databricks storeanalysis

(https://databricks.com)

```
#creating the mount point
dbutils.fs.mount(
  source = "wasbs://input@yourstorageservername.blob.core.windows.net",
  mount_point = "/mnt/store",
  extra_configs =
{"fs.azure.account.key.yourstorageservername.blob.core.windows.net":dbutils.sec
rets.get(scope = "databricks-resource-scope", key = "keysecretstorage")})

True
```

```
#reading the parquet file along with schema definition
from pyspark.sql.types import StructType, StructField, StringType, IntegerType,
DateType, DoubleType

customerschema=StructType([
StructField("Customer_Id", StringType(), True),
StructField("Customer_name", StringType(), True),
StructField("Segment", StringType(), True),
StructField("Country", StringType(), True),
StructField("City", StringType(), True),
StructField("State", StringType(), True),
StructField("Postal_code", StringType(), True),
StructField("Region", StringType(), True),
])

orderschema=StructType([
StructField("Order_Id", StringType(), True),
StructField("Order_date", DateType(), False),
StructField("Customer_Id", StringType(), True),
StructField("Product_Id", StringType(), True),
])

productschema=StructType([
StructField("Product_Id", StringType(), True),
StructField("Category", StringType(), True),
StructField("Sub_category", StringType(), True),
StructField("Product_name", StringType(), True),
])

saleschema=StructType([
StructField("Order_Id", StringType(), True),
StructField("Sales", DoubleType(), True),
StructField("Quantity", DoubleType(), True),
StructField("Discount", DoubleType(), True),
StructField("Profit", DoubleType(), True),
])

shipschema=StructType([
StructField("Order_Id", StringType(), True),
StructField("Ship_date", DateType(), True),
StructField("Ship_mode", StringType(), True),
])
dbutils.fs.ls("/mnt/store/")
dateFormat = "MM/dd/yyyy"
```

```
customers = spark.read.parquet("/mnt/store/customer",
header=True,schema=customerschema)
products=spark.read.parquet("/mnt/store/product",
header=True,schema=productschema)
sales=spark.read.parquet("/mnt/store/sales", header=True,schema=saleschema)
ships=spark.read.parquet("/mnt/store/ships", header=True,schema=shipschema)
orders = spark.read.parquet("/mnt/store/orders",
header=True,schema=orderschema)

print(orders.printSchema())
print(customers.printSchema())
print(products.printSchema())
print(sales.printSchema())
print(ships.printSchema())
display(orders)
orders.createOrReplaceTempView("orders")
```

```
root
 |-- Order_Id: string (nullable = true)
 |-- Order_date: string (nullable = true)
 |-- Customer_Id: string (nullable = true)
 |-- Product_Id: string (nullable = true)

None
root
 |-- Customer_Id: string (nullable = true)
 |-- Customer_name: string (nullable = true)
 |-- Segment: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Postal_code: string (nullable = true)
 |-- Region: string (nullable = true)

None
root
 |-- Product_Id: string (nullable = true)
 |-- Category: string (nullable = true)
```

Table

	Order_Id	Order_date	Customer_Id	Product_Id
1	null	null	null	null
2	CA-2016-152156	2016-11-08	CG-12520	FUR-BO-10001798
3	CA-2016-152156	2016-11-08	CG-12520	FUR-CH-10000454
4	CA-2016-138688	2016-06-12	DV-13045	OFF-LA-10000240

5	US-2015-108966	2015-10-11	SO-20335	FUR-TA-10000577
6	US-2015-108966	2015-10-11	SO-20335	OFF-ST-10000760
7	CA-2014-115812	2014-06-09	BH-11710	FUR-FU-10001487

10,000 rows | Truncated data

```
#checking the count before data cleaning
print(orders.count())
print(products.count())
print(ships.count())
print(sales.count())
print(customers.count())
```

```
10001
10001
10001
10001
10001
```

```
#removing the duplicates
orders=orders.dropDuplicates()
products=products.dropDuplicates()
ships=ships.dropDuplicates()
sales=sales.dropDuplicates()
customers=customers.dropDuplicates()
```

```
#checking the count after data cleaning
print(orders.count())
print(products.count())
print(ships.count())
print(sales.count())
print(customers.count())
```

```
9988
1896
5011
9996
4911
```

```
#removing rows with null values for all the columns
orders=orders.dropna(how='all')
products=products.dropna(how='all')
ships=ships.dropna(how='all')
sales=sales.dropna(how='all')
customers=customers.dropna(how='all')


#checking the count after removing rows with null values for all the columns
print(orders.count())
print(products.count())
print(ships.count())
print(sales.count())
print(customers.count())
```

```
9987
1895
5010
9995
4910
```

```
#check if any of the columns is null in orders is null and drop them

from pyspark.sql.functions import col
print(orders.count())
condition = ~col("Order_Id").isNull() & ~col("Order_date").isNull() &
~col("Customer_Id").isNull() & ~col("Product_Id").isNull()
# Apply the filter condition
orders = orders.filter(condition)
print(orders.count())
#orders.createOrReplaceTempView("orders")
#spark.sql("select * from orders where Order_Id IS NULL").show()
# Show the DataFrame after filling null values and filtering
#print(filtered_orders.count())
```

```
9987
9986
```

```
#check if any of the columns in products is null and drop them using dropna()
print(products.count())
products=products.dropna()
print(products.count())
```

```
1895
1894

#check if any of the columns in ship is null and drop them using dropna()
print(ships.count())
ship=ships.dropna()
print(ships.count())


5010
5010


#drop if customers_Id is null
customers=customers.na.drop(subset=['Customer_Id'])
#fill "not available" if region of customers is null
customers=customers.fillna("not available", subset=["region"])
customers.filter(col("region")=="not available").show()


+-----------+-------------+--------+-------------+----+-----+-----------+------
-------+
|Customer_Id|Customer_name| Segment|      Country|City|State|Postal_code|
Region|
+-----------+-------------+--------+-------------+----+-----+-----------+------
-------+
|   EB-13870|  Emily Burns|Consumer|United States|Orem| Utah|      84057|not av
ailable|
+-----------+-------------+--------+-------------+----+-----+-----------+------
-------+



#creating the tables
orders.createOrReplaceTempView("orders_t")
ships.createOrReplaceTempView("ships_t")
products.createOrReplaceTempView("products_t")
sales.createOrReplaceTempView("sales_t")
customers.createOrReplaceTempView("customers_t")
```

```
#testing the dataset from table
spark.sql("select * from orders_t ").show()
spark.sql("select * from ships_t ").show()
spark.sql("select * from products_t ").show()
spark.sql("select * from sales_t ").show()
spark.sql("select * from customers_t ").show()
#Verifying the count of dataset from table
spark.sql("select count(*) as orders_count from orders_t ").show()
spark.sql("select count(*) as ship_count from ships_t ").show()
spark.sql("select count(*) as products_count from products_t ").show()
spark.sql("select count(*) as sales_count from sales_t ").show()
spark.sql("select count(*) as customers_count from customers_t ").show()
```

```
+--------------+----------+-----------+---------------+
|      Order_Id|Order_date|Customer_Id|     Product_Id|
+--------------+----------+-----------+---------------+
|US-2015-134026|2015-04-26|   JE-15745|FUR-CH-10000513|
|CA-2015-119697|2015-12-28|   EM-13960|TEC-AC-10003657|
|CA-2017-153339|2017-11-03|   DJ-13510|FUR-FU-10001967|
|CA-2017-144904|2017-09-25|   KW-16435|FUR-FU-10000023|
|CA-2016-155516|2016-10-21|   MK-17905|OFF-ST-10002406|
|CA-2017-127432|2017-01-22|   AD-10180|OFF-ST-10004507|
|CA-2017-117947|2017-08-18|   NG-18355|FUR-FU-10000521|
|CA-2017-135279|2017-04-09|   BS-11800|OFF-PA-10001281|
|CA-2014-117639|2014-05-21|   MW-18235|OFF-BI-10003925|
|CA-2015-133025|2015-09-17|   MO-17800|OFF-PA-10004100|
|CA-2017-117212|2017-02-26|   BT-11530|FUR-CH-10003973|
|CA-2016-144344|2016-10-28|   PG-18820|OFF-BI-10003719|
|CA-2017-113481|2017-01-02|   AS-10045|OFF-BI-10003694|
|CA-2014-126032|2014-06-23|   BS-11665|TEC-AC-10000158|
|CA-2016-101378|2016-07-14|   RH-19600|TEC-AC-10002345|
|CA-2017-129378|2017-10-01|   NS-18505|OFF-BI-10001922|
|US-2015-147739|2015-12-25|   JD-16150|FUR-FU-10001468|
|CA-2016-169922|2016-06-11|   MZ-17515|FUR-FU-10004415|
```

```
#setting azure sql connection
azure_sql_properties = {
    "url":
"jdbc:sqlserver://yourstorageserver.database.windows.net:1433;databaseName=your
dbname",
    "user": "username",
    "password": "password",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
}
```

```
#no of orders placed according to every year and state
orders_per_state=spark.sql('Select YEAR(o.Order_date) AS
year,state,count(Order_Id) as count \
from orders_t o \
inner join customers_t c \
on o.Customer_Id = c.Customer_Id \
group by c.state, year \
order by year ' )
orders_per_state.show()
print(orders.printSchema())

#writing the data to AzureSQL
table_name = "orders_per_state"
orders_per_state.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
```

```
+----+--------------+-----+
|year|         state|count|
+----+--------------+-----+
|2014|      Virginia|  356|
|2014|       Vermont|   15|
|2014| West Virginia|    4|
|2014|    New Jersey|  181|
|2014|North Carolina|  375|
|2014|      Arkansas|   98|
|2014|        Oregon|  201|
|2014|    New Mexico|   66|
|2014|      Oklahoma|   97|
|2014|     Wisconsin|  144|
|2014|     Minnesota|  150|
|2014|     Tennessee|  243|
|2014|South Carolina|   85|
|2014|      Michigan|  326|
|2014|      Missouri|   66|
|2014|       Georgia|  235|
|2014|      Kentucky|  164|
|2014|          Utah|   71|
```

```
#orders placed by customer till now
orders_per_customer = spark.sql(' Select c.customer_id, year(o.Order_date) as
year, count(o.Order_Id) as count \
from orders_t o \
inner join customers_t c \
on o.Customer_Id = c.Customer_Id \
group by c.customer_id, year \
order by year,count desc ' )
orders_per_customer.show()

#writing the data to AzureSQL
table_name = "orders_per_customer"
orders_per_customer.write.jdbc(url=azure_sql_properties["url"],
table=table_name, mode="overwrite", properties=azure_sql_properties)
```

```
+-----------+----+-----+
|customer_id|year|count|
+-----------+----+-----+
|   RP-19390|2014|  168|
|   AP-10915|2014|  126|
|   EA-14035|2014|  117|
|   XP-21865|2014|  110|
|   MA-17560|2014|  110|
|   CS-12250|2014|  110|
|   SD-20485|2014|  108|
|   RD-19900|2014|   99|
|   ML-17395|2014|   99|
|   BF-11170|2014|   99|
|   SC-20095|2014|   99|
|   SM-20950|2014|   96|
|   JK-15730|2014|   90|
|   JD-15895|2014|   90|
|   WB-21850|2014|   88|
|   CK-12205|2014|   88|
|   JL-15835|2014|   88|
|   BP-11095|2014|   88|
```

```
spark.sql('select * from products_t p').show()
```

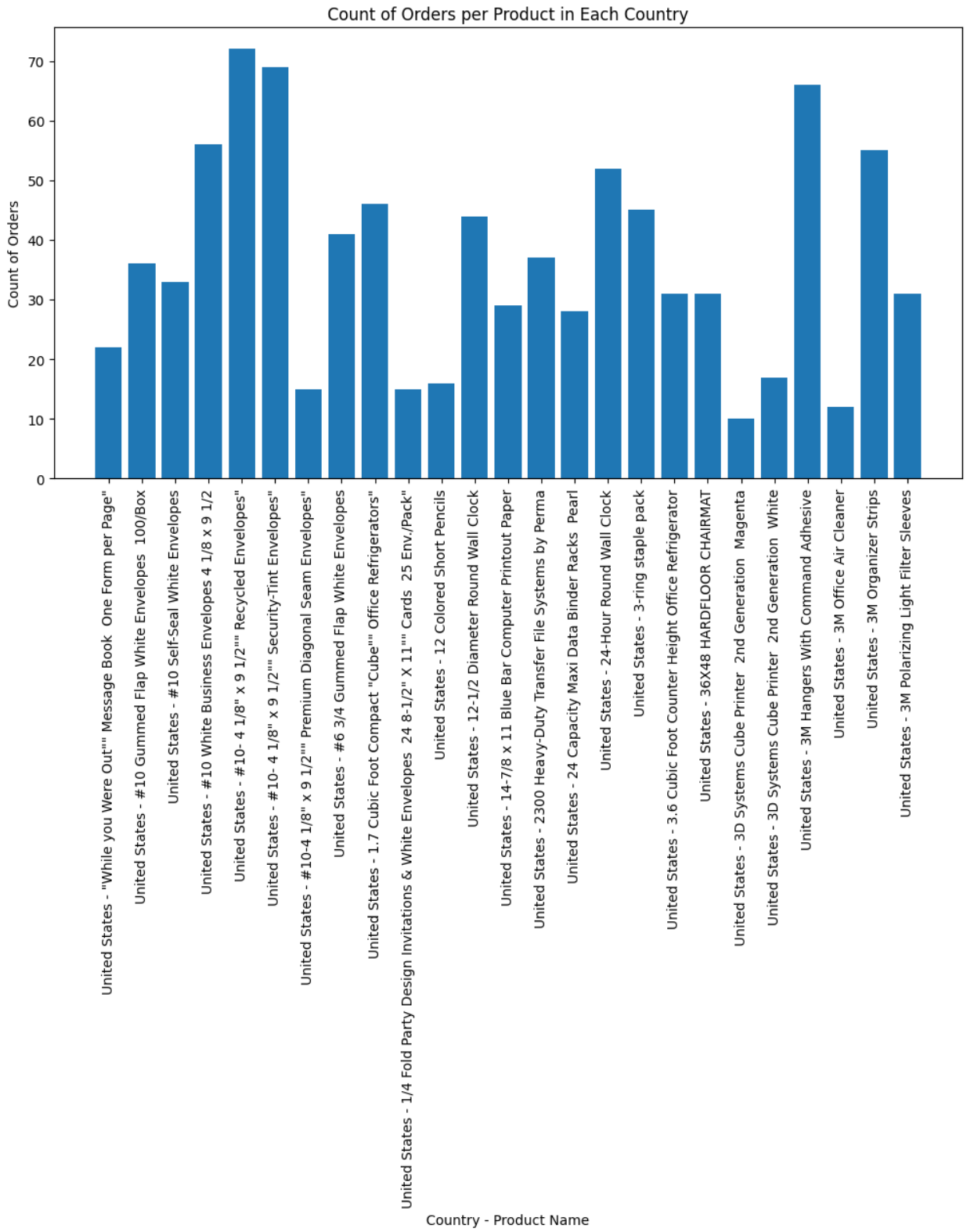
```
+---------------+---------------+------------+--------------------+
|     Product_Id|       Category|Sub_category|        Product_name|
+---------------+---------------+------------+--------------------+
|TEC-AC-10001767|     Technology| Accessories|SanDisk Ultra 64 ...|
|OFF-EN-10002504|Office Supplies|   Envelopes|Tyvek  Top-Openin...|
|TEC-PH-10001552|     Technology|      Phones|I Need's 3d Hello...|
|OFF-PA-10001937|Office Supplies|       Paper|            Xerox 21|
```

```
|OFF-PA-10003039|Office Supplies|       Paper|          Xerox 1960|
|OFF-AR-10001953|Office Supplies|         Art|Boston 1645 Delux...|
|OFF-BI-10000605|Office Supplies|     Binders|Acco Pressboard C...|
|FUR-FU-10003708|      Furniture| Furnishings|Tenex Traditional...|
|OFF-BI-10003460|Office Supplies|     Binders|   Acco 3-Hole Punch|
|FUR-FU-10002253|      Furniture| Furnishings|Howard Miller 13"...|
|OFF-PA-10000788|Office Supplies|       Paper|           Xerox 210|
|TEC-PH-10000148|     Technology|      Phones|Cyber Acoustics A...|
|TEC-AC-10001314|     Technology| Accessories|Case Logic 2.4GHz...|
|OFF-AR-10002053|Office Supplies|         Art|Premium Writing P...|
|OFF-ST-10002974|Office Supplies|     Storage|Trav-L-File Heavy...|
|FUR-CH-10000015|      Furniture|      Chairs|Hon Multipurpose ...|
|OFF-PA-10001667|Office Supplies|       Paper|Great White Multi...|
```

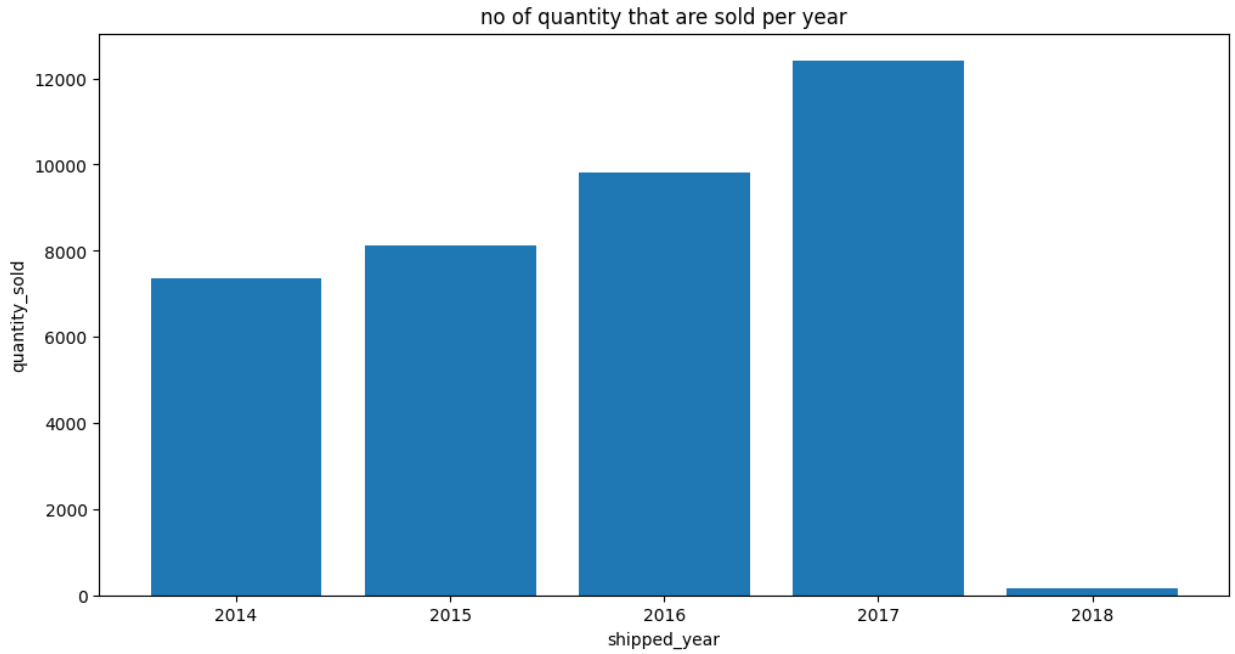
```
#df.coalesce(1).write.mode("overwrite").json("dbfs:/mnt/store/customer.json")
#dbutils.fs.unmount("/mnt/store")


#no of times a product appeared in order according to countries
product_per_category_country = spark.sql('select country, p.Product_name,
count(o.Order_Id) as count from orders_t o \
inner join Customers_t c on o.Customer_Id = c.Customer_Id \
inner join products_t p on o.Product_Id = p.Product_Id \
group by c.country, p.Product_name \
order by c.country, p.Product_name Limit 25')
product_per_category_country.show()

#writing the data to AzureSQL
table_name = "product_per_category_country"
product_per_category_country.write.jdbc(url=azure_sql_properties["url"],
table=table_name, mode="overwrite", properties=azure_sql_properties)

#presenting the data through visualization
import matplotlib.pyplot as plt
data = product_per_category_country.toPandas()
# Create a bar chart
plt.figure(figsize=(12, 6))
plt.bar(data['country'] + ' - ' + data['Product_name'], data['count'])
plt.xlabel("Country - Product Name")
plt.ylabel("Count of Orders")
plt.title("Count of Orders per Product in Each Country")
plt.xticks(rotation=90) # Rotate the x-axis labels for better readability
plt.show()
```

```
+--------------+--------------------+-----+
|       country|        Product_name|count|
+--------------+--------------------+-----+
|United States|"While you Were O...|   22|
```

```
|United States|#10 Gummed Flap W...|   36|
|United States|#10 Self-Seal Whi...|   33|
|United States|#10 White Busines...|   56|
|United States|#10- 4 1/8" x 9 1...|   72|
|United States|#10- 4 1/8" x 9 1...|   69|
|United States|#10-4 1/8" x 9 1/...|   15|
|United States|#6 3/4 Gummed Fla...|   41|
|United States|1.7 Cubic Foot Co...|   46|
|United States|1/4 Fold Party De...|   15|
|United States|12 Colored Short ...|   16|
|United States|12-1/2 Diameter R...|   44|
|United States|14-7/8 x 11 Blue ...|   29|
|United States|2300 Heavy-Duty T...|   37|
|United States|24 Capacity Maxi ...|   28|
|United States|24-Hour Round Wal...|   52|
|United States|  3-ring staple pack|   45|
|United States|3.6 Cubic Foot Co...|   31|
```

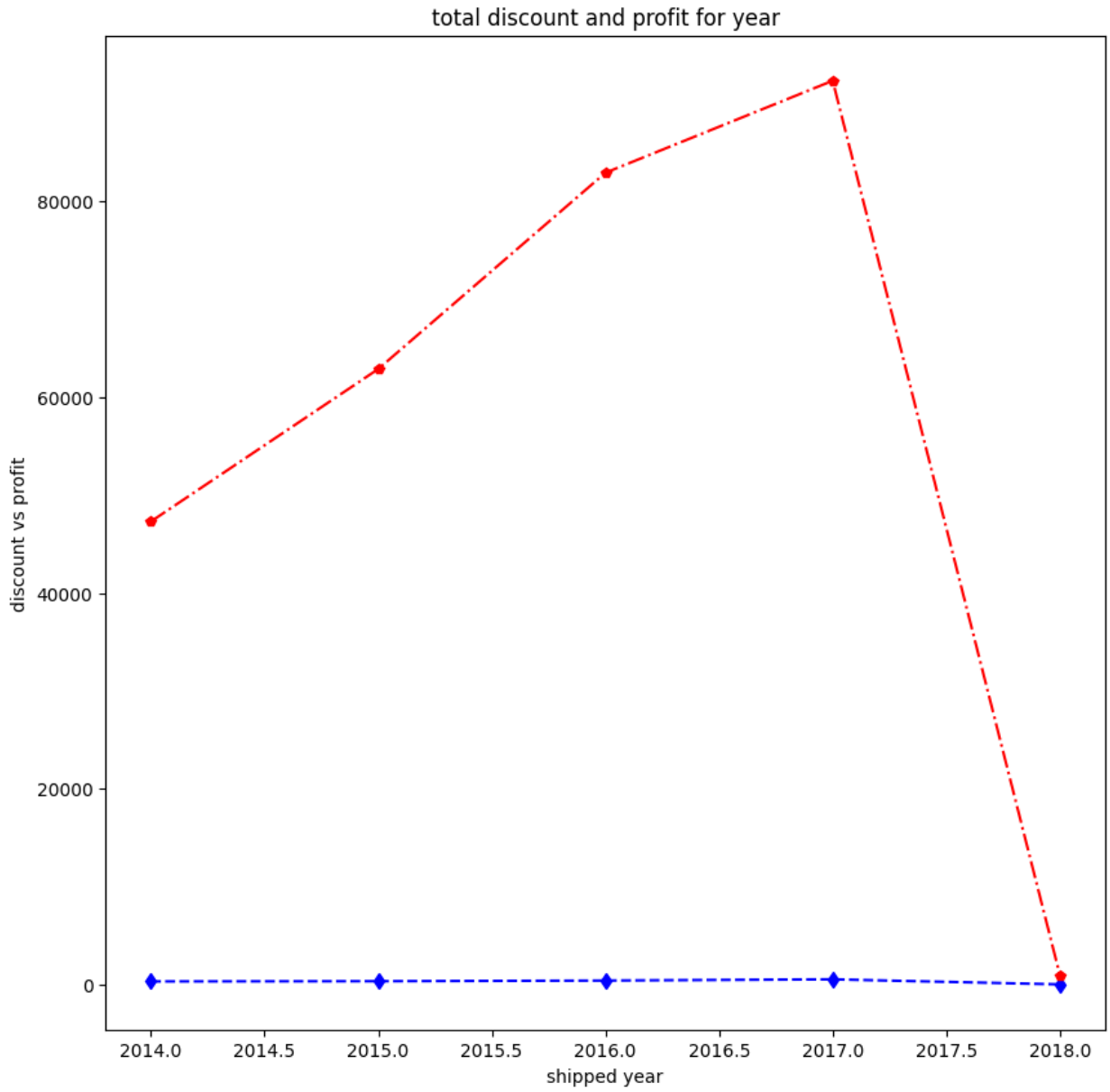

Count of Orders per Product in Each Country
Count of Orders
0
10
20
30
40
50
60
70
United States - "While you Were Out"" Message Book One Form per Page"
United States - #10 Gummed Flap White Envelopes 100/Box
United States - #10 Self-Seal White Envelopes
United States - #10 White Business Envelopes 4 1/8 x 9 1/2
United States - #10- 4 1/8" x 9 1/2"" Recycled Envelopes"
United States - #10- 4 1/8" x 9 1/2"" Security-Tint Envelopes"
United States - #10-4 1/8" x 9 1/2"" Premium Diagonal Seam Envelopes"
United States - #6 3/4 Gummed Flap White Envelopes
United States - 1.7 Cubic Foot Compact "Cube"" Office Refrigerators"
United States - 1/4 Fold Party Design Invitations & White Envelopes 24 8-1/2" X 11"" Cards 25 Env./Pack"
United States - 12 Colored Short Pencils
United States - 12-1/2 Diameter Round Wall Clock
United States - 14-7/8 x 11 Blue Bar Computer Printout Paper
United States - 2300 Heavy-Duty Transfer File Systems by Perma
United States - 24 Capacity Maxi Data Binder Racks Pearl
United States - 24-Hour Round Wall Clock
United States - 3-ring staple pack
United States - 3.6 Cubic Foot Counter Height Office Refrigerator
United States - 36X48 HARDFLOOR CHAIRMAT
United States - 3D Systems Cube Printer 2nd Generation Magenta
United States - 3D Systems Cube Printer 2nd Generation White
United States - 3M Hangers With Command Adhesive
United States - 3M Office Air Cleaner
United States - 3M Organizer Strips
United States - 3M Polarizing Light Filter Sleeves
Country - Product Name

```
#total no of quantity that are sold per year")
plt.show()
quantity_sold_per_year = spark.sql(" select year(Ship_date) as shipped_year,
sum(Quantity) quantity_sold from ships_t s \
inner join sales_t sl on s.order_Id = sl.order_Id \
group by shipped_year \
order by shipped_year")
quantity_sold_per_year.show()

#writing the data to AzureSQL
table_name = "quantity_sold_per_year"
quantity_sold_per_year.write.jdbc(url=azure_sql_properties["url"],
table=table_name, mode="overwrite", properties=azure_sql_properties)


#plot graph to show data visualy
import matplotlib.pyplot as plt
# Create a bar chart
plt.figure(figsize=(12, 6))
data = quantity_sold_per_year.toPandas()
plt.bar(data['shipped_year'],data['quantity_sold'])
plt.xlabel("shipped_year")
plt.ylabel("quantity_sold")
plt.title("no of quantity that are sold per year")
plt.show()


+------------+-------------+
|shipped_year|quantity_sold|
+------------+-------------+
|        2014|       7363.0|
|        2015|       8109.0|
|        2016|       9823.0|
|        2017|      12425.0|
|        2018|        151.0|
+------------+-------------+
```




```
#calculate total discount and profit for year
discount_profit_per_year=spark.sql(" select year(Ship_date) as shipped_year,
sum(Discount) total_discount,sum(Profit) total_profit from ships_t s \
inner join sales_t sl on s.order_Id = sl.order_Id \
group by shipped_year \
order by shipped_year")
discount_profit_per_year.show()

#writing the data to AzureSQL
table_name = "discount_profit_per_year"
discount_profit_per_year.write.jdbc(url=azure_sql_properties["url"],
table=table_name, mode="overwrite", properties=azure_sql_properties)

#plot line graph to show the data visually
import matplotlib.pyplot as plt
data=discount_profit_per_year.toPandas()
plt.figure(figsize=(10,10))
plt.plot(data['shipped_year'],data['total_discount'],linestyle='--
',color='blue',label='discount',marker='d')
plt.plot(data['shipped_year'],data['total_profit'],linestyle='-.',color='red',l
abel='profit',marker='p')
plt.xlabel('shipped year')
plt.ylabel('discount vs profit')
plt.title("total discount and profit for year")
plt.show()

+------------+------------------+-----------------+
|shipped_year|    total_discount|     total_profit|
+------------+------------------+-----------------+
```

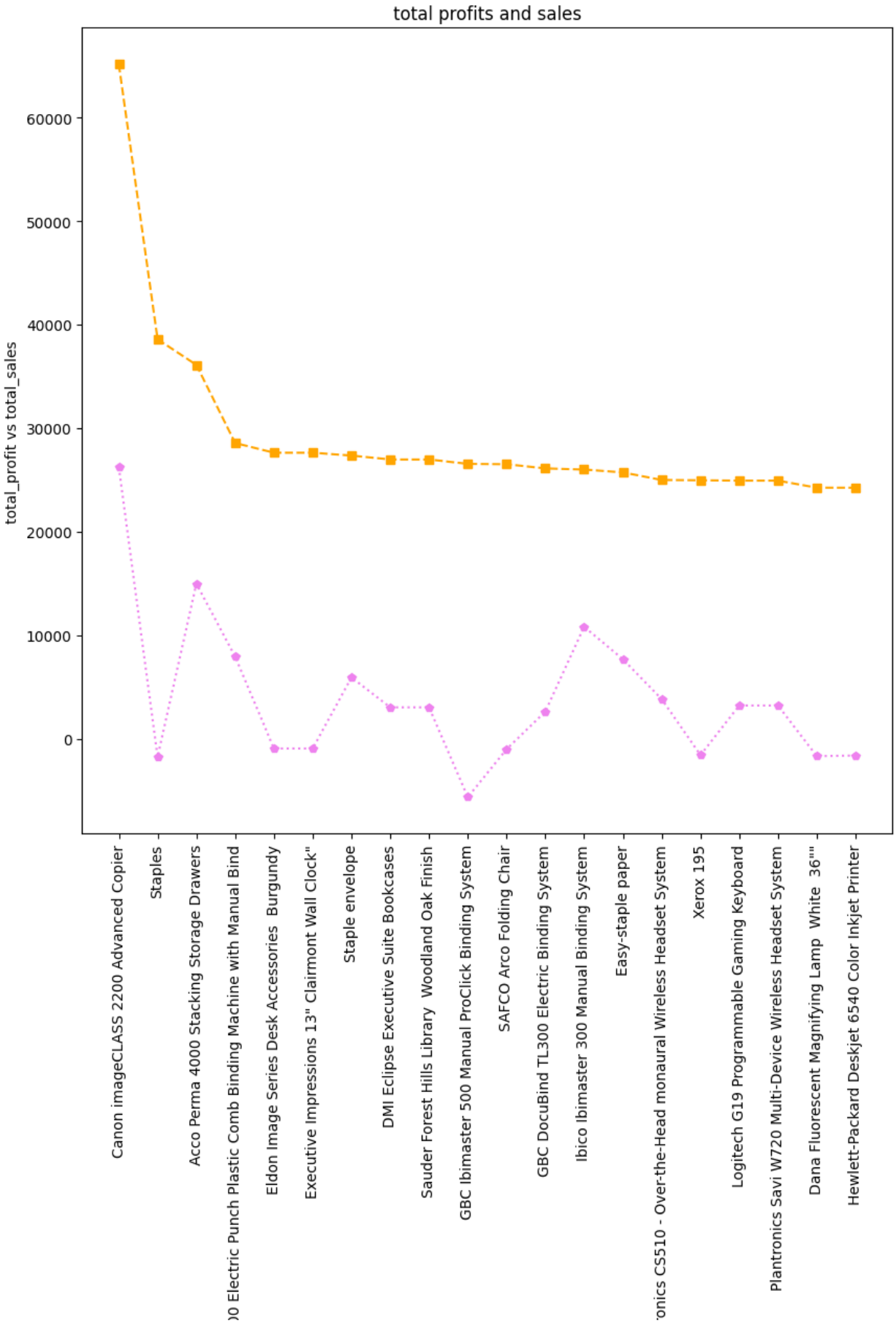
```
|        2014| 308.2599999999961|47304.78430000007|
|        2015|330.98999999999535|62881.52990000017|
|        2016| 396.2199999999938|82941.10169999996|
|        2017| 517.6999999999915|92346.87519999982|
|        2018| 7.620000000000001|934.7894000000001|
+------------+------------------+-----------------+
```

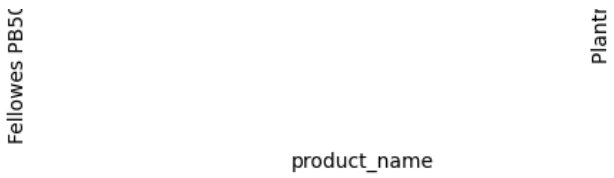


```
#select 20 product_name, total profits and sales according to max sales
max_product_profit_20=spark.sql('select product_name, sum(profit) total_profit,
sum(s.sales) as total_sales from products_t p \
join orders_t o on o.Product_Id = p.Product_Id \
join Sales_t s on o.order_id = s.order_id \
group by p.product_name \
order by total_sales desc limit 20')
max_product_profit_20.show()

#writing the data to AzureSQL
table_name = "max_product_profit_20"
max_product_profit_20.write.jdbc(url=azure_sql_properties["url"],
table=table_name, mode="overwrite", properties=azure_sql_properties)

#plot a line graph to show the data visually
import matplotlib.pyplot as plt
data=max_product_profit_20.toPandas()
plt.figure(figsize=(10,10))
plt.plot(data['product_name'],data['total_profit'],linestyle=':',marker='p',col
or='violet')
plt.plot(data['product_name'],data['total_sales'],linestyle='--
',marker='s',color='orange')
plt.xlabel('product_name')
plt.ylabel('total_profit vs total_sales')
plt.title('total profits and sales')
plt.xticks(rotation=90) # Rotate the x-axis labels for better readability
plt.show()
```

```
+--------------------+-------------------+------------------+
|        product_name|       total_profit|       total_sales|
+--------------------+-------------------+------------------+
|Canon imageCLASS ...| 26228.965600000003|65144.668000000005|
|             Staples|-1754.5587999999993| 38589.23000000002|
|Acco Perma 4000 S...| 14968.131100000002| 36112.30499999999|
|Fellowes PB500 El...|  7926.508300000002|         28566.544|
|Eldon Image Serie...| -953.2524999999999|          27626.53|
|Executive Impress...| -953.2524999999999|          27626.53|
|     Staple envelope|          5913.3291|27353.259000000002|
|DMI Eclipse Execu...| 3023.0678000000003|         26973.941|
|Sauder Forest Hil...| 3023.0678000000003| 26973.94099999999|
|GBC Ibimaster 500...|         -5654.5375|         26551.547|
|SAFCO Arco Foldin...|-1018.0236000000002|26514.726999999995|
|GBC DocuBind TL30...| 2617.6763000000005|         26112.959|
|Ibico Ibimaster 3...|         10810.3899|         26001.031|
|   Easy-staple paper| 7670.0686000000005|25726.921000000006|
|Plantronics CS510...| 3816.6946000000007| 24993.08599999999|
```

```
|           Xerox 195|         -1588.9422|         24961.262|
|Logitech G19 Prog...|  3196.986399999999|24928.610999999994|
```


total profits and sales
total_profit vs total_sales
60000
50000
40000
30000
20000
10000
0
Canon imageCLASS 2200 Advanced Copier
Staples
Acco Perma 4000 Stacking Storage Drawers
00 Electric Punch Plastic Comb Binding Machine with Manual Bind
Eldon Image Series Desk Accessories Burgundy
Executive Impressions 13" Clairmont Wall Clock"
Staple envelope
DMI Eclipse Executive Suite Bookcases
Sauder Forest Hills Library Woodland Oak Finish
GBC Ibimaster 500 Manual ProClick Binding System
SAFCO Arco Folding Chair
GBC DocuBind TL300 Electric Binding System
Ibico Ibimaster 300 Manual Binding System
Easy-staple paper
onics CS510 - Over-the-Head monaural Wireless Headset System
Xerox 195
Logitech G19 Programmable Gaming Keyboard
Plantronics Savi W720 Multi-Device Wireless Headset System
Dana Fluorescent Magnifying Lamp White 36""
Hewlett-Packard Deskjet 6540 Color Inkjet Printer




```
azure_sql_properties = {
    "url":
"jdbc:sqlserver://yourstorageserver.database.windows.net:1433;databaseName=your
dbname",
    "user": "username",
    "password": "password",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
}
table_name = "orders"
orders.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
table_name = "customers"
customers.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
table_name = "ships"
ships.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
table_name = "sales"
sales.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
table_name = "products"
products.write.jdbc(url=azure_sql_properties["url"], table=table_name,
mode="overwrite", properties=azure_sql_properties)
```

```
/mnt/store has been unmounted.
```

```
True
```