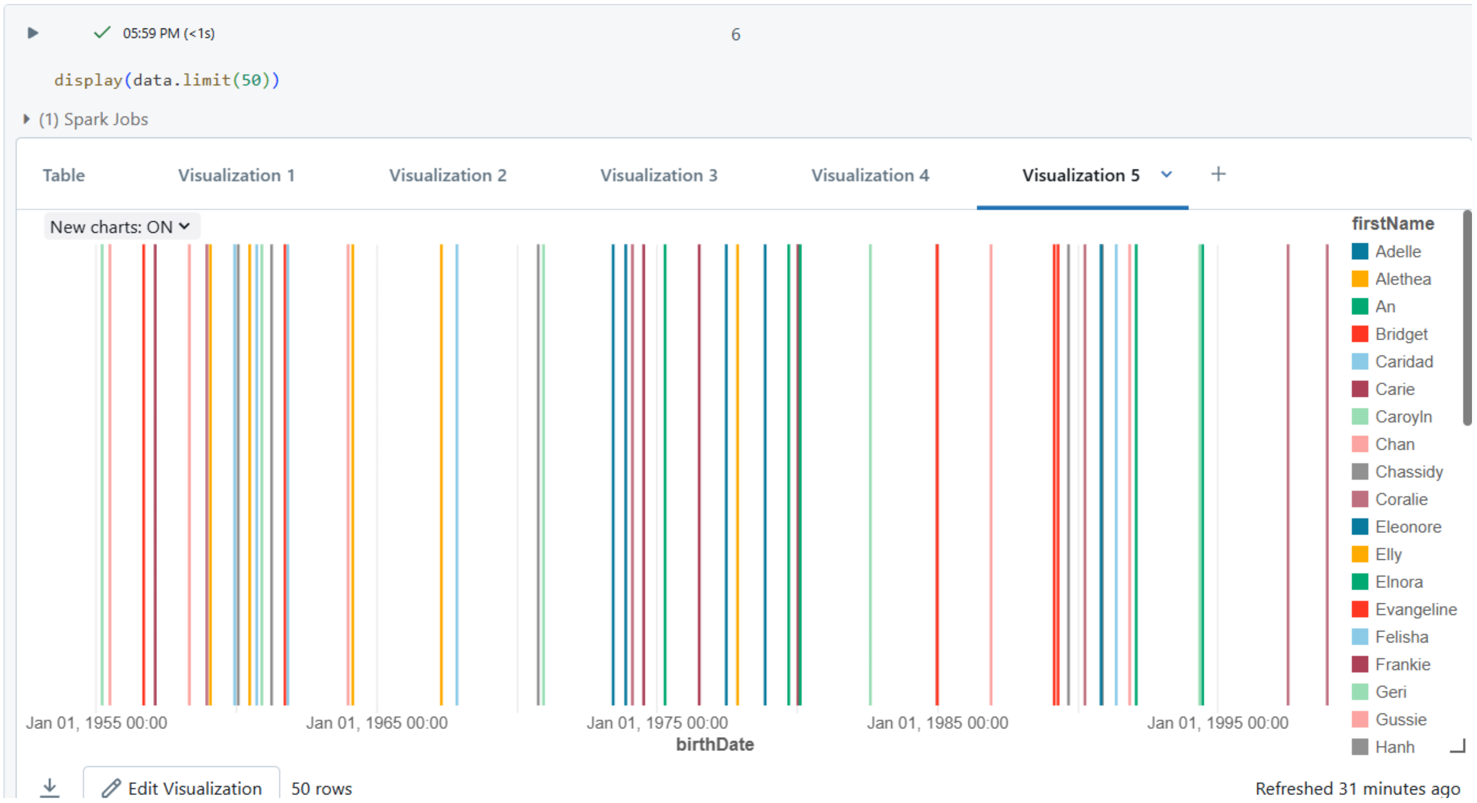


```
05:59 PM (<1s)                                    6

display(data.limit(50))
```

(1) Spark Jobs

50 rows

Refreshed 30 minutes ago



05:59 PM (<1s)

6

```
display(data.limit(50))
```

(1) Spark Jobs

50 rows

Refreshed 31 minutes ago