Batch Data Processing with Apache Spark

Tolgahan Cepel - Mert Akat

Contents

1. Introduction

SQL Tables Description

- **FactSale:** Sales transactions fact table
- **FactPurchase:** Purchases fact table
- **DimRetailer:** Retailer details dimension table
- **DimCustomer:** Customer details dimension table
- **DimProduct:** Product details dimension table
- **DimRegion:** Region details dimension table
- **DimDate:** Date dimension table
- **DimSupplier:** Supplier details dimension table

2. Importing the libraries

In [1]:
```
from pyspark.sql import SparkSession, functions as F
```

3. Reading the data

In [51]:
```
# Creating new SparkSession instance
spark = SparkSession.builder.appName("PySparkExample").getOrCreate()

# Reading parquet data and assigning to DataFrame variables
df_pur = spark.read.parquet("data/purchase")
df_sal = spark.read.parquet("data/sale")
df_cus = spark.read.parquet("data/customer")
df_ret = spark.read.parquet("data/retailer")
df_pro = spark.read.parquet("data/product")
df_sup = spark.read.parquet("data/supplier")
df_reg = spark.read.parquet("data/region")
df_date = spark.read.parquet("data/date")

# Creating temporary view tables for Spark SQL queries
df_cus.createOrReplaceTempView("DimCustomer")
df_pur.createOrReplaceTempView("FactPurchase")
df_sal.createOrReplaceTempView("FactSale")
df_ret.createOrReplaceTempView("DimRetailer")
df_pro.createOrReplaceTempView("DimProduct")
df_sup.createOrReplaceTempView("DimSupplier")
df_reg.createOrReplaceTempView("DimRegion")
df_date.createOrReplaceTempView("DimDate")
```

4. Spark SQL Practices

Selecting columns

In [3]:
```
spark.sql("SELECT customer_id, name, surname, birth_date FROM DimCustom
```

```
+-----------+-------+--------+----------+
|customer_id|   name| surname|birth_date|
+-----------+-------+--------+----------+
|          1| Jazmin|  Burril|1958-09-22|
|          2| Dalila|   Faers|2000-11-08|
|          3|Wayland|Walework|1976-03-08|
|          4|Amberly|  Haquin|1948-10-08|
|          5|Garrett|   Frear|1957-09-25|
+-----------+-------+--------+----------+
```

In [4]:
```
df_cus.select("customer_id", "name", "surname", "birth_date").show(5)
```

```
+-----------+-------+--------+----------+
|customer_id|   name| surname|birth_date|
+-----------+-------+--------+----------+
|          1| Jazmin|  Burril|1958-09-22|
|          2| Dalila|   Faers|2000-11-08|
|          3|Wayland|Walework|1976-03-08|
|          4|Amberly|  Haquin|1948-10-08|
|          5|Garrett|   Frear|1957-09-25|
+-----------+-------+--------+----------+
only showing top 5 rows
```

Data manipulation: Calculating the ages from date of birth data.

In [5]:
```
spark.sql("""
SELECT
    customer_id
    ,name
    ,surname
    ,YEAR(CURRENT_DATE()) - YEAR(birth_date) AS age
FROM DimCustomer
LIMIT 5
""").show()
```

```
+-----------+-------+--------+---+
|customer_id|   name| surname|age|
+-----------+-------+--------+---+
|          1| Jazmin|  Burril| 66|
|          2| Dalila|   Faers| 24|
|          3|Wayland|Walework| 48|
|          4|Amberly|  Haquin| 76|
|          5|Garrett|   Frear| 67|
+-----------+-------+--------+---+
```

In [6]:
```
(
    df_cus.withColumn("age", F.year(F.current_date()) - F.year("birth_d
    .select("customer_id", "name", "surname", "age")
    .show(5)
)
```

```
+-----------+-------+--------+---+
|customer_id|   name| surname|age|
+-----------+-------+--------+---+
|          1| Jazmin|  Burril| 66|
|          2| Dalila|   Faers| 24|
|          3|Wayland|Walework| 48|
|          4|Amberly|  Haquin| 76|
|          5|Garrett|   Frear| 67|
+-----------+-------+--------+---+
only showing top 5 rows
```

Filtering rows

In [7]:
```
spark.sql("""
SELECT
    name
    ,surname
    ,age
FROM
(
    SELECT
        customer_id
        ,name
        ,surname
        ,YEAR(CURRENT_DATE()) - YEAR(birth_date) AS age
    FROM DimCustomer
)
WHERE age >= 30
LIMIT 5
""").show()
```

```
+-------+--------+---+
|   name| surname|age|
+-------+--------+---+
| Jazmin|  Burril| 66|
|Wayland|Walework| 48|
|Amberly|  Haquin| 76|
|Garrett|   Frear| 67|
|  Horst|   Isted| 49|
+-------+--------+---+
```

In [8]:
```
(
    df_cus.withColumn("age", F.year(F.current_date()) - F.year("birth_d
    .select("name", "surname", "age")
    .filter(F.col("age") >= 30)
    .show(5)
)
```

```
+-------+--------+---+
|   name| surname|age|
+-------+--------+---+
| Jazmin|  Burril| 66|
|Wayland|Walework| 48|
|Amberly|  Haquin| 76|
|Garrett|   Frear| 67|
|  Horst|   Isted| 49|
+-------+--------+---+
only showing top 5 rows
```

Aggregating data

In [9]:
```
spark.sql("""
SELECT
    order_id
    ,SUM(quantity) AS total_quantity
    ,SUM(total_amt) AS total_amount
FROM FactSale
GROUP BY order_id
ORDER BY total_quantity DESC
LIMIT 10
""").show()
```

```
+--------+--------------+------------+
|order_id|total_quantity|total_amount|
+--------+--------------+------------+
|    3647|            13|         521|
|    2574|            13|         488|
|    3515|            13|         402|
|     101|            12|         359|
|     440|            12|         426|
|    3763|            12|         323|
|    1585|            12|         488|
|    3289|            12|         327|
|    1382|            11|         452|
|    1752|            11|         298|
+--------+--------------+------------+
```

In [10]:
```
(
    df_sal.groupBy("order_id").agg(
        F.sum("quantity").alias("total_quantity"),
        F.sum("total_amt").alias("total_amount")
    ).orderBy("total_quantity", ascending=False)
    .show(10)
)
```

```
+--------+--------------+------------+
|order_id|total_quantity|total_amount|
+--------+--------------+------------+
|    3647|            13|         521|
|    2574|            13|         488|
|    3515|            13|         402|
|     101|            12|         359|
|     440|            12|         426|
|    3763|            12|         323|
|    1585|            12|         488|
|    3289|            12|         327|
|    2337|            11|         357|
|    3743|            11|         359|
+--------+--------------+------------+
only showing top 10 rows
```

Joining

In [11]:
```
spark.sql("""
SELECT
    region_name
    ,AVG(YEAR(CURRENT_DATE()) - YEAR(birth_date)) AS age
FROM DimCustomer cus
INNER JOIN DimRegion reg
ON cus.city_id = reg.city_id
GROUP BY region_name
ORDER BY age DESC
""").show()
```

```
+----------------+-----------------+
|     region_name|              age|
+----------------+-----------------+
|          Akdeniz| 50.81521739130435|
|     Dogu Anadolu| 50.13095238095238|
|Guneydogu Anadolu| 48.58119658119658|
|          Marmara|48.189542483660134|
|       Ic Anadolu| 48.07772020725388|
|        Karadeniz| 47.75121951219512|
|             Ege|46.888888888888886|
+----------------+-----------------+
```

In [12]:
```
(
    df_cus
    .join(df_reg, df_cus.city_id == df_reg.city_id)
    .groupBy("region_name").agg(
        F.avg(F.year(F.current_date()) - F.year("birth_date")).alias("a
    )
    .orderBy("age", ascending=False)
    .show()
)
```

```
+----------------+-----------------+
|     region_name|              age|
+----------------+-----------------+
|          Akdeniz| 50.81521739130435|
|     Dogu Anadolu| 50.13095238095238|
|Guneydogu Anadolu| 48.58119658119658|
|          Marmara|48.189542483660134|
|       Ic Anadolu| 48.07772020725388|
|        Karadeniz| 47.75121951219512|
|             Ege|46.888888888888886|
+----------------+-----------------+
```

5. Case Studies

Assignment 1: Jacket sales per region

Write SparkSQL and Python API scripts that results: Region-based total quantity and amount of jacket sales between June and August 2023.

The expected out is as follows:

region_name	product_type	total_quantity	total_amount
Marmara	Jacket	213	8358
Dogu Anadolu	Jacket	284	11547
Guneydogu Anadolu	Jacket	176	6981
Ic Anadolu	Jacket	260	10496
Akdeniz	Jacket	162	6637
Karadeniz	Jacket	310	12582
Ege	Jacket	101	3953

External links

- https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.join.
(https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.join
- https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.Dat
(https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.Da
- https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.grou
(https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.gro
| |

In [25]:

```
# Your Spark SQL Solution:
from pyspark.sql import SparkSession
from pyspark.sql.functions import month, year, sum, lit
# Filtering jacket sales between June and August 2023
jacket_sales = spark.sql("""
    SELECT
        reg.region_name,
        prod.product_type,
        SUM(sal.quantity) AS total_quantity,
        SUM(sal.total_amt) AS total_amount
    FROM
        FactSale sal
    INNER JOIN
        DimProduct prod ON sal.product_id = prod.product_id
    INNER JOIN
        DimCustomer cus ON sal.customer_id = cus.customer_id
    INNER JOIN
        DimRegion reg ON cus.city_id = reg.city_id
    INNER JOIN
        DimDate dat ON sal.date = dat.date
    WHERE
        prod.product_type = 'Jacket'
        AND dat.month BETWEEN 6 AND 8
        AND dat.year = 2023
    GROUP BY
        reg.region_name, prod.product_type

""")

# Displaying the result
jacket_sales.show()

```

```
+----------------+------------+--------------+------------+
|     region_name|product_type|total_quantity|total_amount|
+----------------+------------+--------------+------------+
|         Marmara|      Jacket|           213|        8358|
|    Dogu Anadolu|      Jacket|           284|       11547|
|Guneydogu Anadolu|      Jacket|           176|        6981|
|       Ic Anadolu|      Jacket|           260|       10496|
|          Akdeniz|      Jacket|           162|        6637|
|       Karadeniz|      Jacket|           310|       12582|
|             Ege|      Jacket|           101|        3953|
+----------------+------------+--------------+------------+
```

In [31]:
```
# Your PySpark Solution:


from pyspark.sql import functions as F

# Ceket satışlarını filtreleme ve bölgeye göre gruplama
jacket_sales_per_region = df_sal.join(df_pro, df_sal.product_id == df_p
    .join(df_cus, df_sal.customer_id == df_cus.customer_id) \
    .join(df_reg, df_cus.city_id == df_reg.city_id) \
    .join(df_date, df_sal.date == df_date.date) \
    .filter((df_pro.product_type == 'Jacket') &
            (F.month(df_date.date).between(6, 8)) &
            (df_date.year == 2023)) \
    .groupBy("region_name", "product_type") \
    .agg(F.sum("quantity").alias("total_quantity"),
         F.sum("total_amt").alias("total_amount"))

# Sonucu gösterme
jacket_sales_per_region.show()

```

```
+----------------+------------+--------------+------------+
|     region_name|product_type|total_quantity|total_amount|
+----------------+------------+--------------+------------+
|         Marmara|      Jacket|           213|        8358|
|    Dogu Anadolu|      Jacket|           284|       11547|
|Guneydogu Anadolu|      Jacket|           176|        6981|
|      Ic Anadolu|      Jacket|           260|       10496|
|         Akdeniz|      Jacket|           162|        6637|
|       Karadeniz|      Jacket|           310|       12582|
|             Ege|      Jacket|           101|        3953|
+----------------+------------+--------------+------------+
```

Assignment 2: Maximum turnover of the retailer regions

Find the maximum turnover region of each retailer, and obtain total amount for each retailer and region.

The expected out is as follows:

retailer_id	retailer_name	region_name	total_amount
1	Hepsiburada	Karadeniz	42642
2	Trendyol	Ic Anadolu	71689
3	n11	Ic Anadolu	11995
4	Gittigidiyor	Karadeniz	16081

In [62]:

```
# Your Spark SQL Solution:


max_turnover_per_retailer = spark.sql("""
    SELECT
        retailer_id,
        retailer_name,
        region_name,
        total_amount
    FROM (
        SELECT
            retailer_id,
            retailer_name,
            region_name,
            total_amount,
            ROW_NUMBER() OVER(PARTITION BY retailer_id ORDER BY total_a
        FROM (
            SELECT
                ret.retailer_id,
                ret.retailer_name,
                reg.region_name,
                SUM(sal.total_amt) AS total_amount
            FROM
                FactSale sal
            JOIN
                DimRetailer ret ON sal.retailer_id = ret.retailer_id
            JOIN
                DimCustomer cus ON sal.customer_id = cus.customer_id
            JOIN
                DimRegion reg ON cus.city_id = reg.city_id
            GROUP BY
                ret.retailer_id, ret.retailer_name, reg.region_name
        ) turnover_per_region
    ) ranked_turnover
    WHERE rn = 1
""")

# Displaying the result
max_turnover_per_retailer.show()


```

```
+-----------+-------------+-----------+------------+
|retailer_id|retailer_name|region_name|total_amount|
+-----------+-------------+-----------+------------+
|          1|  Hepsiburada|  Karadeniz|       42642|
|          2|     Trendyol| Ic Anadolu|       71689|
|          3|          n11| Ic Anadolu|       11995|
|          4| Gittigidiyor|  Karadeniz|       16081|
+-----------+-------------+-----------+------------+
```

In [40]:

```
# Your PySpark Solution:
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Joining necessary tables with DataFrame aliases
joined_df = df_sal.alias("sal").join(df_ret.alias("ret"), F.col("sal.re
                  .join(df_cus.alias("cus"), F.col("sal.customer_id") =
                  .join(df_reg.alias("reg"), F.col("cus.city_id") == F.

# Grouping data by retailer_id, region_name, and aggregating total amou
grouped_df = joined_df.groupBy("ret.retailer_id", "ret.retailer_name",
                      .agg(F.sum("total_amt").alias("total_amount"))

# Defining a window partitioned by retailer_id and ordered by total_amo
window_spec = Window.partitionBy("ret.retailer_id").orderBy(F.desc("tot

# Adding a rank column to find the maximum turnover region for each ret
result_df = grouped_df.withColumn("rank", F.rank().over(window_spec)) \
                      .filter(F.col("rank") == 1) \
                      .drop("rank")

# Renaming columns and ordering the result
result_df = result_df.select("ret.retailer_id", "ret.retailer_name", "r
                     .orderBy("ret.retailer_id")

# Showing the result
result_df.show()


```

```
+-----------+-------------+-----------+------------+
|retailer_id|retailer_name|region_name|total_amount|
+-----------+-------------+-----------+------------+
|          1|  Hepsiburada|  Karadeniz|       42642|
|          2|     Trendyol| Ic Anadolu|       71689|
|          3|          n11| Ic Anadolu|       11995|
|          4| Gittigidiyor|  Karadeniz|       16081|
+-----------+-------------+-----------+------------+
```