Views, ETL and widgets 5th Sep

SaiPrabath Chowdary S

Views :

```
[48] import pandas as pd

     data = {
         "name": ["John", "Jane", "Mike", "Emily"],
         "age": [28, 32, 55, 23],
         "gender": ["Male", "Female", "Male", "Female"],
         "city": ["New York", "San Francisco", "Los Angeles", "Chicago"]
     }

     df = pd.DataFrame(data)

     # Save DF into csv
     csv_file_path = '/content/drive/MyDrive/DataEngineering/sample_people.csv'
     df.to_csv(csv_file_path, index=False)
```

```
     from pyspark.sql import SparkSession
     from pyspark.sql.functions import col

     # Initialize a Spark session
     spark = SparkSession.builder \
         .appName("Create View Example") \
         .getOrCreate()
```

```
[89] # loading a CSV
     csv_path = '/content/drive/MyDrive/DataEngineering/people_data.csv'
     people_df = spark.read.format("csv").option("header", "true").load(csv_file_path)
     people_df.show()
```

```
+-----+---+------+-------------+
| name|age|gender|         city|
+-----+---+------+-------------+
| John| 28|  Male|     New York|
| Jane| 32|Female|San Francisco|
| Mike| 55|  Male|  Los Angeles|
|Emily| 23|Female|      Chicago|
+-----+---+------+-------------+
```

```
[107] # creating local temporary view -> only this session
      people_df.createOrReplaceTempView("people_temp_view")

      # Running SQL query on view
      temp_view_result = spark.sql("SELECT * FROM people_temp_view WHERE age > 30")
      temp_view_result.show()
```

```
+----+---+------+-------------+
|name|age|gender|         city|
+----+---+------+-------------+
|Jane| 32|Female|San Francisco|
|Mike| 55|  Male|  Los Angeles|
+----+---+------+-------------+
```

```
      # creating global temporary view -> accessible in all sessions
      people_df.createOrReplaceGlobalTempView("people_global_view")

      # Running SQL query on global view
      global_view_result = spark.sql("SELECT name,age,city FROM global_temp.people_global_view WHERE gender = 'Male'")
      global_view_result.show()
```

```
+----+---+-----------+
|name|age|       city|
+----+---+-----------+
|John| 28|   New York|
|Mike| 55|Los Angeles|
+----+---+-----------+
```

```
[108] # list all temp views and tables
      spark.catalog.listTables()
```

```
[Table(name='people_temp_view', catalog=None, namespace=[], description=None, tableType='TEMPORARY', isTemporary=True)]
```

```
[109] # drop local temp view
      spark.catalog.dropTempView("people_temp_view")

      # drop global temp view
      spark.catalog.dropGlobalTempView("people_global_view")

      spark.catalog.listTables()
```

```
[]
```

ETL:

```
# https://codeshare.io/deEM8e
# Create a new database in Spark SQL
spark.sql("CREATE DATABASE IF NOT EXISTS my_database")

# Use the created database
spark.sql("USE my_database")

# Verify that the database is being used
spark.sql("SHOW DATABASES").show()
```

```
+-----------+
|  namespace|
+-----------+
|    default|
|my_database|
+-----------+
```

```
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Initialize a Spark session
spark = SparkSession.builder.appName("EmployeeETL").getOrCreate()

# Create a sample CSV data
data = {
    "name": ["John", "Jane", "Mike", "Emily", "Alex"],
    "age": [28, 32, 45, 23, 36],
    "gender": ["Male", "Female", "Male", "Female", "Male"],
    "salary": [60000, 72000, 84000, 52000, 67000]
}

df = pd.DataFrame(data)

# Save the DataFrame as a CSV file
csv_file_path = "/content/drive/MyDrive/DataEngineering/sample_people.csv"
df.to_csv(csv_file_path, index=False)

# Confirm the CSV file is created
print(f"CSV file created at: {csv_file_path}")
```

```
CSV file created at: /content/drive/MyDrive/DataEngineering/sample_people.csv
```

```
[117] # 1. Extract: Load the employee data from a CSV file containing the following columns: `name`, `age`, `gender`, and `salary`.
      employee_df = spark.read.csv(csv_file_path, header=True, inferSchema=True)
      employee_df.show()
```

```
+-----+---+------+------+
| name|age|gender|salary|
+-----+---+------+------+
| John| 28|  Male| 60000|
| Jane| 32|Female| 72000|
| Mike| 45|  Male| 84000|
|Emily| 23|Female| 52000|
| Alex| 36|  Male| 67000|
+-----+---+------+------+
```

```
# 2. Transform:
# Filter: Only include employees aged 30 and above in the analysis.
filtered_df1 = employee_df.filter(col("age") >= 30)
print("employees aged 30:")
filtered_df1.show()

# Add New Column: Calculate a 10% bonus on the current salary for each employee and add it as a new column (`salary_with_bonus`).
filtered_df2 = employee_df.withColumn("salary_with_bonus", col("salary") * 1.1)
print("employees with bonus:")
filtered_df2.show()

# Aggregation: Group the employees by gender and compute the average salary for each gender.
from pyspark.sql.functions import avg
avg_salary_by_gender = filtered_df2.groupBy("gender").agg(avg("salary").alias("avg_salary"))
print("average salary by gender:")
avg_salary_by_gender.show()
```

```
employees aged 30:
+----+---+------+------+
|name|age|gender|salary|
+----+---+------+------+
|Jane| 32|Female| 72000|
|Mike| 45|  Male| 84000|
|Alex| 36|  Male| 67000|
+----+---+------+------+

employees with bonus:
+-----+---+------+------+------------------+
| name|age|gender|salary|salary_with_bonus|
+-----+---+------+------+------------------+
| John| 28|  Male| 60000|           66000.0|
| Jane| 32|Female| 72000|           79200.0|
| Mike| 45|  Male| 84000|92400.00000000001|
|Emily| 23|Female| 52000|57200.00000000001|
| Alex| 36|  Male| 67000|           73700.0|
+-----+---+------+------+------------------+

average salary by gender:
+------+-----------------+
|gender|       avg_salary|
+------+-----------------+
|Female|          62000.0|
|  Male|70333.33333333333|
+------+-----------------+
```

```
# Load Data
filtered_df2.write.parquet("/content/drive/MyDrive/DataEngineering/employee_data_transformed.parquet","overwrite")
print("Data loaded successfully!")
```

```
Data loaded successfully!
```

Full Refresh:

```
[128] from pyspark.sql import SparkSession
      from pyspark.sql.functions import col

      # Initialize a Spark session
      spark = SparkSession.builder.appName("Full Refresh Example").getOrCreate()
```

```
# full refresh load data
csv_file = "/content/drive/MyDrive/DataEngineering/sales_data.csv"
sales_df = spark.read.csv(csv_file, header=True, inferSchema=True)

# Transformation
sales_df_with_totalsales = sales_df.withColumn("TotalSales", col("Quantity") * col("Price"))
sales_df_with_totalsales.show()

# full refresh -> partition by date
output_path = "/content/sample_data/partioned_data_fullrefresh"
sales_df_with_totalsales.write.partitionBy("Date").mode("overwrite").parquet(output_path) # override mode
#sales_df_with_totalsales.write.partitionBy("Date").mode("append").parquet(output_path)   # incremental load mode

# verifing data
partiond_date_df = spark.read.parquet(output_path)
partiond_date_df.show()
```

```
+-------------+----------+---------+--------+-----+----------+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|      Date|TotalSales|
+-------------+----------+---------+--------+-----+----------+----------+
|            1|       101|      501|       2|150.0|2024-09-01|     300.0|
|            2|       102|      502|       1|250.0|2024-09-01|     250.0|
|            3|       103|      501|       4|150.0|2024-09-02|     600.0|
|            4|       101|      503|       3|300.0|2024-09-02|     900.0|
|            5|       104|      504|       1|450.0|2024-09-03|     450.0|
|            6|       102|      502|       2|250.0|2024-09-03|     500.0|
|            7|       103|      503|       5|300.0|2024-09-04|    1500.0|
|            8|       104|      504|       1|450.0|2024-09-04|     450.0|
|            9|       101|      501|       2|150.0|2024-09-05|     300.0|
|           10|       105|      505|       1|550.0|2024-09-05|     550.0|
+-------------+----------+---------+--------+-----+----------+----------+

+-------------+----------+---------+--------+-----+----------+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|TotalSales|      Date|
+-------------+----------+---------+--------+-----+----------+----------+
|            5|       104|      504|       1|450.0|     450.0|2024-09-03|
|            6|       102|      502|       2|250.0|     500.0|2024-09-03|
|            3|       103|      501|       4|150.0|     600.0|2024-09-02|
|            4|       101|      503|       3|300.0|     900.0|2024-09-02|
|            1|       101|      501|       2|150.0|     300.0|2024-09-01|
|            2|       102|      502|       1|250.0|     250.0|2024-09-01|
|            7|       103|      503|       5|300.0|    1500.0|2024-09-04|
|            8|       104|      504|       1|450.0|     450.0|2024-09-04|
|            9|       101|      501|       2|150.0|     300.0|2024-09-05|
|           10|       105|      505|       1|550.0|     550.0|2024-09-05|
+-------------+----------+---------+--------+-----+----------+----------+
```

Wdigets:

```
[133] from pyspark.sql import SparkSession
      import ipywidgets as widgets
      from IPython.display import display
      from pyspark.sql.functions import col

      # Initialize a Spark session
      spark = SparkSession.builder.appName("Widgets Example").getOrCreate()
```

```
      # Create a simple DataFrame
      data = [
          ("John", 28, "Male", 60000),
          ("Jane", 32, "Female", 72000),
          ("Mike", 45, "Male", 84000),
          ("Emily", 23, "Female", 52000),
          ("Alex", 36, "Male", 67000)
      ]

      df = spark.createDataFrame(data, ["name", "age", "gender", "salary"])

      # Show the DataFrame
      df.show()
```

```
+-----+---+------+------+
| name|age|gender|salary|
+-----+---+------+------+
| John| 28|  Male| 60000|
| Jane| 32|Female| 72000|
| Mike| 45|  Male| 84000|
|Emily| 23|Female| 52000|
| Alex| 36|  Male| 67000|
+-----+---+------+------+
```

```
[145] # widget -> dropdown
      dropdown = widgets.Dropdown(
          options=["age", "salary","both"],
          value="age",                    # default select when displayed
          description="Filter By:"
      )
```

```
      # widget -> slider
      slider1 = widgets.IntSlider(
          value=30,
          min=20,
          max=100,
          step=5,
          description="Age Threshold:",
          continuous_update=False
      )

      # widget -> slider
      slider2 = widgets.IntSlider(
          value=30000,
          min=10000,
          max=300000,
          step=5000,
          description="Salary Threshold:",
          continuous_update=False
      )

      # widget -> button
      button = widgets.Button(description="Apply Filter")

      # output area to show results
      output = widgets.Output()

      # display widgets
      display(dropdown, slider1, slider2, button, output)
```

```python
def apply_filter(b):
    column = dropdown.value
    threshold1 = slider1.value
    threshold2 = slider2.value

    if column == "age":
        filtered_df = df.filter((col(column) >= threshold1))
    elif column == "salary":
        filtered_df = df.filter(col(column) >= threshold2)
    else:
        filtered_df = df.filter((col("age") >= threshold1) & (col("salary") >= threshold2))

    with output:
        output.clear_output() # clear previous output
        print("Filtered DataFrame for:")
        filtered_df.show()

# button click event
button.on_click(apply_filter)
```