Market Basket Analysis Using Instacart

Course-end Project 3

DESCRIPTION

Instacart is a grocery delivery and pick-up service that is available in the United States of America and Canada. The company's services can be accessed through a website and a mobile application. The data was collected anonymously and contained a sample of over 3 million grocery orders from over 200,000 Instacart consumers.

The company also provides the week, hour, and day of the order, as well as the time interval between orders to their customers. The tables, are included in this dataset, and a description of each row is provided below:

Dataset description:

1. **Orders CSV**: Consists of 3-4 million rows
2. **Products CSV**: 50 thousand rows
3. **Aisles CSV**: 134 rows
4. **Departments CSV**: 21 rows
5. **order_products__SET**: 30 million rows where SET is defined as:

- **order_products _prior csv**: 3-2 million previous orders
- **order_products_train csv:** 3-2 million order information

Objective:

6. To analyze company data in order to assist businesses in identifying the day when the most orders were placed in order to provide deals for that day
7. To determine which department is responsible for the most product launches

Steps to be performed:

Step 1: Upload the "**insta-cart**" file to the HDFS

- 1.1 Download the relevant dataset from the "**Course Resources**" section or the project description
- 1.2 Upload the dataset to the **"FTP"** lab from your local system
- 1.3 To move the dataset to **"HDFS"** from the "Webconsole" use the put command

I am using Jupyter Notebook

Step 2: Perform the below tasks on the uploaded dataset using PySpark:

- Login to the Pyspark shell
- Explore the orders CSV file and create a DataFrame
    - Read the orders data as a DataFrame in PySpark

  Note: The column "**days_since_prior_order**" may contain NULL values

- Display the data up to 10 rows

```
df = spark.read.csv('instacart/orders.csv',header=True,inferSchema=True)
df.show(10)
df.printSchema()
```

```
+--------+-------+--------+------------+---------+-----------------+---------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+---------------------+
| 2539329|      1|   prior|           1|        2|                8|                 null|
| 2398795|      1|   prior|           2|        3|                7|                 15.0|
|  473747|      1|   prior|           3|        3|               12|                 21.0|
| 2254736|      1|   prior|           4|        4|                7|                 29.0|
|  431534|      1|   prior|           5|        4|               15|                 28.0|
| 3367565|      1|   prior|           6|        2|                7|                 19.0|
|  550135|      1|   prior|           7|        1|                9|                 20.0|
| 3108588|      1|   prior|           8|        1|               14|                 14.0|
| 2295261|      1|   prior|           9|        1|               16|                  0.0|
| 2550362|      1|   prior|          10|        4|                8|                 30.0|
+--------+-------+--------+------------+---------+-----------------+---------------------+
only showing top 10 rows

root
 |-- order_id: integer (nullable = true)
 |-- user_id: integer (nullable = true)
 |-- eval_set: string (nullable = true)
 |-- order_number: integer (nullable = true)
 |-- order_dow: integer (nullable = true)
 |-- order_hour_of_day: integer (nullable = true)
 |-- days_since_prior_order: double (nullable = true)
```

- Replace all null values with a dummy "999" value in the DataFrame that was created in task 1

Replace all null values with a dummy '999' value

```
]: df.count()
```

```
]: 3421083
```

```
]: df.na.drop().count()
```

```
]: 3214874
```

```
]: treated_df = df.na.fill(999,['days_since_prior_order'])
```

```
]: treated_df.createOrReplaceTempView('table')
   spark.sql('select * from table').show(5)
```

```
+--------+-------+--------+------------+---------+-----------------+---------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+---------------------+
| 2539329|      1|   prior|           1|        2|                8|                999.0|
| 2398795|      1|   prior|           2|        3|                7|                 15.0|
|  473747|      1|   prior|           3|        3|               12|                 21.0|
| 2254736|      1|   prior|           4|        4|                7|                 29.0|
|  431534|      1|   prior|           5|        4|               15|                 28.0|
+--------+-------+--------+------------+---------+-----------------+---------------------+
only showing top 5 rows
```

- Examine the orders CSV file and find the busiest day of the week by reading the data as a PySpark DataFrame

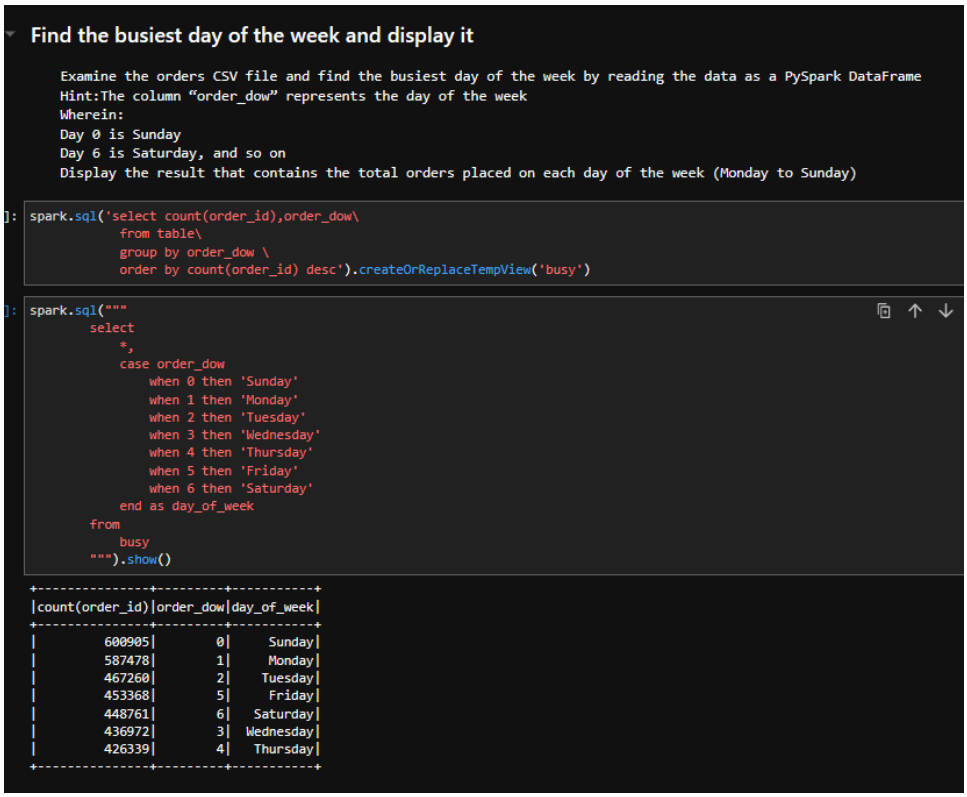
Hint: The column "**order_dow**" represents the day of the week

Wherein:

Day 0 is Sunday

Day 6 is Saturday, and so on

- Display the result that contains the total orders placed on each day of the week (Monday to Sunday)



- Give a breakdown of orders by the hour and identify the busiest hour
    - Select the number of order IDs as “**Total_Orders**” and the hour at which the order was placed
    - Display the result that contains total orders and the hour

Give a breakdown of orders by the hour and identify the busiest hour

Select the number of order IDs as "Total_Orders" and the hour at which the order was placed
Display the result that contains total orders and the hour

```
spark.sql("""
        select
            order_hour_of_day,
            count(order_id) as total_orders
        from
            table
        group by
            order_hour_of_day
        order by
            total_orders desc
        """).show(24)
```

```
+-----------------+------------+
|order_hour_of_day|total_orders|
+-----------------+------------+
|               10|      288418|
|               11|      284728|
|               15|      283639|
|               14|      283042|
|               13|      277999|
|               12|      272841|
|               16|      272553|
|                9|      257812|
|               17|      228795|
|               18|      182912|
|                8|      178201|
|               19|      140569|
|               20|      104292|
|                7|       91868|
|               21|       78109|
|               22|       61468|
|               23|       40043|
|                6|       30529|
|                0|       22758|
|                1|       12398|
|                5|        9569|
|                2|        7539|
|                4|        5527|
|                3|        5474|
+-----------------+------------+
```

- Identify the most popular item based on the order count by exploring order_products__prior and products datasets
    - Calculate the top 10 popular items based on the count of orders
    - Display the result that contains the product name as

"**Popular_product_name**" and the count of order id as "**Order_Count**"

Identify the most popular item based on the order count by exploring order_products__prior and products datasets

Calculate the top 10 popular items based on the count of orders
Display the result that contains the product name as
"Popular_product_name" and the count of order id as "Order_Count"

```
df2 = spark.read.csv('instacart/order_products__prior.csv',header=True,inferSchema=True)
```

```
df2.createOrReplaceTempView('order_products_prior')
```

```
df3 = spark.read.csv('instacart/products.csv',header=True,inferSchema=True)
df3.createOrReplaceTempView('products')
spark.sql('select * from products').show(5)
```

```
+----------+--------------------+--------+-------------+
|product_id|        product_name|aisle_id|department_id|
+----------+--------------------+--------+-------------+
|         1|Chocolate Sandwic...|      61|           19|
|         2|    All-Seasons Salt|     104|           13|
|         3|Robust Golden Uns...|      94|            7|
|         4|Smart Ones Classi...|      38|            1|
|         5|Green Chile Anyti...|       5|           13|
+----------+--------------------+--------+-------------+
only showing top 5 rows
```

```
spark.sql("""
        select
            *
        from
            order_products_prior o
        left join
            products p using (product_id)
        """).createOrReplaceTempView('joined')
```

```
spark.sql("""
        select
            product_name as popular_product_name,
            product_id,
            count(order_id) as order_count
        from
            joined
        group by
            product_name,
            product_id
        order by
            count(order_id) desc
        limit
            10
        """).show()
```

```
+--------------------+----------+-----------+
|popular_product_name|product_id|order_count|
+--------------------+----------+-----------+
|              Banana|     24852|     472565|
|Bag of Organic Ba...|     13176|     379450|
|Organic Strawberries|     21137|     264683|
|Organic Baby Spinach|     21903|     241921|
|Organic Hass Avocado|     47209|     213584|
|     Organic Avocado|     47766|     176815|
|         Large Lemon|     47626|     152657|
|        Strawberries|     16797|     142951|
|               Limes|     26209|     140627|
|  Organic Whole Milk|     27845|     137905|
+--------------------+----------+-----------+
```

- Explore the department dataset and create a DataFrame
- Recognize the department which has published the maximum products
    - Display the department ID that has published the maximum products

Recognize the department which has published the maximum products

Display the department ID that has published the maximum products

```
df4 = spark.read.csv('instacart/departments.csv',header=True,inferSchema=True)
df4.createOrReplaceTempView('dept')
spark.sql('select * from dept').show(5)
```

```
+-------------+----------+
|department_id|department|
+-------------+----------+
|            1|    frozen|
|            2|     other|
|            3|    bakery|
|            4|   produce|
|            5|   alcohol|
+-------------+----------+
only showing top 5 rows
```

```
spark.sql("""
            with tb1 as
                (select
                    *
                from
                    products p
                join
                    dept d using(department_id))
            select
                department_id,
                department,
                count(product_id)
            from
                tb1
            group by
                department_id, department
            order by
                count(product_id) desc
            limit 5
        """).show()
```

```
+-------------+-------------+-----------------+
|department_id|   department|count(product_id)|
+-------------+-------------+-----------------+
|           11|personal care|             6563|
|           19|       snacks|             6264|
|           13|       pantry|             5371|
|            7|    beverages|             4365|
|            1|       frozen|             4007|
+-------------+-------------+-----------------+
```