Untitled3

June 4, 2025

[4]:
```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, TimestampType
from pyspark.sql.functions import col, to_timestamp
```

[5]:
```
spark = SparkSession.builder\
                    .master("local")\
                    .appName("demo")\
                    .getOrCreate()
```

[6]:
```
schema = StructType([
    StructField("customer_nbr", StringType(), True),
    StructField("customer_desc", StringType(), True),
    StructField("start_ts", StringType(), True),
    StructField("end_ts", StringType(), True),
    StructField("create_ts", StringType(), True),
    StructField("last_update_ts", StringType(), True),
    StructField("client_id", StringType(), True)
])
```

[8]:
```
# Load raw data from CSV with the defined schema
df_raw = spark.read.csv("gs://newbucketrgcp1/customer_data_with_values.
 ↪csv",header = True,schema = schema)
df_raw.show(0)
```

```
+------------+-------------+--------+------+---------+--------------+---------+
|customer_nbr|customer_desc|start_ts|end_ts|create_ts|last_update_ts|client_id|
+------------+-------------+--------+------+---------+--------------+---------+
+------------+-------------+--------+------+---------+--------------+---------+
only showing top 0 rows
```

[9]:
```
df_raw.show()
```

```
[Stage 1:>                                                          (0 + 1) / 1]
```

```
+-----------+-------------+-------------------+-------------------+------------
-------+--------------+-------------------+
|customer_nbr|customer_desc|           start_ts|             end_ts|
create_ts|last_update_ts|          client_id|
+-----------+-------------+-------------------+-------------------+------------
-------+--------------+-------------------+
|     CUST003|   Customer C|2022-12-08 22:59:19|2023-11-28 13:23:03|2021-09-22
12:36:06|         user3|2022-03-09 00:45:34|
|     CUST004|   Customer B|2022-09-23 17:31:40|2023-09-25 15:57:53|2021-03-08
11:52:34|         user4|2022-08-31 17:57:57|
|     CUST002|   Customer D|2022-07-08 22:29:18|2023-02-24 03:16:23|2021-07-16
12:57:40|         user2|2022-06-24 22:13:39|
|     CUST004|   Customer D|2022-12-25 20:41:29|2023-05-24 05:42:58|2021-05-30
19:28:21|         user2|2022-06-10 18:07:29|
|     CUST001|   Customer D|2022-12-29 00:06:48|2023-06-03 16:54:36|2021-11-19
20:38:19|         user4|2022-04-09 05:20:27|
|     CUST004|   Customer A|2022-01-20 01:03:28|2023-04-09 04:26:01|2021-12-01
22:47:42|         user4|2022-07-02 22:05:32|
|     CUST002|   Customer C|2022-05-19 15:15:52|2023-09-04 21:57:20|2021-06-14
01:48:49|         user4|2022-10-06 04:34:19|
|     CUST004|   Customer B|2022-09-18 05:45:43|2023-11-11 05:50:45|2021-11-20
11:08:28|         user2|2022-12-12 11:56:58|
|     CUST001|   Customer B|2022-09-22 16:46:56|2023-12-24 09:50:40|2021-12-12
09:21:34|         user4|2022-05-25 00:34:22|
|     CUST003|   Customer D|2022-11-29 23:45:45|2023-03-17 19:42:50|2021-10-20
17:05:13|         user4|2022-10-30 07:01:22|
|     CUST004|   Customer D|2022-04-21 09:46:54|2023-02-15 05:14:24|2021-01-18
12:26:47|         user1|2022-04-30 16:57:42|
|     CUST004|   Customer B|2022-02-26 17:59:11|2023-04-24 23:30:59|2021-02-04
18:24:10|         user1|2022-04-01 03:32:01|
|     CUST002|   Customer C|2022-07-12 23:21:09|2023-12-27 01:53:30|2021-09-20
21:54:35|         user3|2022-06-27 06:35:12|
|     CUST001|   Customer D|2022-10-02 13:36:42|2023-02-19 02:08:02|2021-12-06
00:09:33|         user2|2022-04-15 04:45:35|
|     CUST003|   Customer D|2022-05-05 23:32:11|2023-08-25 11:08:05|2021-07-09
05:51:34|         user2|2022-01-06 03:56:05|
|     CUST003|   Customer C|2022-04-25 09:21:10|2023-11-12 03:24:28|2021-08-01
06:01:45|         user2|2022-05-30 17:08:02|
|     CUST003|   Customer C|2022-09-10 21:59:57|2023-02-07 05:01:21|2021-04-06
15:46:18|         user3|2022-12-17 06:45:53|
|     CUST001|   Customer C|2022-01-05 05:16:43|2023-12-17 04:01:29|2021-12-26
07:47:14|         user1|2022-12-06 17:04:53|
|     CUST004|   Customer C|2022-04-26 10:00:11|2023-05-04 21:14:12|2021-11-19
12:39:04|         user1|2022-10-04 02:08:58|
|     CUST001|   Customer C|2022-04-29 21:19:24|2023-07-09 14:14:15|2021-09-23
17:39:17|         user1|2022-07-06 04:23:33|
+-----------+-------------+-------------------+-------------------+------------
-------+--------------+-------------------+
only showing top 20 rows
```

[10]:
```
# Transform timestamp columns from string to timestamp
df_transformed = df_raw \
    .withColumn("start_timestamp", to_timestamp(col("start_ts"), "yyyy-MM-dd HH:
↪mm:ss")) \
    .withColumn("end_timestamp", to_timestamp(col("end_ts"), "yyyy-MM-dd HH:mm:
↪ss")) \
    .withColumn("create_timestamp", to_timestamp(col("create_ts"), "yyyy-MM-dd␣
↪HH:mm:ss")) \
    .withColumn("last_update_timestamp", to_timestamp(col("last_update_ts"),␣
↪"yyyy-MM-dd HH:mm:ss")) \
    .drop("start_ts", "end_ts", "create_ts", "last_update_ts")
```

[11]:
```
# Save transformed data to CSV
df_transformed.write.csv("transformed_data_struct1.csv", header=True)#,␣
↪mode="overwrite")

df_transformed.show(5)
```

```
+------------+-------------+-------------------+-------------------+------------
-------+-------------------+---------------------+
|customer_nbr|customer_desc|          client_id|    start_timestamp|
end_timestamp|   create_timestamp|last_update_timestamp|
+------------+-------------+-------------------+-------------------+------------
-------+-------------------+---------------------+
|     CUST003|   Customer C|2022-03-09 00:45:34|2022-12-08 22:59:19|2023-11-28
13:23:03|2021-09-22 12:36:06|                 null|
|     CUST004|   Customer B|2022-08-31 17:57:57|2022-09-23 17:31:40|2023-09-25
15:57:53|2021-03-08 11:52:34|                 null|
|     CUST002|   Customer D|2022-06-24 22:13:39|2022-07-08 22:29:18|2023-02-24
03:16:23|2021-07-16 12:57:40|                 null|
|     CUST004|   Customer D|2022-06-10 18:07:29|2022-12-25 20:41:29|2023-05-24
05:42:58|2021-05-30 19:28:21|                 null|
|     CUST001|   Customer D|2022-04-09 05:20:27|2022-12-29 00:06:48|2023-06-03
16:54:36|2021-11-19 20:38:19|                 null|
+------------+-------------+-------------------+-------------------+------------
-------+-------------------+---------------------+
only showing top 5 rows
```

[13]:
```
df_raw1 = spark.read.csv("gs://newbucketrgcp1/employee_data.csv",header = True)
df_raw1.show(0)
```

```
+-----+-------+---------+---------+----------+
|empId|empName|empGender|empSalary|empCountry|
+-----+-------+---------+---------+----------+
+-----+-------+---------+---------+----------+
only showing top 0 rows
```

[14]:
```
df_raw1.show()
```

```
+-----+---------------+---------+------------------+----------+
|empId|        empName|empGender|         empSalary|empCountry|
+-----+---------------+---------+------------------+----------+
|    1|       John Doe|   Female|40256.214977607815|     India|
|    2|       John Doe|   Female| 54628.04698645289|        UK|
|    3|Michael Johnson|     Male| 92119.45817408481|        UK|
|    4|    Emily Davis|     Male|127470.53530973793|     India|
|    5|       John Doe|     Male| 131842.2807401374|    Canada|
|    6|     Jane Smith|   Female| 66457.25032866534| Australia|
|    7|     Jane Smith|     Male|  97659.7503982054|        UK|
|    8|       John Doe|     Male| 102033.2220579916|    Canada|
|    9|     Jane Smith|     Male| 147863.5614979859|    Canada|
|   10|      Sam Brown|     Male|  135020.178183817|       USA|
|   11|      Sam Brown|   Female| 79012.57794462639|        UK|
|   12|Michael Johnson|     Male| 49611.66080687332|     India|
|   13|    Emily Davis|   Female|141959.93422365707|    Canada|
|   14|    Emily Davis|     Male|115595.09477791714| Australia|
|   15|Michael Johnson|     Male| 39123.11478244402|        UK|
|   16|     Jane Smith|     Male|58254.331529684576|       USA|
|   17|     Jane Smith|     Male|  120467.635616164|     India|
|   18|     Jane Smith|     Male| 75427.58315479774|    Canada|
|   19|     Jane Smith|     Male|126288.83638980243|       USA|
|   20|    Emily Davis|     Male|108338.68524293705|        UK|
+-----+---------------+---------+------------------+----------+
only showing top 20 rows
```

[15]:
```
from pyspark.sql.functions import col, explode, split, count

# Split the empName column into words and explode the result
words_df = df_raw1.select(explode(split(col("empName"), " ")).alias("word"))
```

```
# Group by word and count occurrences, then order by count in descending order␣
↪and limit to top 3
top_words = words_df.groupBy("word").count().orderBy(col("count").desc()).
↪limit(3)
top_words.show()
```

```
[Stage 9:>                                                          (0 + 1) / 1]

+-----+-----+
| word|count|
+-----+-----+
| Jane|   10|
|Smith|   10|
|Davis|    9|
+-----+-----+
```

[16]:
```
# Drop duplicate records from the DataFrame
df_no_duplicates = df_raw1.dropDuplicates()
df_no_duplicates.show()
```

```
[Stage 13:>                                                         (0 + 1) / 1]

+-----+---------------+---------+------------------+----------+
|empId|        empName|empGender|         empSalary|empCountry|
+-----+---------------+---------+------------------+----------+
|   21|    Emily Davis|   Female| 87736.58716258111|     India|
|    8|       John Doe|     Male| 102033.2220579916|    Canada|
|   11|      Sam Brown|   Female| 79012.57794462639|        UK|
|   25|     Jane Smith|     Male|  120191.118217398| Australia|
|   15|Michael Johnson|     Male| 39123.11478244402|        UK|
|   20|    Emily Davis|     Male|108338.68524293705|        UK|
|    5|       John Doe|     Male| 131842.2807401374|    Canada|
|   26|    Emily Davis|   Female|126616.69717793733|    Canada|
|    2|       John Doe|   Female| 54628.04698645289|        UK|
|   39|Michael Johnson|   Female|40328.150028249336|    Canada|
|   29|      Sam Brown|     Male|113824.85958333808|    Canada|
|   17|     Jane Smith|     Male|  120467.635616164|     India|
|   13|    Emily Davis|   Female|141959.93422365707|    Canada|
|   38|      Sam Brown|   Female| 66819.14581953104|     India|
|   30|      Sam Brown|     Male|  42052.6967687742|        UK|
|   24|    Emily Davis|   Female| 63520.52748685201| Australia|
|    4|    Emily Davis|     Male|127470.53530973793|     India|
|   27|      Sam Brown|   Female|46661.455291065846|       USA|
|   37|       John Doe|     Male| 96903.92923049428|        UK|
|   28|      Sam Brown|   Female|104601.34691057618|       USA|
+------+---------------+---------+-----------------+----------+
only showing top 20 rows
```

[24]:
```
# Split the empName column into words and explode the result
words_df = df_raw1.select(explode(split(col("empName"), " ")).alias("word"))

# Group by word and count occurrences
word_count = words_df.groupBy("word").count()
word_count.show()
```

```
[Stage 37:>                                                         (0 + 1) / 1]

+-------+-----+
|   word|count|
+-------+-----+
|  Davis|    9|
|  Smith|   10|
|Michael|    6|
|    Doe|    7|
|   John|    7|
|    Sam|    8|
|  Emily|    9|
|  Brown|    8|
|   Jane|   10|
|Johnson|    6|
+-------+-----+
```

[18]:
```
from pyspark.sql.functions import avg

# Group by empCountry and calculate the average empSalary
avg_salary_by_country = df_raw1.groupBy("empCountry").agg(avg("empSalary").
↪alias("avg_salary"))
avg_salary_by_country.show()
```

```
[Stage 23:>                                                         (0 + 1) / 1]

+----------+-----------------+
|empCountry|       avg_salary|
+----------+-----------------+
|     India|75087.37664313955|
|       USA|91601.79658475956|
|        UK|77487.65026014007|
|    Canada|99339.65505965672|
| Australia|98809.73973473044|
+----------+-----------------+
```

[19]:
```
# Drop rows with any null values
df_no_nulls = df_raw1.dropna()
df_no_nulls.show()

# Fill null values with a specific value
df_filled_nulls = df_raw1.fillna({"empSalary": 50000})
df_filled_nulls.show()
```

```
+-----+---------------+---------+------------------+----------+
|empId|        empName|empGender|         empSalary|empCountry|
+-----+---------------+---------+------------------+----------+
|    1|       John Doe|   Female|40256.214977607815|     India|
|    2|       John Doe|   Female| 54628.04698645289|        UK|
|    3|Michael Johnson|     Male| 92119.45817408481|        UK|
|    4|    Emily Davis|     Male|127470.53530973793|     India|
|    5|       John Doe|     Male| 131842.2807401374|    Canada|
|    6|     Jane Smith|   Female| 66457.25032866534| Australia|
|    7|     Jane Smith|     Male|  97659.7503982054|        UK|
|    8|       John Doe|     Male| 102033.2220579916|    Canada|
|    9|     Jane Smith|     Male| 147863.5614979859|    Canada|
|   10|      Sam Brown|     Male|  135020.178183817|       USA|
|   11|      Sam Brown|   Female| 79012.57794462639|        UK|
|   12|Michael Johnson|     Male| 49611.66080687332|     India|
|   13|    Emily Davis|   Female|141959.93422365707|    Canada|
|   14|    Emily Davis|     Male|115595.09477791714| Australia|
|   15|Michael Johnson|     Male| 39123.11478244402|        UK|
|   16|     Jane Smith|     Male|58254.331529684576|       USA|
|   17|     Jane Smith|     Male|  120467.635616164|     India|
|   18|     Jane Smith|     Male| 75427.58315479774|    Canada|
|   19|     Jane Smith|     Male|126288.83638980243|       USA|
|   20|    Emily Davis|     Male|108338.68524293705|        UK|
+-----+---------------+---------+------------------+----------+
only showing top 20 rows

+-----+---------------+---------+------------------+----------+
|empId|        empName|empGender|         empSalary|empCountry|
+-----+---------------+---------+------------------+----------+
|    1|       John Doe|   Female|40256.214977607815|     India|
|    2|       John Doe|   Female| 54628.04698645289|        UK|
|    3|Michael Johnson|     Male| 92119.45817408481|        UK|
|    4|    Emily Davis|     Male|127470.53530973793|     India|
|    5|       John Doe|     Male| 131842.2807401374|    Canada|
|    6|      Jane Smith|   Female| 66457.25032866534| Australia|
|    7|      Jane Smith|     Male|  97659.7503982054|        UK|
|    8|        John Doe|     Male| 102033.2220579916|    Canada|
|    9|      Jane Smith|     Male| 147863.5614979859|    Canada|
|   10|       Sam Brown|     Male|  135020.178183817|       USA|
|   11|       Sam Brown|   Female| 79012.57794462639|        UK|
|   12|Michael Johnson|     Male| 49611.66080687332|     India|
|   13|     Emily Davis|   Female|141959.93422365707|    Canada|
|   14|     Emily Davis|     Male|115595.09477791714| Australia|
|   15|Michael Johnson|     Male| 39123.11478244402|        UK|
|   16|      Jane Smith|     Male|58254.331529684576|       USA|
|   17|      Jane Smith|     Male|  120467.635616164|     India|
|   18|      Jane Smith|     Male| 75427.58315479774|    Canada|
|   19|      Jane Smith|     Male|126288.83638980243|       USA|
|   20|     Emily Davis|     Male|108338.68524293705|        UK|
+-----+---------------+---------+------------------+----------+
only showing top 20 rows
```

[20]:
```python
from pyspark.sql.functions import countDistinct

# Count distinct values in empCountry column
distinct_countries = df_raw1.select(countDistinct("empCountry").
 ↪alias("distinct_countries"))
distinct_countries.show()
```

```
[Stage 30:>                                                         (0 + 1) / 1]

+------------------+
|distinct_countries|
+------------------+
|                 5|
+------------------+
```

[21]:
```python
from pyspark.sql.functions import col

# Filter records where empSalary > 50,000
filtered_df = df_raw1.filter(col("empSalary") > 50000)
filtered_df.show()
```

```
+-----+---------------+---------+------------------+----------+
|empId|        empName|empGender|         empSalary|empCountry|
+-----+---------------+---------+------------------+----------+
|    2|       John Doe|   Female| 54628.04698645289|        UK|
|    3|Michael Johnson|     Male| 92119.45817408481|        UK|
|    4|    Emily Davis|     Male|127470.53530973793|     India|
|    5|       John Doe|     Male| 131842.2807401374|    Canada|
|    6|     Jane Smith|   Female| 66457.25032866534| Australia|
|    7|     Jane Smith|     Male|  97659.7503982054|        UK|
|    8|       John Doe|     Male| 102033.2220579916|    Canada|
|    9|     Jane Smith|     Male| 147863.5614979859|    Canada|
|   10|      Sam Brown|     Male|  135020.178183817|       USA|
|   11|      Sam Brown|   Female| 79012.57794462639|        UK|
|   13|    Emily Davis|   Female|141959.93422365707|    Canada|
|   14|    Emily Davis|     Male|115595.09477791714| Australia|
|   16|     Jane Smith|     Male|58254.331529684576|       USA|
|   17|     Jane Smith|     Male|  120467.635616164|     India|
|   18|     Jane Smith|     Male| 75427.58315479774|    Canada|
|   19|     Jane Smith|     Male|126288.83638980243|       USA|
|   20|    Emily Davis|     Male|108338.68524293705|        UK|
|   21|    Emily Davis|   Female| 87736.58716258111|     India|
|   23|      Sam Brown|     Male| 91526.81573411886| Australia|
|   24|    Emily Davis|   Female| 63520.52748685201| Australia|
+-----+---------------+---------+------------------+----------+
only showing top 20 rows
```

[26]:

```python
# Read a JSON file and convert it into a DataFrame
json_df = spark.read.json("gs://newbucketrgcp1/sample_json.json")
json_df.show()
```

```
+--------------------+--------+---------+
|     _corrupt_record|duration|     page|
+--------------------+--------+---------+
|                   [|    null|     null|
|                   {|    null|     null|
|    "user_id": "U…|    null|     null|
|    "name": "Alice",|    null|     null|
|       "location": {|    null|     null|
|      "city": "Ne…|    null|     null|
|      "country": …|    null|     null|
|                  },|    null|     null|
|       "sessions": [|    null|     null|
|                   {|    null|     null|
|        "session_…|    null|     null|
|        "start_ti…|    null|     null|
|        "page_vie…|    null|     null|
|                null|      30|    /home|
|                null|     120|/products|
|                   ]|    null|     null|
|                  },|    null|     null|
|                   {|    null|     null|
|        "session_…|    null|     null|
|        "start_ti…|    null|     null|
+--------------------+--------+---------+
only showing top 20 rows
```

[27]:
```python
# Find the second highest salary
second_highest_salary = df_raw1.orderBy(col("empSalary").desc()).
↪select(col("empSalary")).distinct().collect()[1][0]
print(f"The second highest salary is: {second_highest_salary}")
```

```
[Stage 45:>                                                         (0 + 1) / 1]
```

```
The second highest salary is: 91526.81573411886
```

[28]:
```python
# Sample data for the second DataFrame
data2 = [
    (1, "IT"),
    (2, "HR"),
    (3, "Finance"),
    (4, "Marketing"),
    (5, "Sales"),
    (6, "IT"),
    (7, "HR"),
    (8, "Finance"),
    (9, "Marketing"),
    (10, "Sales")
]

# Define the schema for the second DataFrame
schema2 = ["empId", "empDepartment"]

# Create the second DataFrame
df_raw2 = spark.createDataFrame(data2, schema2)

# Join two DataFrames and select specific columns
joined_df = df_raw1.join(df_raw2, df_raw1.empId == df_raw2.empId).
↪select(df_raw1.empId, df_raw1.empName, df_raw2.empDepartment)
joined_df.show()
```

```
[Stage 49:>                                                         (0 + 1) / 1]
```

```
+-----+---------------+-------------+
|empId|       empName|empDepartment|
+-----+--------------+-------------+
|    1|      John Doe|           IT|
|    2|      John Doe|           HR|
|    3|Michael Johnson|      Finance|
|    4|    Emily Davis|    Marketing|
|    5|      John Doe|        Sales|
|    6|    Jane Smith|           IT|
|    7|    Jane Smith|           HR|
|    8|      John Doe|      Finance|
|    9|    Jane Smith|    Marketing|
|   10|     Sam Brown|        Sales|
+-----+--------------+-------------+
```

```
[ ]:
```