databricksProject_ANALYSE_2

(https://databricks.com)

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
from pyspark.sql.functions import *
```

DATA PREPARATION

```
data_detail = spark.read.load('/FileStore/tables/Details-3.csv', sep=',',format='csv',header="True")
```

```
data_detail.show(5)
```

```
+--------+------+------+--------+-----------+----------------+-----------+
|Order ID|Amount|Profit|Quantity|   Category|    Sub-Category|PaymentMode|
+--------+------+------+--------+-----------+----------------+-----------+
| B-25681|  1096|   658|       7|Electronics|Electronic Games|        COD|
| B-26055|  5729|    64|      14|  Furniture|          Chairs|        EMI|
| B-25955|  2927|   146|       8|  Furniture|       Bookcases|        EMI|
| B-26093|  2847|   712|       8|Electronics|        Printers|Credit Card|
| B-25602|  2617|  1151|       4|Electronics|          Phones|Credit Card|
+--------+------+------+--------+-----------+----------------+-----------+
only showing top 5 rows
```

```
data_detail.printSchema()
```

```
root
 |-- Order ID: string (nullable = true)
 |-- Amount: string (nullable = true)
 |-- Profit: string (nullable = true)
 |-- Quantity: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Sub-Category: string (nullable = true)
 |-- PaymentMode: string (nullable = true)
```

Data Describe

```
data_detail.describe().toPandas()
```

	summary	Order ID	Amount	Profit	Quantity	Category	Sub-Category	PaymentMode
0	count	1500	1500	1500	1500	1500	1500	1500
1	mean	None	291.8473333333333	24.642	3.743333333333333	None	None	None
2	stddev	None	461.92462039764547	168.5588103631313	2.184941998447543	None	None	None
3	min	B-25601	10	-1	1	Clothing	Accessories	COD
4	max	B-26100	99	99	9	Furniture	Trousers	UPI

valeur manquantes

```
data_detail.select([count(when(isnan(c), c)).alias(c) for c in data_detail.columns]).show()
```

```
+--------+------+------+--------+--------+------------+-----------+
|Order ID|Amount|Profit|Quantity|Category|Sub-Category|PaymentMode|
+--------+------+------+--------+--------+------------+-----------+
|       0|     0|     0|       0|       0|           0|          0|
+--------+------+------+--------+--------+------------+-----------+
```

Number of unique values

```
[(col_name, data_detail.agg(approx_count_distinct(col(col_name))).collect()[0][0]) for col_name in
data_detail.columns]
```

```
Out[93]: [('Order ID', 510),
 ('Amount', 574),
 ('Profit', 421),
 ('Quantity', 14),
 ('Category', 3),
 ('Sub-Category', 17),
 ('PaymentMode', 5)]
```

Duplication

```
duplicates_count = data_detail.groupBy(data_detail.columns).count()
total_duplicates= duplicates_count.filter(col("count") > 1).agg({"count": "sum"}).collect()[0][0]
print(total_duplicates)
```

```
None
```

Unique Values for each Category

```
colonnes = ["Quantity","Category", "Sub-Category", "PaymentMode"]

# Afficher le nom de chaque colonne suivi de ses valeurs uniques
for colonne in colonnes:
    print(f"Valeurs uniques pour la colonne '{colonne}':")
    unique_values = data_detail.select(col(colonne)).distinct().rdd.map(lambda row: row[0]).collect()
    for valeur in unique_values:
        print(valeur)
    print()
```

```
Valeurs uniques pour la colonne 'Quantity':
7
11
3
8
5
6
9
1
10
4
12
13
14
2

Valeurs uniques pour la colonne 'Category':
Electronics
Clothing
Furniture
```

Convert Types

```
data_detail = data_detail.withColumn("Amount", col("Amount").cast("integer")) \
       .withColumn("Profit", col("Profit").cast("integer")) \
       .withColumn("Quantity", col("Quantity").cast("integer"))
```

```
data_detail.printSchema()
```

```
root
 |-- Order ID: string (nullable = true)
 |-- Amount: integer (nullable = true)
 |-- Profit: integer (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- Category: string (nullable = true)
 |-- Sub-Category: string (nullable = true)
 |-- PaymentMode: string (nullable = true)
```

```
data_order = spark.read.load('/FileStore/tables/Orders-4.csv', sep=',',format='csv',header="True")
```

```
data_order.show(5)
```

```
+--------+----------+------------+--------------+-------+
|Order ID|Order Date|CustomerName|         State|   City|
+--------+----------+------------+--------------+-------+
| B-26055|10-03-2018|   Harivansh| Uttar Pradesh|Mathura|
| B-25993|03-02-2018|      Madhav|         Delhi|  Delhi|
| B-25973|24-01-2018| Madan Mohan| Uttar Pradesh|Mathura|
| B-25923|27-12-2018|       Gopal|   Maharashtra| Mumbai|
| B-25757|21-08-2018|    Vishakha|Madhya Pradesh| Indore|
+--------+----------+------------+--------------+-------+
only showing top 5 rows
```

```
data_order.printSchema()
```

```
root
 |-- Order ID: string (nullable = true)
 |-- Order Date: string (nullable = true)
 |-- CustomerName: string (nullable = true)
 |-- State: string (nullable = true)
 |-- City: string (nullable = true)
```

```
data_order.select([count(when(isnan(c), c)).alias(c) for c in data_order.columns]).show()
```

```
+--------+----------+------------+-----+----+
|Order ID|Order Date|CustomerName|State|City|
+--------+----------+------------+-----+----+
|       0|         0|           0|    0|   0|
+--------+----------+------------+-----+----+
```

```
duplicates_count = data_order.groupBy(data_order.columns).count()
total_duplicates= duplicates_count.filter(col("count") > 1).agg({"count": "sum"}).collect()[0][0]
print(total_duplicates)
```

```
None
```

```
[(col_name, data_order.agg(approx_count_distinct(col(col_name))).collect()[0][0]) for col_name in
data_order.columns]
```

```
Out[103]: [('Order ID', 510),
 ('Order Date', 311),
 ('CustomerName', 320),
 ('State', 18),
 ('City', 22)]
```

```
data_order = data_order.withColumn("Order Date", to_date("Order Date", "dd-MM-yyyy"))
```

```
data_order.printSchema()
```

```
root
 |-- Order ID: string (nullable = true)
 |-- Order Date: date (nullable = true)
 |-- CustomerName: string (nullable = true)
 |-- State: string (nullable = true)
 |-- City: string (nullable = true)
```

Data Analysis

```
data_detail.createOrReplaceTempView("detail")
data_order.createOrReplaceTempView("order")
```

Top Category Vendu

```
%sql SELECT Category, sum(Quantity) AS TotalQuantity FROM detail
group by Category
order by TotalQuantity desc
```

Table

	Category	TotalQuantity
1	Clothing	3516
2	Electronics	1154
3	Furniture	945

3 rows

Top Methode payement utilisé

```
%sql SELECT PaymentMode,count(PaymentMode) AS Numberpayment FROM detail
group by PaymentMode
order by Numberpayment desc
```

Table

	PaymentMode	Numberpayment
1	COD	684
2	UPI	331
3	Debit Card	202
4	Credit Card	163
5	EMI	120

5 rows

Top Category Amount

```
%sql SELECT Category,sum(Amount) AS topamount FROM detail
group by Category
order by topamount desc
```

Table

	Category	topamount

1	Electronics	166267
2	Clothing	144323
3	Furniture	127181

3 rows

top profit

```
%sql SELECT Category, sum(Profit)  As topprofit FROM detail
group by Category
order by topprofit desc
```

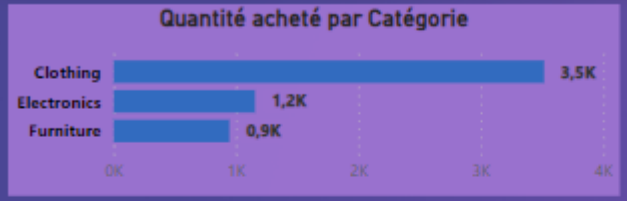
Table

	Category	topprofit
1	Clothing	13325
2	Electronics	13162
3	Furniture	10476

3 rows

Sauvegarder les modifications

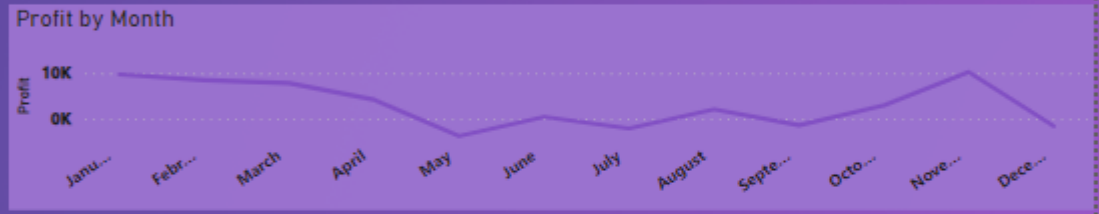
Sales Dashboard

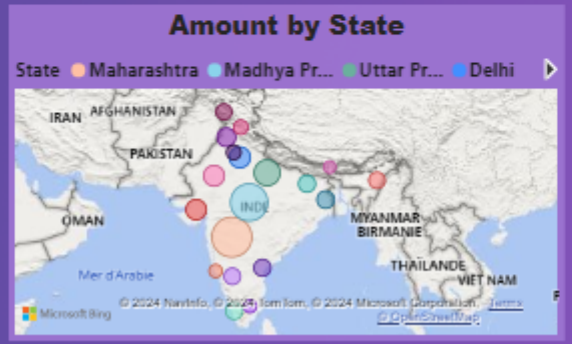


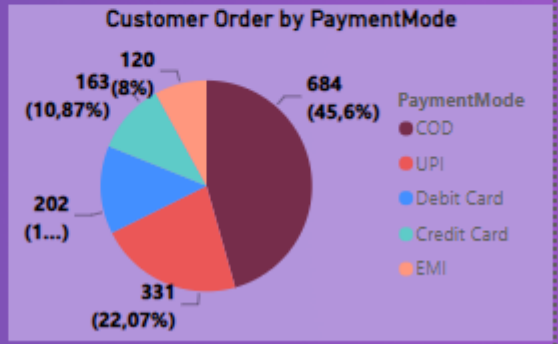
438K
Amount

37K
Profit

5615
Quantity







Filters

Page 1