BASKET ANALYSIS

Import Libraries

In [1]:
```
import findspark
findspark.init()
findspark.find()
```

Out[1]:
```
'C:\\Users\\Lenovo\\anaconda3\\Lib\\site-packages\\pyspark'
```

In [2]:
```
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
import pyarrow as pa
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import pyplot
from datetime import datetime, timedelta
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf
from pyspark.mllib.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline
from pyspark.sql.functions import *
```

In [4]:
```
from pyspark.sql import functions as F
from pyspark.ml.fpm import FPGrowth
```

In [5]:
```
#create Session Without Information
# create the session
conf = pyspark.SparkConf()
# create the context
sc = pyspark.SparkContext.getOrCreate(conf=conf)
sqlcontext = SQLContext(sc)
spark = SparkSession.builder.getOrCreate()
```

In [6]:
```
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")
```

Reading Datasets

In [7]:
```
df1=spark.read.option("header",True).csv("sample assortment ranking 20220628_509_.csv")
```

In [8]:
```
data = df1.select('kassabon_nummer','sitenummer','transactie_datum','transactie_tijd','artikelnumm
```

In [9]:
```
data.show(1)
```

```
+--------------+---------+-------------------+--------------+------------+--------+-----+----
-----+----------------+----+-----+----+--------------+
|kassabon_nummer|sitenummer|    transactie_datum|transactie_tijd|artikelnummer|    merk|omzet|aant
al_ce|scan_marge_waarde|week|maand|jaar|unit_cost_price|
+--------------+---------+-------------------+--------------+------------+--------+-----+----
-----+----------------+----+-----+----+--------------+
|          13031|       792|2022-02-27T00:00:...|        12:9:40|       118001|Red Bull| 2.05|
1|            0.86|   8|    2|2022|        1.0228|
+--------------+---------+-------------------+--------------+------------+--------+-----+----
-----+----------------+----+-----+----+--------------+
only showing top 1 row
```

Market Basket Analysis

In [10]:
```
basketdata = data.dropDuplicates(['kassabon_nummer', 'artikelnummer']).sort('kassabon_nummer')
```

In [11]:
```
basketdata = basketdata.groupBy("kassabon_nummer").agg(F.collect_list("artikelnummer")).sort('kass
```

In [12]:
```
basketdata.show()
```

```
+--------------+--------------------------+
|kassabon_nummer|collect_list(artikelnummer)|
+--------------+--------------------------+
|             10|                  [118001]|
|           1000|          [9156536, 118001]|
|          10006|                  [118001]|
|           1001|          [9156536, 118001]|
|          10025|                  [118001]|
|           1003|                 [2394901]|
|          10047|                  [118001]|
|          10048|                  [118001]|
|           1005|          [9156536, 118001]|
|          10052|                 [2324229]|
|           1006|       [118001, 2324229,...|
|          10060|                 [2324237]|
|           1007|          [118001, 2394901]|
|          10073|                  [118001]|
|          10076|                 [9153240]|
|          10077|                 [2324229]|
|           1008|                  [118001]|
|          10117|                  [118001]|
|           1012|         [9156536, 9153240]|
|           1013|          [2394901, 118001]|
+--------------+--------------------------+
only showing top 20 rows
```

In [13]:
```
#Frequent Pattern Growth – FP Growth is a method of mining frequent itemsets using support, lift,
fpGrowth = FPGrowth(itemsCol="collect_list(artikelnummer)", minSupport=0.006, minConfidence=0.006)
model = fpGrowth.fit(basketdata)
# Display frequent itemsets.
model.freqItemsets.show()
```

```
items = model.freqItemsets
# Display generated association rules.
model.associationRules.show()
rules = model.associationRules
# transform examines the input items against all the association rules and summarize the consequen
model.transform(basketdata).show()
transformed = model.transform(basketdata)
```

```
+------------------+-----+
|             items| freq|
+------------------+-----+
|         [9156536]| 2503|
|[9156536, 2394901]|  194|
| [9156536, 118001]|  832|
|         [2324237]|  979|
| [2324237, 118001]|  227|
|         [2324229]| 1041|
| [2324229, 118001]|  218|
|         [9168049]|  471|
| [9168049, 118001]|  154|
|         [2394901]| 2772|
| [2394901, 118001]|  743|
|         [9153240]| 2385|
|[9153240, 9156536]|  210|
|[9153240, 2394901]|  176|
| [9153240, 118001]|  751|
|         [5507308]|  505|
|          [118001]|13970|
+------------------+-----+
```

```
+----------+----------+--------------------+-------------------+--------------------+
|antecedent|consequent|          confidence|               lift|             support|
+----------+----------+--------------------+-------------------+--------------------+
|  [118001]| [9156536]| 0.05955619183965641| 0.4985778840623893| 0.03970602271642646|
|  [118001]| [2324237]|0.016249105225483177|0.34778728385574514|0.010833253794025006|
|  [118001]| [2324229]| 0.01560486757337151| 0.3141060471973358|0.010403741529063664|
|  [118001]| [9168049]|0.011023622047244094| 0.4904224551548891|0.007349432089338551|
|  [118001]| [2394901]| 0.05318539727988547|0.40203709040502167| 0.03545862365180873|
|  [118001]| [9153240]| 0.05375805297065139| 0.4723045039610186| 0.03584041233177436|
| [9153240]| [9156536]|  0.0880503144654088| 0.7371179741542853|0.010021952849098023|
| [9153240]| [2394901]| 0.07379454926624739| 0.5578250307810056|0.008399350959244059|
| [9153240]|  [118001]|  0.3148846960167715| 0.4723045039610186| 0.03584041233177436|
| [2324229]|  [118001]| 0.20941402497598463| 0.3141060471973358|0.010403741529063664|
| [9156536]| [2394901]| 0.07750699161006792| 0.5858879878056865|0.009258375489166746|
| [9156536]|  [118001]| 0.33240111865761085|0.49857788406238923| 0.03970602271642646|
| [9156536]| [9153240]| 0.08389932081502198| 0.7371179741542854|0.010021952849098023|
| [2324237]|  [118001]| 0.23186925434116445|0.34778728385574514|0.010833253794025006|
| [2394901]| [9156536]| 0.06998556998556998| 0.5858879878056866|0.009258375489166746|
| [2394901]|  [118001]|   0.268037518037518| 0.4020370904050216| 0.03545862365180873|
| [2394901]| [9153240]| 0.06349206349206349| 0.5578250307810056|0.008399350959244059|
| [9168049]|  [118001]| 0.32696390658174096|  0.490422455154889|0.007349432089338551|
+----------+----------+--------------------+-------------------+--------------------+
```

```
+---------------+--------------------------+--------------------+
|kassabon_nummer|collect_list(artikelnummer)|          prediction|
+---------------+--------------------------+--------------------+
|             10|                  [118001]|[9156536, 2324237...|
|           1000|         [9156536, 118001]|[2324237, 2324229...|
|          10006|                  [118001]|[9156536, 2324237...|
|           1001|         [9156536, 118001]|[2324237, 2324229...|
|          10025|                  [118001]|[9156536, 2324237...|
|           1003|                 [2394901]|[9156536, 118001,...|
|          10047|                  [118001]|[9156536, 2324237...|
|          10048|                  [118001]|[9156536, 2324237...|
|           1005|         [9156536, 118001]|[2324237, 2324229...|
|          10052|                 [2324229]|            [118001]|
|           1006|      [118001, 2324229,...]|[2324237, 9168049...|
```

```
|          10060|                  [2324237]|            [118001]|
|           1007|          [118001, 2394901]|[9156536, 2324237...|
|          10073|                   [118001]|[9156536, 2324237...|
|          10076|                  [9153240]|[9156536, 2394901...|
|          10077|                  [2324229]|            [118001]|
|           1008|                   [118001]|[9156536, 2324237...|
|          10117|                   [118001]|[9156536, 2324237...|
|           1012|         [9156536, 9153240]|   [2394901, 118001]|
|           1013|          [2394901, 118001]|[9156536, 2324237...|
+---------------+--------------------------+--------------------+
only showing top 20 rows
```

Analysis 1: Total Number of Baskets for each Product

This shows in how many baskets thus the product appear. through this we can find quantity of different combination the product appear.

In [14]:
```
basket = items.select("items").rdd.flatMap(lambda x: x).collect()
basket =pd.Series(basket)
```

In [15]:
```
prod = data.select("artikelnummer").distinct().select("artikelnummer").rdd.flatMap(lambda x: x).cc
prod = pd.Series(prod)
```

In [16]:
```
def getbasketcount(prod):

    basket_count = 0
    for i in basket:
        if prod in i:
            basket_count = basket_count + 1
    collect_df.append({"artikelnummer" : prod, "Total_basket" : basket_count})
```

In [17]:
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('artikelnummer', StringType(), True),
                     StructField('total_basket', DoubleType(),True)])
dff2 = spark.createDataFrame([], schema)
collect_df = dff2.collect()

prod.apply(getbasketcount)

basket_data = spark.createDataFrame(collect_df)
```

In [18]:
```
basket_data.show()
```

```
+------------+-------------+
|Total_basket|artikelnummer|
+------------+-------------+
|           4|      9156536|
|           2|      2324229|
|           1|      5507308|
|           0|      9161540|
|           4|      9153240|
|           7|       118001|
|           2|      9168049|
|           2|      2324237|
|           4|      2394901|
```

```
+-----------+------------+
```

Analysis 2: Product Categorization Based on Number of Baskets

like more appear , less appear in terms of average

In [19]:
```
avg_appearance = basket_data.select('Total_basket').agg(avg('Total_basket').alias('avg')).collect(
```

In [20]:
```
# basket_data.where(col('artikelnummer')=="9156536").select('Total_basket').collect()[0][0]
```

Out[20]: 4

In [21]:
```
def getoccurancelabel(prod):
    aa = basket_data.where(col('artikelnummer')==prod).select('Total_basket').collect()[0][0]
    if aa < avg_appearance:
        collect_df.append({"artikelnummer" : prod, "basket_occurance" : "less appear"})
    elif aa> avg_appearance:
        collect_df.append({"artikelnummer" : prod, "basket_occurance" : "more appear"})
    else:
        collect_df.append({"artikelnummer" : prod, "basket_occurance" : "average"})
```

In [22]:
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('artikelnummer', StringType(), True),
                     StructField('basket_occurance', DoubleType(),True)])
dff2 = spark.createDataFrame([], schema)
collect_df = dff2.collect()

prod.apply(getoccurancelabel)

occurance_data = spark.createDataFrame(collect_df)
```

In [23]:
```
occurance_data.show()
```

```
+-------------+----------------+
|artikelnummer|basket_occurance|
+-------------+----------------+
|      9156536|     more appear|
|      2324229|     less appear|
|      5507308|     less appear|
|      9161540|     less appear|
|      9153240|     more appear|
|       118001|     more appear|
|      9168049|     less appear|
|      2324237|     less appear|
|      2394901|     more appear|
+-------------+----------------+
```

In [24]:
```
MBA_data=basket_data.join(occurance_data,['artikelnummer'],"leftouter")
```

In [25]:
```
MBA_data.show()
```

```
+-------------+------------+----------------+
|artikelnummer|Total_basket|basket_occurance|
+-------------+------------+----------------+
|       118001|           7|     more appear|
|      2324229|           2|     less appear|
|      2324237|           2|     less appear|
|      2394901|           4|     more appear|
|      5507308|           1|     less appear|
|      9153240|           4|     more appear|
|      9156536|           4|     more appear|
|      9161540|           0|     less appear|
|      9168049|           2|     less appear|
+-------------+------------+----------------+
```

Analysis 3: Identification of basket in terms of Size for each Product

Here we identify the type of basket in which particular product mostly occur. we categorize the basket as small, medium and large, by comparing the size of the basket with the average size.

In [26]:
```
def getbasketlenght(basket):
    collect_df.append({"basket" : basket, "lenght" : len(basket)})
```

In [27]:
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('basket', StringType(), True),
                     StructField('lenght', DoubleType(),True)])
dff2 = spark.createDataFrame([], schema)
collect_df = dff2.collect()

basket.apply(getbasketlenght)

basket_len_data = spark.createDataFrame(collect_df)
```

In [28]:
```
basket_len_data.show()
```

```
+------------------+------+
|            basket|lenght|
+------------------+------+
|         [9156536]|     1|
|[9156536, 2394901]|     2|
| [9156536, 118001]|     2|
|         [2324237]|     1|
| [2324237, 118001]|     2|
|         [2324229]|     1|
| [2324229, 118001]|     2|
|         [9168049]|     1|
| [9168049, 118001]|     2|
|         [2394901]|     1|
| [2394901, 118001]|     2|
|         [9153240]|     1|
|[9153240, 9156536]|     2|
|[9153240, 2394901]|     2|
| [9153240, 118001]|     2|
|         [5507308]|     1|
|          [118001]|     1|
```

```
+------------------+------+
```

To Categorize the basket on the bases of size i have used the criteria given below. these criteria will work well not the every size of data. this criteria uses the maximum size of the basket and find the ranges of each level using percentage.

- small: < max_len * 33.334%
- medium: >= max_len *33.334% and < max_len* 66.668%
- large: >= max_len *66.668%

In [29]:
```
max_basket_len = basket_len_data.agg(max('lenght')).collect()[0][0]
```

In [30]:
```
def label_basket(x):
    if (x < (max_basket_len*0.33334)):
        return "small"
    elif ((x>=(max_basket_len*0.33334)) and (x<(max_basket_len*0.66668))):
        return "medium"
    elif (x>=(max_basket_len*0.66668)):
        return "large"
    return "not run"
```

In [31]:
```
label = udf(lambda q : label_basket(q), StringType())
```

In [32]:
```
cc=basket_len_data.withColumn('basket_label',label(col("lenght")))
```

In [33]:
```
cc.show()
```

```
+------------------+------+------------+
|            basket|lenght|basket_label|
+------------------+------+------------+
|         [9156536]|     1|      medium|
|[9156536, 2394901]|     2|       large|
| [9156536, 118001]|     2|       large|
|         [2324237]|     1|      medium|
| [2324237, 118001]|     2|       large|
|         [2324229]|     1|      medium|
| [2324229, 118001]|     2|       large|
|         [9168049]|     1|      medium|
| [9168049, 118001]|     2|       large|
|         [2394901]|     1|      medium|
| [2394901, 118001]|     2|       large|
|         [9153240]|     1|      medium|
|[9153240, 9156536]|     2|       large|
|[9153240, 2394901]|     2|       large|
| [9153240, 118001]|     2|       large|
|         [5507308]|     1|      medium|
|          [118001]|     1|      medium|
+------------------+------+------------+
```

Since the dataset is very small, so the basket size is also small. that's why the criteria code label size 1 as medium and 2 as large. but when you run this code in your huge data these labels will be more accurate, as they are based on the criteria that works on percentage.

Here we have labelled each basket. now we see, in which types of basket each product mostly occur

In [34]:
```
def getprodbasketlist(prod):
    for i in basket:
        if prod in i:
            prod_basket_list.append({"artikelnummer" : prod, "basket" : i})
```

In [35]:
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('product', StringType(), True),
                     StructField('basket', StringType(),True)
                    ])
dff2 = spark.createDataFrame([], schema)
prod_basket_list = dff2.collect()

prod.apply(getprodbasketlist)

prod_basket_list = spark.createDataFrame(prod_basket_list)
```

In [36]:
```
prod_basket_list.show()
```

```
+-------------+------------------+
|artikelnummer|            basket|
+-------------+------------------+
|      9156536|         [9156536]|
|      9156536|[9156536, 2394901]|
|      9156536| [9156536, 118001]|
|      9156536|[9153240, 9156536]|
|      2324229|         [2324229]|
|      2324229| [2324229, 118001]|
|      5507308|         [5507308]|
|      9153240|         [9153240]|
|      9153240|[9153240, 9156536]|
|      9153240|[9153240, 2394901]|
|      9153240| [9153240, 118001]|
|       118001| [9156536, 118001]|
|       118001| [2324237, 118001]|
|       118001| [2324229, 118001]|
|       118001| [9168049, 118001]|
|       118001| [2394901, 118001]|
|       118001| [9153240, 118001]|
|       118001|          [118001]|
|      9168049|         [9168049]|
|      9168049| [9168049, 118001]|
+-------------+------------------+
only showing top 20 rows
```

In [37]:
```
xx=prod_basket_list.join(cc,['basket'],"leftouter")
```

In [38]:
```
xx = xx.groupBy('artikelnummer','basket_label').count()
```

In [39]:
```
vv = xx.groupBy('artikelnummer').agg(max('count').alias('count'))
vv.show()
```

```
+-------------+-----+
|artikelnummer|count|
+-------------+-----+
|      9156536|    3|
|      2324229|    1|
|      5507308|    1|
```

```
|      9153240|    3|
|       118001|    6|
|      9168049|    1|
|      2324237|    1|
|      2394901|    3|
+-------------+-----+
```

In [40]:
```
ff = vv.join(xx,['artikelnummer','count'],"leftouter")
ff = ff.select('artikelnummer','basket_label')
ff = ff.withColumnRenamed("basket_label","basket_type")
```

In [41]:
```
ff.show()
```

```
+-------------+-----------+
|artikelnummer|basket_type|
+-------------+-----------+
|      9153240|      large|
|      9156536|      large|
|       118001|      large|
|      2324237|     medium|
|      2324237|      large|
|      9168049|     medium|
|      9168049|      large|
|      2324229|      large|
|      2324229|     medium|
|      5507308|     medium|
|      2394901|      large|
+-------------+-----------+
```

In [42]:
```
nn=MBA_data.join(ff,['artikelnummer'],"leftouter")
```

In [43]:
```
basket_data = nn.groupBy('artikelnummer').agg(first('Total_basket').alias('Total_basket'),first('b
```

In [44]:
```
basket_data.show()
```

```
+-------------+------------+----------------+-----------+
|artikelnummer|Total_basket|basket_occurance|basket_type|
+-------------+------------+----------------+-----------+
|       118001|           7|     more appear|      large|
|      2324229|           2|     less appear|      large|
|      2324237|           2|     less appear|     medium|
|      2394901|           4|     more appear|      large|
|      5507308|           1|     less appear|     medium|
|      9153240|           4|     more appear|      large|
|      9156536|           4|     more appear|      large|
|      9161540|           0|     less appear|       null|
|      9168049|           2|     less appear|     medium|
+-------------+------------+----------------+-----------+
```

Analysis 4: Product Relevance

Here to find the product relevance we compare the total turnover and market share of a product "A" with the turnover and market share of other products that are bought with the product "A". so to find the turnover of other products we only take those transactions data in which product "A" is also present.

In [48]:
```
total_turnover = data.select('omzet').agg(sum('omzet')).collect()[0][0]
```

In [51]:
```
prod_turnover = data.groupBy('artikelnummer').agg(sum('omzet').alias('total_prod_turnover'))
```

In [53]:
```
prod_turnover=prod_turnover.withColumn('prod_market_share',(col("total_prod_turnover")/total_turnc
```

In [54]:
```
prod_turnover.show()
```

```
+-------------+------------------+-------------------+
|artikelnummer|total_prod_turnover|  prod_market_share|
+-------------+------------------+-------------------+
|      9156536|  5149.219999999944|  9.106207908357657|
|      2324229| 1847.6499999999983|  3.267501687998802|
|      5507308|  682.4000000000001|  1.206799530154729|
|      9161540| 137.30999999999997|0.24282773078186665|
|      9153240|   5138.82999999994|  9.087833572017809|
|       118001|  37980.49999999786|  67.16713006307019|
|      9168049| 1226.1299999999994|  2.168366219092346|
|      2324237| 1697.7399999999993|  3.002391316419825|
|      2394901| 2686.4799999999605|  4.750941972113171|
+-------------+------------------+-------------------+
```

In [59]:
```
prod_bask_filter_list = prod_basket_list.where(size(col('basket'))>1)
```

In [62]:
```
trans_list = data.groupBy('artikelnummer').agg(collect_list('kassabon_nummer').alias('transaction_
```

In [148…
```
product = prod_bask_filter_list.groupBy('artikelnummer').agg(collect_list('basket')).select('artik
product = pd.Series(product)
```

In [147…
```
def other_prod(prod):
    aa = trans_list.where(col('artikelnummer')==prod).select('transaction_list').collect()[0][0]
    aa = list(dict.fromkeys(aa))
    other_prod_turnover = data.where((col('kassabon_nummer').isin(aa)) & (col('artikelnummer') !=
    aa = prod_bask_filter_list.where(col('artikelnummer')== prod).select('basket')
    bask_len = aa.select('basket').agg(count('basket')).collect()[0][0]

    other_prod = []
    for i in range(0,bask_len):
        other_prod.extend(aa.select('basket').collect()[i][0])
    other_prod_turnover = other_prod_turnover.where(col('artikelnummer').isin(other_prod)).agg(sum
    other_prod_market_share = (other_prod_turnover/total_turnover)*100


    collect_df.append({"artikelnummer" : prod, "other_product_turnover" : other_prod_turnover, "ot
```

In [149…
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('artikelnummer', StringType(), True),
                     StructField('other_product_turnover', DoubleType(),True),
                     StructField('other_product_marketshare', DoubleType(),True)
                    ])
dff2 = spark.createDataFrame([], schema)
collect_df = dff2.collect()
```

```
product.apply(other_prod)

dd = spark.createDataFrame(collect_df)
```

In [150…

```
dd.show()
```

```
+-------------+------------------------+----------------------+
|artikelnummer|other_product_marketshare|other_product_turnover|
+-------------+------------------------+----------------------+
|      9156536|        5.713870377988446|     3230.979999999992|
|      2324229|       1.1485817099133857|     649.4800000000009|
|      9153240|       5.5322668554919785|    3128.2899999999913|
|       118001|          9.3200682061034|     5270.150000000013|
|      9168049|       0.8594732878886261|     486.0000000000002|
|      2324237|       1.1777436739407179|     665.9700000000009|
|      2394901|        5.403823347468626|     3055.659999999993|
+-------------+------------------------+----------------------+
```

In [151…

```
Basket_Data=prod_turnover.join(dd,['artikelnummer'],"leftouter")
```

In [152…

```
Basket_Data.show()
```

```
+-------------+------------------+------------------+------------------------+----------------------
-----+
|artikelnummer|total_prod_turnover|  prod_market_share|other_product_marketshare|other_product_tur
nover|
+-------------+------------------+------------------+------------------------+----------------------
-----+
|       118001|  37980.49999999786|  67.16713006307019|          9.3200682061034|     5270.1500000
00013|
|      2324229| 1847.6499999999983|  3.267501687998802|       1.1485817099133857|     649.48000000
00009|
|      2324237| 1697.7399999999993|  3.002391316419825|       1.1777436739407179|     665.97000000
00009|
|      2394901| 2686.4799999999605|  4.750941972113171|        5.403823347468626|     3055.6599999
99993|
|      5507308|  682.4000000000001|  1.206799530154729|                     null|
null|
|      9153240|   5138.82999999994|  9.087833572017809|       5.5322668554919785|    3128.28999999
99913|
|      9156536|  5149.219999999944|  9.106207908357657|        5.713870377988446|     3230.9799999
99992|
|      9161540| 137.30999999999997|0.24282773078186665|                     null|
null|
|      9168049| 1226.1299999999994|  2.168366219092346|       0.8594732878886261|     486.00000000
00002|
+-------------+------------------+------------------+------------------------+----------------------
-----+
```

In [153…

```
basket_data = basket_data.join(Basket_Data,['artikelnummer'],"leftouter")
```

In [155…

```
basket_data.show()
```

```
+-------------+-----------+---------------+-----------+------------------+------------------+-
-----------------------+----------------------+
|artikelnummer|Total_basket|basket_occurance|basket_type|total_prod_turnover|  prod_market_share|o
ther_product_marketshare|other_product_turnover|
```

```
+------------+-----------+---------------+----------+------------------+------------------+-
----------------------+---------------------+
|      118001|          7|     more appear|     large|  37980.49999999786|  67.16713006307019|
9.3200682061034|    5270.150000000013|
|     2324229|          2|     less appear|     large| 1847.6499999999983|  3.267501687998802|
1.1485817099133857|     649.4800000000009|
|     2324237|          2|     less appear|    medium| 1697.7399999999993|  3.002391316419825|
1.1777436739407179|     665.9700000000009|
|     2394901|          4|     more appear|     large| 2686.4799999999605|  4.750941972113171|
5.403823347468626|    3055.659999999993|
|     5507308|          1|     less appear|    medium|  682.4000000000001|  1.206799530154729|
null|                 null|
|     9153240|          4|     more appear|     large|   5138.82999999994|  9.087833572017809|
5.5322668554919785|    3128.2899999999913|
|     9156536|          4|     more appear|     large|  5149.219999999944|  9.106207908357657|
5.713870377988446|    3230.979999999992|
|     9161540|          0|     less appear|      null| 137.30999999999997|0.24282773078186665|
null|                 null|
|     9168049|          2|     less appear|    medium| 1226.1299999999994|  2.168366219092346|
0.8594732878886261|     486.0000000000002|
+------------+-----------+---------------+----------+------------------+------------------+-
----------------------+---------------------+
```

Analysis 5: Correlation and Percentage of each Product with Mostly bought Product

(here results will be in the forms of table with these columns, product id, most bought with product, correlation, percentage)

In [337…
```
data = data.withColumn("aantal_ce",col("aantal_ce").cast("int"))
```

In [338…
```
data.dtypes
```

Out[338…
```
[('kassabon_nummer', 'string'),
 ('sitenummer', 'string'),
 ('transactie_datum', 'string'),
 ('transactie_tijd', 'string'),
 ('artikelnummer', 'string'),
 ('merk', 'string'),
 ('omzet', 'string'),
 ('aantal_ce', 'int'),
 ('scan_marge_waarde', 'string'),
 ('week', 'string'),
 ('maand', 'string'),
 ('jaar', 'string'),
 ('unit_cost_price', 'string')]
```

In [339…
```
aa = data.groupBy("kassabon_nummer").pivot("artikelnummer").sum("aantal_ce")
```

In [340…
```
aa = aa.na.fill(value=0)
```

In [341…
```
from pyspark.ml.stat import Correlation
from pyspark.ml.linalg import DenseMatrix, Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.functions import *
```

In [342…

```
assembler = VectorAssembler(inputCols=aa.columns[1:],
outputCol="features",handleInvalid='keep')
df = assembler.transform(aa).select("features")

# correlation will be in Dense Matrix
correlation = Correlation.corr(df,"features","pearson").collect()[0][0]

# To convert Dense Matrix into DataFrame
rows = correlation.toArray().tolist()
df = spark.createDataFrame(rows,aa.columns[1:])
```

In [343…

```
b = spark.createDataFrame([(l,) for l in aa.columns[1:]], ['artikelnummer'])

#add 'sequential' index and join both dataframe to get the final result
a = df.withColumn("row_idx", row_number().over(Window.orderBy(monotonically_increasing_id())))
b = b.withColumn("row_idx", row_number().over(Window.orderBy(monotonically_increasing_id())))

final_df = a.join(b, a.row_idx == b.row_idx).\
             drop("row_idx")
final_df.show()
```

```
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|              118001|             2324229|             2324237|             2394901|             5507308|             9153240|             9156536|             9161540|             9168049|artikelnummer|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|                 1.0|-0.14716870861206266| -0.1361981589997906|-0.18804282490427637|-0.09332760143015721|-0.13906270200554333|-0.16080190394529276| -0.0427711968833376|-0.06849733953761666|       118001|
|-0.14716870861206266|                 1.0| 0.07463135662332179|-0.04824101646022188|-0.01229265597765...|-0.03608143321569...|-0.03926484105429...|-0.01061906270035972|-0.01514041437874...|      2324229|
| -0.1361981589997906| 0.07463135662332179|                 1.0|-0.04368600854159483|-0.01154408488762...|-0.04344608145148...|-0.04084616482589...| -0.0041626405527703|-0.01871039995462...|      2324237|
|-0.18804282490427637|-0.04824101646022188|-0.04368600854159483|                 1.0|-0.02178218943761...|-0.04798868646162651|-0.04275425920830...|-0.01596956964574149|-0.02632459352467036|      2394901|
|-0.09332760143015721|-0.01229265597765...|-0.01154408488762...|-0.02178218943761...|                 1.0|-0.02734776318403...|-0.02794746253820679|0.007068879205617159|-0.01500075159963...|      5507308|
|-0.13906270200554333|-0.03608143321569...|-0.04344608145148...|-0.04798868646162651|-0.02734776318403...|                 1.0|-0.02222307518629...|-0.01234413224698...|-0.00716630532659085|      9153240|
|-0.16080190394529276|-0.03926484105429...|-0.04084616482589...|-0.04275425920830...|-0.02794746253820679|-0.02222307518629...|                 1.0|-0.01646771138873251|-0.01160147302946...|      9156536|
| -0.0427711968833376|-0.01061906270035972| -0.0041626405527703|-0.01596956964574149|0.007068879205617159|-0.01234413224698...|-0.01646771138873251|                 1.0|-0.00705077373165...|      9161540|
|-0.06849733953761666|-0.01514041437874...|-0.01871039995462...|-0.02632459352467036|-0.01500075159963...|-0.00716630532659085|-0.01160147302946...|-0.00705077373165...|                 1.0|      9168049|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+-------------+
```

In [350…
```
product = prod_bask_filter_list.groupBy('artikelnummer').agg(collect_list('basket')).select('artik
product = pd.Series(product)
```

In [352…
```
def relationship(prod):
    corr_data = final_df.select(abs(prod),'artikelnummer').sort(abs(col(prod)).desc()).collect()[1
    corr_value = corr_data[0]
    other_prod = corr_data[1]
    numerator = items.where((array_contains(col("items"),other_prod)) & (array_contains(col("items
    denominator = data.where((col('artikelnummer') == other_prod) | (col('artikelnummer') == prod)
    relation_percentage = (numerator/denominator)*100

    collect_df.append({"artikelnummer" : prod, "Mostly_bought_product" : other_prod, "Correlation_
```

In [353…
```
from pyspark.sql.types import StructType,StructField, StringType, DoubleType
schema = StructType([StructField('artikelnummer', StringType(), True),
                     StructField('Mostly_bought_product', StringType(),True),
                     StructField('Correlation_value', DoubleType(),True),
                     StructField('Relationship_in_percent', DoubleType(),True)
                    ])
dff2 = spark.createDataFrame([], schema)
collect_df = dff2.collect()

product.apply(relationship)

relation_table = spark.createDataFrame(collect_df)
```

In [354…
```
relation_table.show()
```

```
+------------------+--------------------+----------------------+-------------+
|  Correlation_value|Mostly_bought_product|Relationship_in_percent|artikelnummer|
+------------------+--------------------+----------------------+-------------+
|0.16080190394529276|              118001|     3.4179607263166547|      9156536|
|0.14716870861206266|              118001|     0.9665262691199291|      2324229|
|0.13906270200554333|              118001|     3.0952479083377984|      9153240|
|0.18804282490427637|             2394901|      2.725505300612597|       118001|
|0.06849733953761666|              118001|     0.7013389197558976|      9168049|
| 0.1361981589997906|              118001|     1.0102808313676621|      2324237|
|0.18804282490427637|              118001|      2.725505300612597|      2394901|
+------------------+--------------------+----------------------+-------------+
```

In [355…
```
complete_basket_data = basket_data.join(relation_table,['artikelnummer'],"leftouter")
```

In [357…
```
complete_basket_data.select('artikelnummer','Total_basket','basket_occurance','basket_type','total
```

```
+-------------+------------+----------------+-----------+------------------+------------------+-
---------------------+-------------------------+---------------------+------------------+--------
---------------+
|artikelnummer|Total_basket|basket_occurance|basket_type|total_prod_turnover|  prod_market_share|o
ther_product_turnover|other_product_marketshare|Mostly_bought_product|  Correlation_value|Relation
ship_in_percent|
+-------------+------------+----------------+-----------+------------------+------------------+-
---------------------+-------------------------+---------------------+------------------+--------
---------------+
|       118001|           7|     more appear|      large|  37980.49999999786|  67.16713006307019|
5270.150000000013|          9.3200682061034|              2394901|0.18804282490427637|      2.7255
05300612597|
```

```
|      2324229|           2|     less appear|      large| 1847.6499999999983|  3.267501687998802|
649.4800000000009|       1.1485817099133857|                118001|0.14716870861206266|     0.96652
62691199291|
|      2324237|           2|     less appear|     medium| 1697.7399999999993|  3.002391316419825|
665.9700000000009|       1.1777436739407179|                118001| 0.1361981589997906|     1.01028
08313676621|
|      2394901|           4|     more appear|      large| 2686.4799999999605|  4.750941972113171|
3055.659999999993|       5.403823347468626|                118001|0.18804282490427637|      2.7255
05300612597|
|      5507308|           1|     less appear|     medium|  682.4000000000001|  1.206799530154729|
null|                     null|                null|               null|                   null|
|      9153240|           4|     more appear|      large|   5138.82999999994|  9.087833572017809|
3128.2899999999913|       5.5322668554919785|                118001|0.13906270200554333|     3.0952
479083377984|
|      9156536|           4|     more appear|      large|  5149.219999999944|  9.106207908357657|
3230.979999999992|       5.713870377988446|                118001|0.16080190394529276|     3.41796
07263166547|
|      9161540|           0|     less appear|       null| 137.30999999999997|0.24282773078186665|
null|                     null|                null|               null|                   null|
|      9168049|           2|     less appear|     medium| 1226.1299999999994|  2.168366219092346|
486.0000000000002|       0.8594732878886261|                118001|0.06849733953761666|     0.70133
89197558976|
+-------------+------------+----------------+-----------+-------------------+-------------------+-
---------------------+------------------------+--------------------+-------------------+--------
---------------+
```

Analysis

- Total_basket column: this shows, in how many baskets a particlur product appear
- basket_occurance column: this column categorize the product based on the total number of basket they appear. it uses average as a criteria measure
- basket_type Column: this indicate that type of basket which a particular product mostly occur. the Categorization of the basket is done using the largest basket size.
- total_prod_turnover Column: this column indicate the total turnover each particular product generates
- prod_market_share column: this column shows the market share of each particular product
- other_product_marketshare column: this column shows the total market share of all products that are bought with each particular product in a basket. for the calculation of this column only those transactions are used in which other basket products and a particular product are bought together
- other_product_turnover column: this columns indicate the total turnover, which is generated by the other products when a particular product is bought. for the calculation of this column only those transactions are used in which other basket products and a particular product are bought together
- Mostly_bought_product column: this column show the product which is mostly bought with a particular product
- Correlation_value columns: this column shows the relationship value in terms of correlation. the closer the value of correlation to one, the stronger the relationship between them.
- Relationship_in_percent column:this columns indicate the relationship between mostly bought product with a particular product in terms of percentage.