# Market Basket Analysis Using Instacart

### Objective:

1. 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
2. To determine which department is responsible for the most product launches

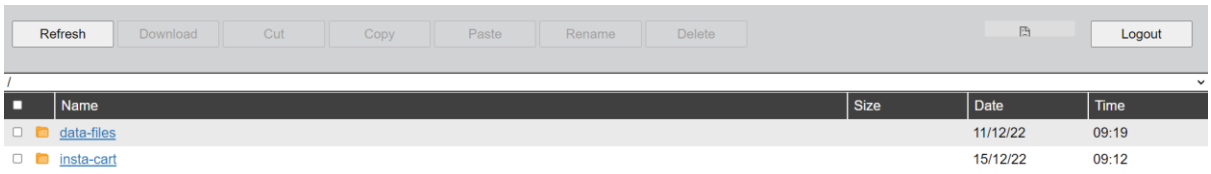
### Steps to be performed:

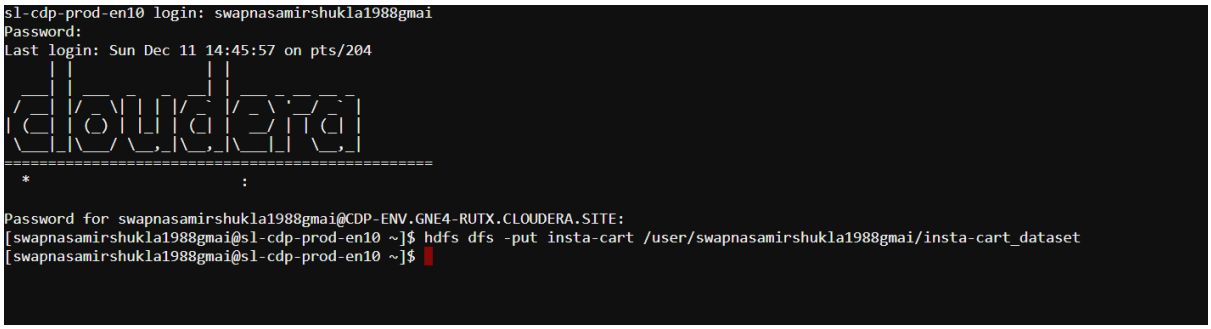
### Step 1: Upload the “insta-cart” file to the HDFS

- **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**

***Commands:***

*hdfs dfs -put insta-cart /user/swapnasamirshukla1988gmai/insta-cart_dataset*




### 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**

***Commands:***

*pyspark3*
*df = spark.read.option("header",True).csv("insta-cart_dataset/orders.csv")*
*df.printSchema()*
*df.show(10)*

```
Spark context available as 'sc' (master = local[*], app id = local-1671105748722).
SparkSession available as 'spark'.
>>> df = spark.read.option("header",True).csv("insta-cart_dataset/orders.csv")
22/12/15 12:03:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/12/15 12:03:34 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
>>> df.printSchema()
root
 |-- order_id: string (nullable = true)
 |-- user_id: string (nullable = true)
 |-- eval_set: string (nullable = true)
 |-- order_number: string (nullable = true)
 |-- order_dow: string (nullable = true)
 |-- order_hour_of_day: string (nullable = true)
 |-- days_since_prior_order: string (nullable = true)

>>> df.show(10)
+--------+-------+--------+------------+---------+-----------------+----------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+----------------------+
| 2539329|      1|   prior|           1|        2|               08|                  null|
| 2398795|      1|   prior|           2|        3|               07|                  15.0|
|  473747|      1|   prior|           3|        3|               12|                  21.0|
| 2254736|      1|   prior|           4|        4|               07|                  29.0|
|  431534|      1|   prior|           5|        4|               15|                  28.0|
| 3367565|      1|   prior|           6|        2|               07|                  19.0|
|  550135|      1|   prior|           7|        1|               09|                  20.0|
| 3108588|      1|   prior|           8|        1|               14|                  14.0|
| 2295261|      1|   prior|           9|        1|               16|                   0.0|
| 2550362|      1|   prior|          10|        4|               08|                  30.0|
+--------+-------+--------+------------+---------+-----------------+----------------------+
only showing top 10 rows
```

***Commands:***
*df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()*

```
>>> from pyspark.sql.functions import isnan, when, count, col
>>> df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
+--------+-------+--------+------------+---------+-----------------+----------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+----------------------+
|       0|      0|       0|           0|        0|                0|                206209|
+--------+-------+--------+------------+---------+-----------------+----------------------+
```

There are almost 206209 null values in the column “**days_since_prior_order”**

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

***Commands:***

*df1 = df.fillna({'days_since_prior_order':'999'})*
*df1.select([count(when(col(c).isNull(), c)).alias(c) for c in df1.columns]).show()*

```
>>> df1 = df.fillna({'days_since_prior_order':'999'})
>>> df1.select([count(when(col(c).isNull(), c)).alias(c) for c in df1.columns]).show()
+--------+-------+--------+------------+---------+-----------------+----------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+----------------------+
|       0|      0|       0|           0|        0|                0|                     0|
+--------+-------+--------+------------+---------+-----------------+----------------------+

>>> df1.show(10)
+--------+-------+--------+------------+---------+-----------------+----------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+----------------------+
| 2539329|      1|   prior|           1|        2|               08|                   999|
| 2398795|      1|   prior|           2|        3|               07|                  15.0|
|  473747|      1|   prior|           3|        3|               12|                  21.0|
| 2254736|      1|   prior|           4|        4|               07|                  29.0|
|  431534|      1|   prior|           5|        4|               15|                  28.0|
| 3367565|      1|   prior|           6|        2|               07|                  19.0|
|  550135|      1|   prior|           7|        1|               09|                  20.0|
| 3108588|      1|   prior|           8|        1|               14|                  14.0|
| 2295261|      1|   prior|           9|        1|               16|                   0.0|
| 2550362|      1|   prior|          10|        4|               08|                  30.0|
+--------+-------+--------+------------+---------+-----------------+----------------------+
only showing top 10 rows
```

All null values have been replaced with 999

**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)**

***Commands:***

```
from pyspark.sql.functions import expr, col

df2 = df1.select(col("*"), expr("CASE WHEN order_dow = '0' THEN 'Sunday' " +
        "WHEN order_dow = '1' THEN 'Monday'" +
                "WHEN order_dow = '2' THEN 'Tuesday'" +
                "WHEN order_dow = '3' THEN 'Wednesday'" +
                "WHEN order_dow = '4' THEN 'Thursday'" +
                "WHEN order_dow = '5' THEN 'Friday'" +
        "ELSE 'Saturday' END").alias("Day of the Week"))
```

```
>>> from pyspark.sql.functions import expr, col
>>> df2 = df1.select(col("*"), expr("CASE WHEN order_dow = '0' THEN 'Sunday' " +
...             "WHEN order_dow = '1' THEN 'Monday'" +
...                 "WHEN order_dow = '2' THEN 'Tuesday'" +
...                 "WHEN order_dow = '3' THEN 'Wednesday'" +
...                 "WHEN order_dow = '4' THEN 'Thursday'" +
...                 "WHEN order_dow = '5' THEN 'Friday'" +
...             "ELSE 'Saturday' END").alias("Day of the Week"))
>>> df2.show(10)
+--------+-------+--------+------------+---------+-----------------+----------------------+---------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|Day of the Week|
+--------+-------+--------+------------+---------+-----------------+----------------------+---------------+
| 2539329|      1|   prior|           1|        2|               08|                   999|        Tuesday|
| 2398795|      1|   prior|           2|        3|               07|                  15.0|      Wednesday|
|  473747|      1|   prior|           3|        3|               12|                  21.0|      Wednesday|
| 2254736|      1|   prior|           4|        4|               07|                  29.0|       Thursday|
|  431534|      1|   prior|           5|        4|               15|                  28.0|       Thursday|
| 3367565|      1|   prior|           6|        2|               07|                  19.0|        Tuesday|
|  550135|      1|   prior|           7|        1|               09|                  20.0|         Monday|
| 3108588|      1|   prior|           8|        1|               14|                  14.0|         Monday|
| 2295261|      1|   prior|           9|        1|               16|                   0.0|         Monday|
| 2550362|      1|   prior|          10|        4|               08|                  30.0|       Thursday|
+--------+-------+--------+------------+---------+-----------------+----------------------+---------------+
only showing top 10 rows
```

```
from pyspark.sql.functions import expr, col, desc
df2.groupBy("Day of the Week").count() \
  .select(col("count").alias("Total Orders"),col("Day of the Week")) \
  .sort(desc("Total Orders")).show(truncate=False)
```

```
>>> from pyspark.sql.functions import sum, col, desc
>>> df2.groupBy("Day of the Week").count() \
...   .select(col("count").alias("Total Orders"),col("Day of the Week")) \
...   .sort(desc("Total Orders")).show(truncate=False)
+------------+---------------+
|Total Orders|Day of the Week|
+------------+---------------+
|600905      |Sunday         |
|587478      |Monday         |
|467260      |Tuesday        |
|453368      |Friday         |
|448761      |Saturday       |
|436972      |Wednesday      |
|426339      |Thursday       |
+------------+---------------+
```

Inference: Sunday is the busiest day of the week

- **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**

***Commands:***

```
df2.groupBy("order_hour_of_day").count() \
  .select(col("count").alias("Total_Orders"),col("order_hour_of_day").alias("Hour")) \
  .sort(desc("Total_Orders")).show(24)
```

```
>>> df2.groupBy("order_hour_of_day").count() \
...   .select(col("count").alias("Total_Orders"),col("order_hour_of_day").alias("Hour")) \
...   .sort(desc("Total_Orders")).show(truncate=False)
+------------+----+
|Total_Orders|Hour|
+------------+----+
|288418      |10  |
|284728      |11  |
|283639      |15  |
|283042      |14  |
|277999      |13  |
|272841      |12  |
|272553      |16  |
|257812      |09  |
|228795      |17  |
|182912      |18  |
|178201      |08  |
|140569      |19  |
|104292      |20  |
|91868       |07  |
|78109       |21  |
|61468       |22  |
|40043       |23  |
|30529       |06  |
|22758       |00  |
|12398       |01  |
+------------+----+
only showing top 20 rows
```

Inference: 10 a.m is the busiest hour of the day

- **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”**

***Commands:***

*prior_df = spark.read.option("header",True).csv("insta-cart_dataset/order_products__prior.csv")*

*product_df = spark.read.option("header",True).csv("insta-cart_dataset/products.csv")*

*prior_df.printSchema()*

*product_df.printSchema()*

*prior_df.join(product_df,prior_df["product_id"] == product_df["product_id"]).show(10)*

*merged_df = prior_df.join(product_df,prior_df["product_id"] == product_df["product_id"])*

```
>>> prior_df = spark.read.option("header",True).csv("insta-cart_dataset/order_products__prior.csv")
>>> product_df = spark.read.option("header",True).csv("insta-cart_dataset/products.csv")
>>> prior_df.printSchema()
root
 |-- order_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- add_to_cart_order: string (nullable = true)
 |-- reordered: string (nullable = true)

>>> product_df.printSchema()
root
 |-- product_id: string (nullable = true)
 |-- product_name: string (nullable = true)
 |-- aisle_id: string (nullable = true)
 |-- department_id: string (nullable = true)

>>> prior_df.join(product_df,prior_df["product_id"] == product_df["product_id"]).show(10)
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
|order_id|product_id|add_to_cart_order|reordered|product_id|        product_name|aisle_id|department_id|
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
|       2|     33120|                1|        1|     33120|  Organic Egg Whites|      86|           16|
|       2|     28985|                2|        1|     28985|Michigan Organic ...|      83|            4|
|       2|      9327|                3|        0|      9327|       Garlic Powder|     104|           13|
|       2|     45918|                4|        1|     45918|      Coconut Butter|      19|           13|
|       2|     30035|                5|        0|     30035|   Natural Sweetener|      17|           13|
|       2|     17794|                6|        1|     17794|             Carrots|      83|            4|
|       2|     40141|                7|        1|     40141|Original Unflavor...|     105|           13|
|       2|      1819|                8|        1|      1819|All Natural No St...|      88|           13|
|       2|     43668|                9|        0|     43668|Classic Blend Col...|     123|            4|
|       3|     33754|                1|        1|     33754|Total 2% with Str...|     120|           16|
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
only showing top 10 rows
```

***Commands:***

```
merged_df.groupBy("product_name").count() \
  .select(col("count").alias("Order_Count"),col("product_name").alias("Popular_product_name")) \
  .sort(desc("Order_Count")).show(10,False)
```

```
>>> merged_df = prior_df.join(product_df,prior_df["product_id"] == product_df["product_id"])
>>> merged_df.show(5)
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
|order_id|product_id|add_to_cart_order|reordered|product_id|        product_name|aisle_id|department_id|
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
|       2|     33120|                1|        1|     33120|  Organic Egg Whites|      86|           16|
|       2|     28985|                2|        1|     28985|Michigan Organic ...|      83|            4|
|       2|      9327|                3|        0|      9327|       Garlic Powder|     104|           13|
|       2|     45918|                4|        1|     45918|      Coconut Butter|      19|           13|
|       2|     30035|                5|        0|     30035|   Natural Sweetener|      17|           13|
+--------+----------+-----------------+---------+----------+--------------------+--------+-------------+
only showing top 5 rows

>>> merged_df.groupBy("product_name").count() \
...   .select(col("count").alias("Order_Count"),col("product_name").alias("Popular_product_name")) \
...   .sort(desc("Order_Count")).show(10,False)
+-----------+---------------------+
|Order_Count|Popular_product_name  |
+-----------+---------------------+
|472565     |Banana               |
|379450     |Bag of Organic Bananas|
|264683     |Organic Strawberries |
|241921     |Organic Baby Spinach |
|213584     |Organic Hass Avocado |
|176815     |Organic Avocado      |
|152657     |Large Lemon          |
|142951     |Strawberries         |
|140627     |Limes                |
|137905     |Organic Whole Milk   |
+-----------+---------------------+
only showing top 10 rows
```

Inference: Banana is the most popular item. The top 10 most popular items are shown

- **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**

***Commands:***

*dept_df = spark.read.option("header",True).csv("insta-cart_dataset/departments.csv")*

*merged_dept_df = product_df.join(dept_df,"department_id")*
*merged_dept_df.show(5)*

*merged_dept_df.groupBy("department_id").count() *
*.select(col("department_id").alias("Department ID"),col("count").alias("Max_Product")) *
*.sort(desc("Max_Product")).show(truncate=False)*

```
>>> dept_df = spark.read.option("header",True).csv("insta-cart_dataset/departments.csv")
>>> merged_dept_df = product_df.join(dept_df,"department_id")
>>> merged_dept_df.show(5)
+-------------+----------+--------------------+--------+----------+
|department_id|product_id|        product_name|aisle_id|department|
+-------------+----------+--------------------+--------+----------+
|           19|         1|Chocolate Sandwic...|      61|    snacks|
|           13|         2|    All-Seasons Salt|     104|    pantry|
|            7|         3|Robust Golden Uns...|      94| beverages|
|            1|         4|Smart Ones Classi...|      38|    frozen|
|           13|         5|Green Chile Anyti...|       5|    pantry|
+-------------+----------+--------------------+--------+----------+
only showing top 5 rows

>>> merged_dept_df.groupBy("department_id").count() \
...   .select(col("department_id").alias("Department ID"),col("count").alias("Max_Product")) \
...   .sort(desc("Max_Product")).show(truncate=False)
+-------------+-----------+
|Department ID|Max_Product|
+-------------+-----------+
|11           |6563       |
|19           |6264       |
|13           |5371       |
|7            |4365       |
|1            |4007       |
|16           |3449       |
|17           |3084       |
|15           |2092       |
|9            |1858       |
|4            |1684       |
|3            |1516       |
|20           |1322       |
|21           |1258       |
|6            |1139       |
|14           |1115       |
|18           |1081       |
|5            |1054       |
|8            |972        |
|12           |907        |
```

Department# 11 has published the maximum number of products

***Commands:***

*dept_df.filter(dept_df.department_id=='11').show()*

```
>>> merged_dept_df.groupBy("department_id").count() \
...    .select(col("department_id").alias("Department ID"),col("count").alias("Max_Product")) \
...    .sort(desc("Max_Product")).show(truncate=False)
+-------------+-----------+
|Department ID|Max_Product|
+-------------+-----------+
|11           |6563       |
|19           |6264       |
|13           |5371       |
|7            |4365       |
|1            |4007       |
|16           |3449       |
|17           |3084       |
|15           |2092       |
|9            |1858       |
|4            |1684       |
|3            |1516       |
|20           |1322       |
|21           |1258       |
|6            |1139       |
|14           |1115       |
|18           |1081       |
|5            |1054       |
|8            |972        |
|12           |907        |
|2            |548        |
+-------------+-----------+
only showing top 20 rows

>>> dept_df.filter(dept_df.department_id=='11').show()
+-------------+-------------+
|department_id|   department|
+-------------+-------------+
|           11|personal care|
+-------------+-------------+
```

Department# 11 is personal care