databricks AdventureWorks

(https://databricks.com)

Data Management with Databricks: Adventure Works Challenge

At Kaggle, I found a sample database of a fictional multinational company that sells bikes, accessories, and clothing. The sample database contains various tables, i.e. customers, products, returns, and sales. In this notebook, I create delta tables and a database using Databricks. After the sales database has been created, I catch a glimpse of the data. I do some data cleansing and I visualize the data. First, I load the csv files. So, let's get started with Databricks.

Before I load the csv files, I upload the csv files in the Catalog. After storing the files, I would like to display the content of the catalog. Because I have already created multiple folders in the catalog, I would like to display the information about the content of the Adventure Works folder. With dbutils, I can catch a glimpse of the file info. And here is the result:

```
#display information about the content of the catalog or folder of Adventure Works
dbutils.fs.ls("dbfs:/FileStore/AW")
```

```
Out[68]: [FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Customers.csv', name='AdventureWorks_Customers.csv', size=1963594,
modificationTime=1717340887000),
 FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Products.csv', name='AdventureWorks_Products.csv', size=63509, modificationT
ime=1717340886000),
 FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Returns.csv', name='AdventureWorks_Returns.csv', size=87401, modificationTim
e=1717340887000),
 FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Sales_2015.csv', name='AdventureWorks_Sales_2015.csv', size=194786, modifica
tionTime=1717340887000),
 FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Sales_2016.csv', name='AdventureWorks_Sales_2016.csv', size=1786110, modific
ationTime=1717340889000),
 FileInfo(path='dbfs:/FileStore/AW/AdventureWorks_Sales_2017.csv', name='AdventureWorks_Sales_2017.csv', size=2187175, modific
ationTime=1717340891000)]
```

After, I read the csv files using the Spark dataframe API (is the Spark Read Option). I start with the sales table or dataset. I have 3 equal sales datasets. Each dataset contains data from a different year: 2015, 2016, and 2017.

```
# Read csv files for sales using spark dataframeAPI
sales_raw_df = spark.read.option("header","true").csv("dbfs:/FileStore/AW/AdventureWorks_Sales_*.csv")

## Show the datafarme
sales_raw_df.show(n=5, truncate=False)
```

```
+---------+----------+-----------+----------+-----------+-------------+-------------+---------+-------------+-------------+
|OrderDate|StockDate |OrderNumber|ProductKey|CustomerKey|OrderLineItem|OrderQuantity|Region   |Country      |Continent    |
+---------+----------+-----------+----------+-----------+-------------+-------------+---------+-------------+-------------+
|1/1/2017 |12/13/2003|SO61285    |529       |23791      |2            |2            |Northwest|United States|North America|
|1/1/2017 |9/24/2003 |SO61285    |214       |23791      |3            |1            |Northwest|United States|North America|
|1/1/2017 |9/4/2003  |SO61285    |540       |23791      |1            |1            |Northwest|United States|North America|
|1/1/2017 |9/28/2003 |SO61301    |529       |16747      |2            |2            |Northwest|United States|North America|
|1/1/2017 |10/21/2003|SO61301    |377       |16747      |1            |1            |Northwest|United States|North America|
+---------+----------+-----------+----------+-----------+-------------+-------------+---------+-------------+-------------+
only showing top 5 rows
```

```
sales_raw_df.count()
```

```
Out[70]: 56046
```

The entire sales dataframe contains 56046 entries. I have counted all rows in the dataset, containing data from 2015 till 2017. Now, let's create the 'Sales Database'. I create the delta tables with the write mode, and I save the tables in the 'Sales Database'. I do this for the sales table, products table, returned products table, and the customers table.

```
# First, create Database SalesDB if it doesn't exist
dbsales = "SalesDB"

spark.sql(f"CREATE DATABASE IF NOT EXISTS {dbsales}")
spark.sql(f"USE {dbsales}")
```

```
Out[71]: DataFrame[]
```

```
sales_raw_df.write.mode("overwrite").format("delta").option("overwriteSchema","true").saveAsTable("SALES_RAW")
```

```
# Read the other csv files using spark dataframeAPI
customers_raw_df = spark.read.option("header","true").csv("dbfs:/FileStore/AW/AdventureWorks_Customers.csv")
products_raw_df = spark.read.option("header","true").csv("dbfs:/FileStore/AW/AdventureWorks_Products.csv")
returns_raw_df = spark.read.option("header","true").csv("dbfs:/FileStore/AW/AdventureWorks_Returns.csv")
```

```
## Create Delta Tables
customers_raw_df.write.mode("overwrite").format("delta").option("overwriteSchema","true").saveAsTable("CUSTOMERS_RAW")
products_raw_df.write.mode("overwrite").format("delta").option("overwriteSchema","true").saveAsTable("PRODUCTS_RAW")
returns_raw_df.write.mode("overwrite").format("delta").option("overwriteSchema","true").saveAsTable("RETURNS_RAW")
```

Let's show the contents of the 'Sales Database' I created. You can either use Python or SQL to show the tables.

```
display(spark.sql(f"SHOW TABLES"))
```

Table   New result table: ON

	database	tableName	isTemporary
1	salesdb	customers	false
2	salesdb	customers_raw	false
3	salesdb	products	false
4	salesdb	products_raw	false
5	salesdb	returns	false
6	salesdb	returns_raw	false
7	salesdb	sales	false
8	salesdb	sales_raw	false

8 rows

```
%sql
-- Switch to SQL Cell using %SQL
SHOW tables
```

Table   New result table: ON

	database	tableName	isTemporary
1	salesdb	customers	false
2	salesdb	customers_raw	false
3	salesdb	products	false
4	salesdb	products_raw	false
5	salesdb	returns	false
6	salesdb	returns_raw	false

7	salesdb	sales	false
8	salesdb	sales_raw	false

8 rows

Let's use the SQL command to count the number of sales entries from the sales table. Also, let's show the details of the delta table 'Sale'. And lastly, let's catch a glimpse of the first rows of all tables using SQL.

Sales Table

```
%sql
select count(*) from sales_raw;
```

Table   New result table: ON

	count(1)
1	56046

1 row

```
%sql

describe DETAIL sales_raw;
```

Table   New result table: ON

	format	id	name	description	location
1	delta	b81e1a79-eb1a-44b9-84bb-8145a1c634...	spark_catalog.salesdb.sales_r...	null	dbfs:/user/hive/warehouse/salesdb

1 row

```
%sql

select * from sales_raw limit 5;
```

Table   New result table: ON

	OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	OrderLineItem
1	1/1/2015	9/21/2001	SO45080	332	14657	1
2	1/1/2015	12/5/2001	SO45079	312	29255	1
3	1/1/2015	10/29/2001	SO45082	350	11455	1
4	1/1/2015	11/16/2001	SO45081	338	26782	1

5 rows

Customers Table

```
%sql

select * from customers_raw limit 5;
```

Table New result table: ON

	CustomerKey	Prefix	FirstName	LastName	BirthDate	MaritalStatus	Gender
1	11000	MR.	JON	YANG	4/8/1966	M	M
2	11001	MR.	EUGENE	HUANG	5/14/1965	S	M
3	11002	MR.	RUBEN	TORRES	8/12/1965	M	M
4	11003	MS.	CHRISTY	ZHU	2/15/1968	S	F

5 rows

Products Table

```
%sql
select count(*) from products_raw;
```

Table New result table: ON

	count(1)
1	293

1 row

```
%sql

select * from products_raw limit 5;
```

Table New result table: ON

	ProductKey	CategoryName	ProductSubcategory	ProductSKU	ProductName	ModelNar
1	214	Accessories	Helmets	HL-U509-R	Sport-100 Helmet, Red	Sport-100
2	215	Accessories	Helmets	HL-U509	Sport-100 Helmet, Black	Sport-100
3	218	Clothing	Socks	SO-B909-M	Mountain Bike Socks, M	Mountain Bike
4	219	Clothing	Socks	SO-B909-L	Mountain Bike Socks, L	Mountain Bike

5 rows

Product Returns Table

```
%sql

select * from returns_raw limit 5;
```

Table New result table: ON

	ReturnDate	Continent	Country	Region	ProductKey	ReturnQuantity
1	1/18/2015	Pacific	Australia	Australia	312	1
2	1/18/2015	Europe	United Kingdom	United Kingdom	310	1
3	1/21/2015	Europe	Germany	Germany	346	1
4	1/22/2015	North America	United States	Southwest	311	1
5	2/2/2015	North America	Canada	Canada	312	1

5 rows

Transform Data in the Delta Table

All previously loaded data still has string formats. However, some tables contain dates and numerical data. For example, the 'Sales table' has OrderDate and StockDate which are dates. Order Quantity is a numerical value. So, let's convert these so that we can use this in the data analysis.

Sales Table

```
#read Delta Table using spark dataframe
sales_df=  spark.read.table("salesdb.sales_raw")
#And show the first 5 rows
sales_df.show(n=5,truncate=False)
```

```
+---------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+-----------
--+
|OrderDate|StockDate |OrderNumber|ProductKey|CustomerKey|OrderLineItem|OrderQuantity|Region        |Country       |Continent
|
+---------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+-----------
--+
|1/1/2015 |9/21/2001 |SO45080    |332       |14657      |1            |1            |Northwest     |United States |North Ameri
ca|
|1/1/2015 |12/5/2001 |SO45079    |312       |29255      |1            |1            |Southwest     |United States |North Ameri
ca|
|1/1/2015 |10/29/2001|SO45082    |350       |11455      |1            |1            |Australia     |Australia     |Pacific
|
|1/1/2015 |11/16/2001|SO45081    |338       |26782      |1            |1            |Canada        |Canada        |North Ameri
ca|
|1/2/2015 |12/15/2001|SO45083    |312       |14947      |1            |1            |United Kingdom|United Kingdom|Europe
|
+---------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+-----------
--+
only showing top 5 rows
```

```
#Use withColumn method & datetype to convert the columns OrderDate and StockDate to a date format instead of string format
from pyspark.sql.functions import *
from pyspark.sql.types import DateType
from datetime import datetime
from pyspark.sql.functions import col, udf
func = udf(lambda x: datetime.strptime(x,'%m/%d/%Y'),DateType())
sales_df =  sales_df.withColumn("OrderDate",func(col('OrderDate')))
sales_df.show(n=5,truncate=False)
```

```
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|OrderDate |StockDate |OrderNumber|ProductKey|CustomerKey|OrderLineItem|OrderQuantity|Region        |Country       |Continent
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|2015-01-01|9/21/2001 |SO45080    |332       |14657      |1            |1            |Northwest     |United States |North Amer
ica|
|2015-01-01|12/5/2001 |SO45079    |312       |29255      |1            |1            |Southwest     |United States |North Amer
ica|
|2015-01-01|10/29/2001|SO45082    |350       |11455      |1            |1            |Australia     |Australia     |Pacific
|
|2015-01-01|11/16/2001|SO45081    |338       |26782      |1            |1            |Canada        |Canada        |North Amer
ica|
|2015-01-02|12/15/2001|SO45083    |312       |14947      |1            |1            |United Kingdom|United Kingdom|Europe
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
only showing top 5 rows
```

```
func = udf(lambda x: datetime.strptime(x,'%m/%d/%Y'),DateType())
sales_df =  sales_df.withColumn("StockDate",func(col('StockDate')))
sales_df.show(n=5,truncate=False)
```

```
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|OrderDate |StockDate |OrderNumber|ProductKey|CustomerKey|OrderLineItem|OrderQuantity|Region        |Country       |Continent   
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|2015-01-01|2001-09-21|SO45080    |332       |14657      |1            |1            |Northwest     |United States |North Amer
ica|
|2015-01-01|2001-12-05|SO45079    |312       |29255      |1            |1            |Southwest     |United States |North Amer
ica|
|2015-01-01|2001-10-29|SO45082    |350       |11455      |1            |1            |Australia     |Australia     |Pacific   
|
|2015-01-01|2001-11-16|SO45081    |338       |26782      |1            |1            |Canada        |Canada        |North Amer
ica|
|2015-01-02|2001-12-15|SO45083    |312       |14947      |1            |1            |United Kingdom|United Kingdom|Europe    
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
only showing top 5 rows
```

Now, I converted the dates in the 'Sales table'. Let's check for missing values also.

```
# Count missing values for each column
display(sales_df.select([count(when(col(c).isNull(),c)).alias(c) for c in sales_df.columns]))
```

Table

New result table: ON

	OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	OrderLineItem
1	0	0	0	0	0	0

1 row

```
#convert return quantity to a number
from pyspark.sql.types import IntegerType
sales_df =  sales_df.withColumn("OrderQuantity",sales_df["OrderQuantity"].cast(IntegerType()))
sales_df.show(n=5,truncate=False)
```

```
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|OrderDate |StockDate |OrderNumber|ProductKey|CustomerKey|OrderLineItem|OrderQuantity|Region        |Country       |Continent   
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
|2015-01-01|2001-09-21|SO45080    |332       |14657      |1            |1            |Northwest     |United States |North Amer
ica|
|2015-01-01|2001-12-05|SO45079    |312       |29255      |1            |1            |Southwest     |United States |North Amer
ica|
|2015-01-01|2001-10-29|SO45082    |350       |11455      |1            |1            |Australia     |Australia     |Pacific   
|
|2015-01-01|2001-11-16|SO45081    |338       |26782      |1            |1            |Canada        |Canada        |North Amer
ica|
|2015-01-02|2001-12-15|SO45083    |312       |14947      |1            |1            |United Kingdom|United Kingdom|Europe    
|
+----------+----------+-----------+----------+-----------+-------------+-------------+--------------+--------------+----------
---+
only showing top 5 rows
```

No values are missing in the 'Sales table'. Also, I converted the order quantity into a numerical value. I do the same for the other tables.

Customers Table

```
#customers_df
#read Delta Table using spark dataframe
customers_df=  spark.read.table("salesdb.customers_raw")
#And show the first 5 rows
customers_df.show(n=5,truncate=False)
```

```
+-----------+------+---------+--------+---------+-------------+------+------------------------------+------------+------------
-+--------------+------------+---------+
|CustomerKey|Prefix|FirstName|LastName|BirthDate|MaritalStatus|Gender|EmailAddress                  |AnnualIncome|TotalChildre
n|EducationLevel|Occupation  |HomeOwner|
+-----------+------+---------+--------+---------+-------------+------+------------------------------+------------+------------
-+--------------+------------+---------+
|11000      |MR.   |JON      |YANG    |4/8/1966 |M            |M     |jon24@adventure-works.com     |$90,000     |2
|Bachelors     |Professional|Y        |
|11001      |MR.   |EUGENE   |HUANG   |5/14/1965|S            |M     |eugene10@adventure-works.com  |$60,000     |3
|Bachelors     |Professional|N        |
|11002      |MR.   |RUBEN    |TORRES  |8/12/1965|M            |M     |ruben35@adventure-works.com   |$60,000     |3
|Bachelors     |Professional|Y        |
|11003      |MS.   |CHRISTY  |ZHU     |2/15/1968|S            |F     |christy12@adventure-works.com |$70,000     |0
|Bachelors     |Professional|N        |
|11004      |MRS.  |ELIZABETH|JOHNSON |8/8/1968 |S            |F     |elizabeth5@adventure-works.com|$80,000     |5
|Bachelors     |Professional|Y        |
+-----------+------+---------+--------+---------+-------------+------+------------------------------+------------+------------
-+--------------+------------+---------+
only showing top 5 rows
```

```
#Convert the date of birth to the correct format
func = udf(lambda x: datetime.strptime(x,'%m/%d/%Y'),DateType())
customers_df =  customers_df.withColumn("BirthDate",func(col('BirthDate')))
customers_df.show(n=5,truncate=False)
```

```
+-----------+------+---------+--------+----------+-------------+------+------------------------------+------------+-----------
--+--------------+------------+---------+
|CustomerKey|Prefix|FirstName|LastName|BirthDate |MaritalStatus|Gender|EmailAddress                  |AnnualIncome|TotalChildr
en|EducationLevel|Occupation  |HomeOwner|
+-----------+------+---------+--------+----------+-------------+------+------------------------------+------------+-----------
--+--------------+------------+---------+
|11000      |MR.   |JON      |YANG    |1966-04-08|M            |M     |jon24@adventure-works.com     |$90,000     |2
|Bachelors     |Professional|Y        |
|11001      |MR.   |EUGENE   |HUANG   |1965-05-14|S            |M     |eugene10@adventure-works.com  |$60,000     |3
|Bachelors     |Professional|N        |
|11002      |MR.   |RUBEN    |TORRES  |1965-08-12|M            |M     |ruben35@adventure-works.com   |$60,000     |3
|Bachelors     |Professional|Y        |
|11003      |MS.   |CHRISTY  |ZHU     |1968-02-15|S            |F     |christy12@adventure-works.com |$70,000     |0
|Bachelors     |Professional|N        |
|11004      |MRS.  |ELIZABETH|JOHNSON |1968-08-08|S            |F     |elizabeth5@adventure-works.com|$80,000     |5
|Bachelors     |Professional|Y        |
+-----------+------+---------+--------+----------+-------------+------+------------------------------+------------+-----------
--+--------------+------------+---------+
only showing top 5 rows
```

```
# Count missing values for each column
display(customers_df.select([count(when(col(c).isNull(),c)).alias(c) for c in customers_df.columns]))
```

Table   New result table: ON

	CustomerKey	Prefix	FirstName	LastName	BirthDate	MaritalStatus	Gender
1	0	130	0	0	0	0	

1 row

The prefix has 130 missing values. This is not important for the data analysis part. So, here, I ignore this.

```
#Total children is also an integer
customers_df =  customers_df.withColumn("TotalChildren",customers_df["TotalChildren"].cast(IntegerType()))
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import regexp_replace
customers_df = customers_df.withColumn("AnnualIncome", regexp_replace("AnnualIncome", '[^0-9]', ''))
customers_df =  customers_df.withColumn("AnnualIncome",customers_df["AnnualIncome"].cast(IntegerType()))
#And finally show the table
customers_df.show(n=5,truncate=False)
```

```
+-----------+------+---------+--------+----------+-------------+------+-----------------------------+------------+-----------
--+--------------+------------+---------+
|CustomerKey|Prefix|FirstName|LastName|BirthDate |MaritalStatus|Gender|EmailAddress                 |AnnualIncome|TotalChildr
en|EducationLevel|Occupation  |HomeOwner|
+-----------+------+---------+--------+----------+-------------+------+-----------------------------+------------+-----------
--+--------------+------------+---------+
|11000      |MR.   |JON      |YANG    |1966-04-08|M            |M     |jon24@adventure-works.com    |90000       |2
|Bachelors     |Professional|Y        |
|11001      |MR.   |EUGENE   |HUANG   |1965-05-14|S            |M     |eugene10@adventure-works.com |60000       |3
|Bachelors     |Professional|N        |
|11002      |MR.   |RUBEN    |TORRES  |1965-08-12|M            |M     |ruben35@adventure-works.com  |60000       |3
|Bachelors     |Professional|Y        |
|11003      |MS.   |CHRISTY  |ZHU     |1968-02-15|S            |F     |christy12@adventure-works.com |70000       |0
|Bachelors     |Professional|N        |
|11004      |MRS.  |ELIZABETH|JOHNSON |1968-08-08|S            |F     |elizabeth5@adventure-works.com|80000       |5
|Bachelors     |Professional|Y        |
+-----------+------+---------+--------+----------+-------------+------+-----------------------------+------------+-----------
--+--------------+------------+---------+
only showing top 5 rows
```

Product Returns Table

```
#returns_df
#read Delta Table using spark dataframe
returns_df=  spark.read.table("salesdb.returns_raw")
#And show the first 5 rows
returns_df.show(n=5,truncate=False)
```

```
+----------+-------------+--------------+--------------+----------+--------------+
|ReturnDate|Continent    |Country       |Region        |ProductKey|ReturnQuantity|
+----------+-------------+--------------+--------------+----------+--------------+
|1/18/2015 |Pacific      |Australia     |Australia     |312       |1             |
|1/18/2015 |Europe       |United Kingdom|United Kingdom|310       |1             |
|1/21/2015 |Europe       |Germany       |Germany       |346       |1             |
|1/22/2015 |North America|United States |Southwest     |311       |1             |
|2/2/2015  |North America|Canada        |Canada        |312       |1             |
+----------+-------------+--------------+--------------+----------+--------------+
only showing top 5 rows
```

```
#Convert the date of returns to the correct format
func = udf(lambda x: datetime.strptime(x,'%m/%d/%Y'),DateType())
returns_df =  returns_df.withColumn("ReturnDate",func(col('ReturnDate')))
returns_df.show(n=5,truncate=False)
```

```
+----------+-------------+--------------+--------------+----------+--------------+
|ReturnDate|Continent    |Country       |Region        |ProductKey|ReturnQuantity|
+----------+-------------+--------------+--------------+----------+--------------+
|2015-01-18|Pacific      |Australia     |Australia     |312       |1             |
|2015-01-18|Europe       |United Kingdom|United Kingdom|310       |1             |
|2015-01-21|Europe       |Germany       |Germany       |346       |1             |
|2015-01-22|North America|United States |Southwest     |311       |1             |
|2015-02-02|North America|Canada        |Canada        |312       |1             |
+----------+-------------+--------------+--------------+----------+--------------+
only showing top 5 rows
```

```
# Count missing values for each column
display(returns_df.select([count(when(col(c).isNull(),c)).alias(c) for c in returns_df.columns]))
```

Table   New result table: ON

	ReturnDate	Continent	Country	Region	ProductKey	ReturnQuantity
1	0	0	0	0	0	0

1 row

```
#convert return quantity to a number
returns_df =  returns_df.withColumn("ReturnQuantity",returns_df["ReturnQuantity"].cast(IntegerType()))
returns_df.show(n=5,truncate=False)
```

```
+----------+-------------+--------------+--------------+----------+--------------+
|ReturnDate|Continent    |Country       |Region        |ProductKey|ReturnQuantity|
+----------+-------------+--------------+--------------+----------+--------------+
|2015-01-18|Pacific      |Australia     |Australia     |312       |1             |
|2015-01-18|Europe       |United Kingdom|United Kingdom|310       |1             |
|2015-01-21|Europe       |Germany       |Germany       |346       |1             |
|2015-01-22|North America|United States |Southwest     |311       |1             |
|2015-02-02|North America|Canada        |Canada        |312       |1             |
+----------+-------------+--------------+--------------+----------+--------------+
only showing top 5 rows
```

Products Table

```
#read Delta Table using spark dataframe
products_df=  spark.read.table("salesdb.products_raw")
#And show the first 5 rows
products_df.show(n=5,truncate=False)
```

```
+----------+------------+-----------------+----------+----------------------+-------------------+-----------------------
-----------------------------------------------------------------------+------------+-----------+------------+-----------+---
---------+
|ProductKey|CategoryName|ProductSubcategory|ProductSKU|ProductName           |ModelName          |ProductDescription
|ProductColor|ProductSize|ProductStyle|ProductCost|ProductPrice|
+----------+------------+-----------------+----------+----------------------+-------------------+-----------------------
-----------------------------------------------------------------------+------------+-----------+------------+-----------+---
---------+
|214       |Accessories |Helmets          |HL-U509-R |Sport-100 Helmet, Red  |Sport-100          |Universal fit, well-vent
ed, lightweight , snap-on visor.                                       |Red         |0          |0           |13.0863    |34.
99        |
|215       |Accessories |Helmets          |HL-U509   |Sport-100 Helmet, Black|Sport-100          |Universal fit, well-vent
ed, lightweight , snap-on visor.                                       |Black       |0          |0           |12.0278    |33.
6442      |
|218       |Clothing    |Socks            |SO-B909-M |Mountain Bike Socks, M |Mountain Bike Socks|Combination of natural a
nd synthetic fibers stays dry and provides just the right cushioning.|White       |M          |U           |3.3963     |9.5
|
|219       |Clothing    |Socks            |SO-B909-L |Mountain Bike Socks, L |Mountain Bike Socks|Combination of natural a
nd synthetic fibers stays dry and provides just the right cushioning.|White       |L          |U           |3.3963     |9.5
|
|220       |Accessories |Helmets          |HL-U509-B |Sport-100 Helmet, Blue |Sport-100          |Universal fit, well-vent
```

```
from pyspark.sql.types import DoubleType
from pyspark.sql.functions import col, round, format_number

# Convert columns to DoubleType and round them to 2 decimal places
products_df = products_df.withColumn("ProductCost", round(col("ProductCost").cast(DoubleType()), 2))
products_df = products_df.withColumn("ProductPrice", round(col("ProductPrice").cast(DoubleType()), 2))

# If you want to keep the values as strings with exactly 2 decimal places
products_df = products_df.withColumn("ProductCost", format_number(col("ProductCost"), 2))
products_df = products_df.withColumn("ProductPrice", format_number(col("ProductPrice"), 2))
```

Create Delta Table for Adjustments

```
# Save all the adjustments in the sales database
spark.sql(f"USE salesdb")

## Create DeltaTable for the adjusted sales table, customer table, product table, and returns table:

sales_df.write.mode("overwrite").format("delta").saveAsTable("SALES")
customers_df.write.mode("overwrite").format("delta").option("overwriteSchema", "true").saveAsTable("CUSTOMERS")
returns_df.write.mode("overwrite").format("delta").saveAsTable("RETURNS")
products_df.write.mode("overwrite").format("delta").saveAsTable("PRODUCTS")


## Validate that the table was created successfully
display(spark.sql(f"SHOW TABLES"))
```

Table   New result table: ON

	database	tableName	isTemporary
1	salesdb	customers	false
2	salesdb	customers_raw	false
3	salesdb	products	false
4	salesdb	products_raw	false
5	salesdb	returns	false
6	salesdb	returns_raw	false
7	salesdb	sales	false
8	salesdb	sales_raw	false

8 rows

Data Visualization

First, I catch a glimpse of the products and sales table again. This would make it easier for the data analysis. Databricks has some nice build-in data visuals. I use SQL.

```
%sql

select * from products limit 5;
```

Table   New result table: ON

	ProductKey	CategoryName	ProductSubcategory	ProductSKU	ProductName	ModelNar
1	214	Accessories	Helmets	HL-U509-R	Sport-100 Helmet, Red	Sport-100
2	215	Accessories	Helmets	HL-U509	Sport-100 Helmet, Black	Sport-100
3	218	Clothing	Socks	SO-B909-M	Mountain Bike Socks, M	Mountain Bike
4	219	Clothing	Socks	SO-B909-L	Mountain Bike Socks, L	Mountain Bike

5 rows

```
%sql

select * from sales limit 5;
```

Table   New result table: ON

	OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	OrderLineItem
1	2015-01-01	2001-09-21	SO45080	332	14657	1
2	2015-01-01	2001-12-05	SO45079	312	29255	1
3	2015-01-01	2001-10-29	SO45082	350	11455	1
4	2015-01-01	2001-11-16	SO45081	338	26782	1

5 rows

Ordered Quantity by Country

```
%sql

select Country, sum(OrderQuantity) from sales group by Country;
```

Table   Visualization 1

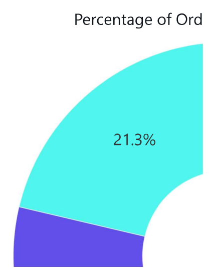
+
−
29,823.00
24,332.75
18,842.50
13,352.25
7,862.00

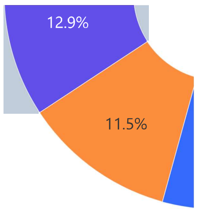
6 rows

```
%sql

select Country, sum(OrderQuantity) from sales group by Country;
```

Table   Visualization 1



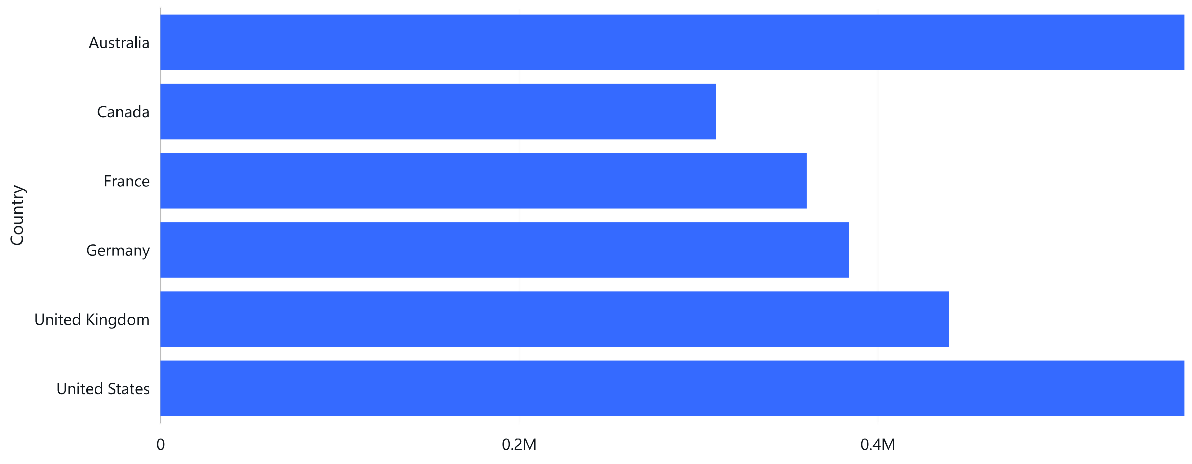


6 rows

Total Sales by Country

```
%sql
select S.Country, SUM(P.ProductPrice * S.OrderQuantity) AS TotalSales from salesdb.sales S join
salesdb.products P
on S.ProductKey=P.ProductKey
group by S.Country order by TotalSales desc;
```

Table | Visualization 1

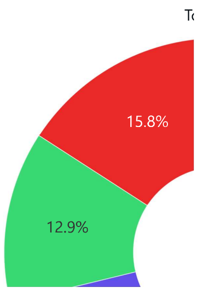


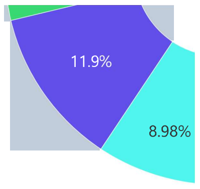
6 rows

Total Profit by Country

```
%sql
select S.Country, ROUND((SUM(P.ProductPrice * S.OrderQuantity) - SUM(P.ProductCost * S.OrderQuantity)), 2) AS TotalProfit from
salesdb.products P
on S.ProductKey=P.ProductKey
group by S.Country order by TotalProfit desc;
```

Table | Visualization 1 | Visualization 2



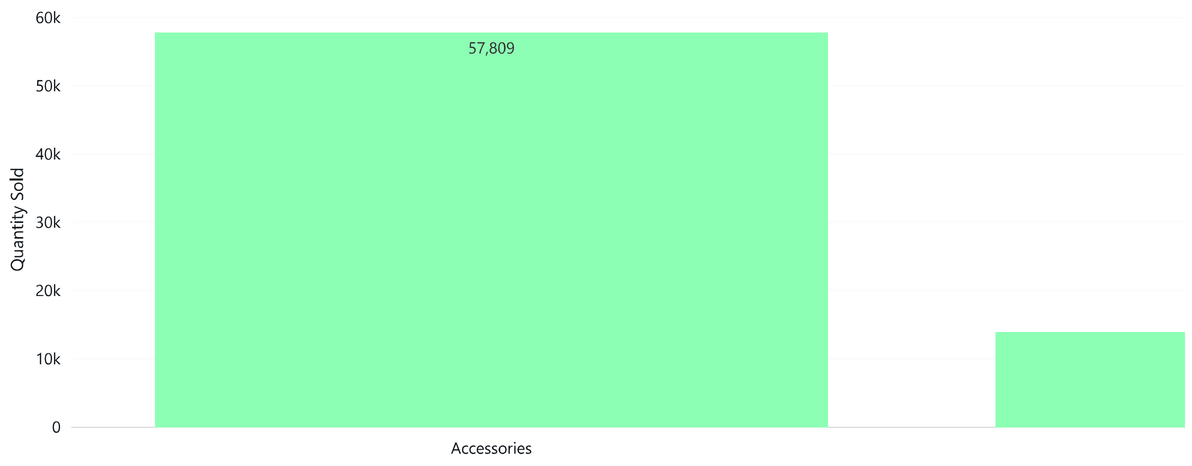


6 rows

Most Sales by Product Category

```
%sql
select P.CategoryName, SUM(S.OrderQuantity) as QuantitySold from salesdb.sales S join
salesdb.products P
on S.ProductKey=P.ProductKey
group by P.CategoryName order by QuantitySold desc;
```

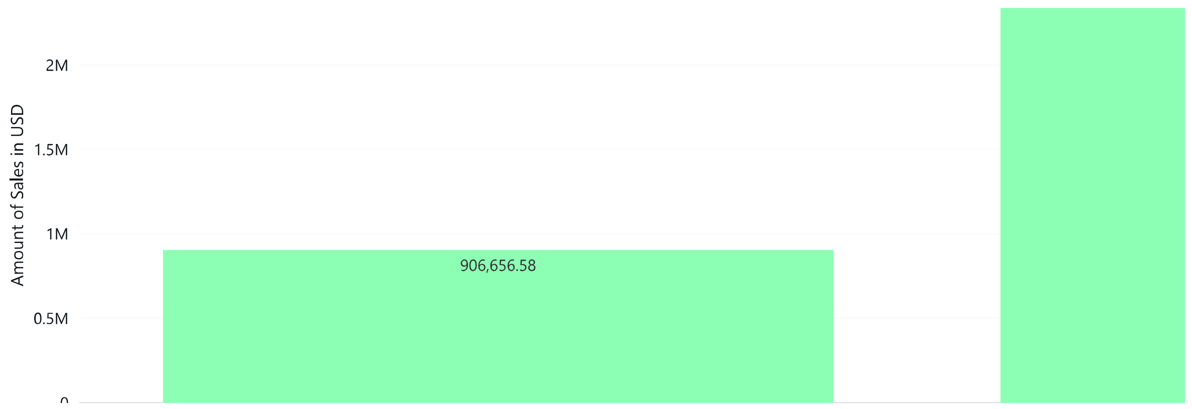
Table Visualization 1



3 rows

```
%sql
select P.CategoryName, ROUND(SUM(P.ProductPrice * S.OrderQuantity),2) AS TotalSales from salesdb.sales S join
salesdb.products P
on S.ProductKey=P.ProductKey
group by P.CategoryName order by TotalSales desc;
```

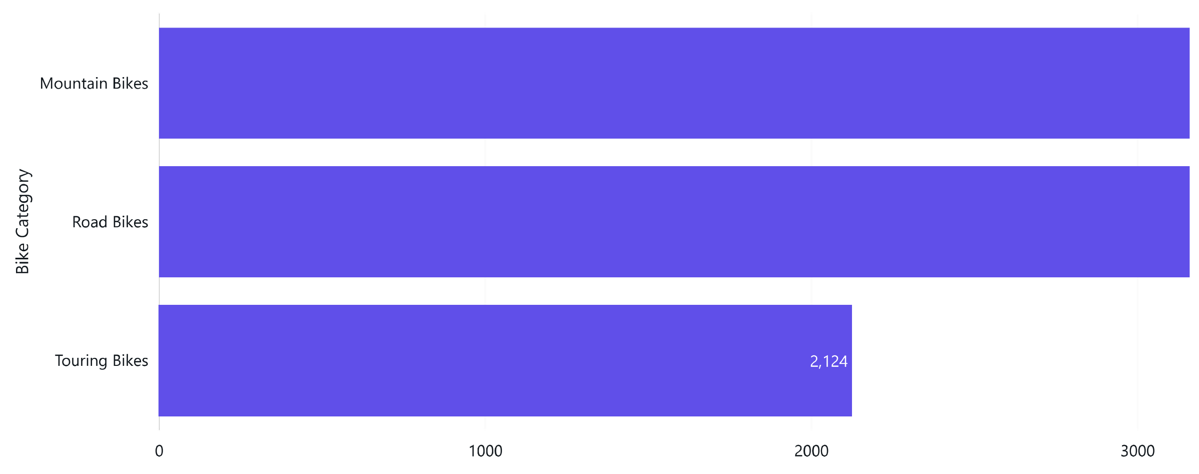
Table Visualization 1



3 rows

```
%sql
select P.ProductSubcategory, SUM(S.OrderQuantity) as QuantitySold from salesdb.sales S join
salesdb.products P
on S.ProductKey=P.ProductKey
where P.ProductSubcategory like '%Bikes%' group by P.ProductSubcategory order by QuantitySold desc;
```

Table | Visualization 1



3 rows

```
%sql
select P.ProductSubcategory, ROUND(SUM(P.ProductPrice * S.OrderQuantity),2) AS TotalSales from salesdb.sales S join
salesdb.products P
on S.ProductKey=P.ProductKey
where P.ProductSubcategory like '%Bikes%' group by P.ProductSubcategory order by TotalSales desc;
```

Table | Visualization 1 | New result table: ON

	ProductSubcategory	TotalSales
1	Road Bikes	1272615.98
2	Mountain Bikes	671178.44
3	Touring Bikes	393445.5

3 rows

Returned Products

```
%sql
select * from returns limit 5;
```

Table | New result table: ON

	ReturnDate	Continent	Country	Region	ProductKey	ReturnQuantity
1	2015-01-18	Pacific	Australia	Australia	312	1
2	2015-01-18	Europe	United Kingdom	United Kingdom	310	1
3	2015-01-21	Europe	Germany	Germany	346	1

4	2015-01-22	North America	United States	Southwest	311	1
5	2015-02-02	North America	Canada	Canada	312	1

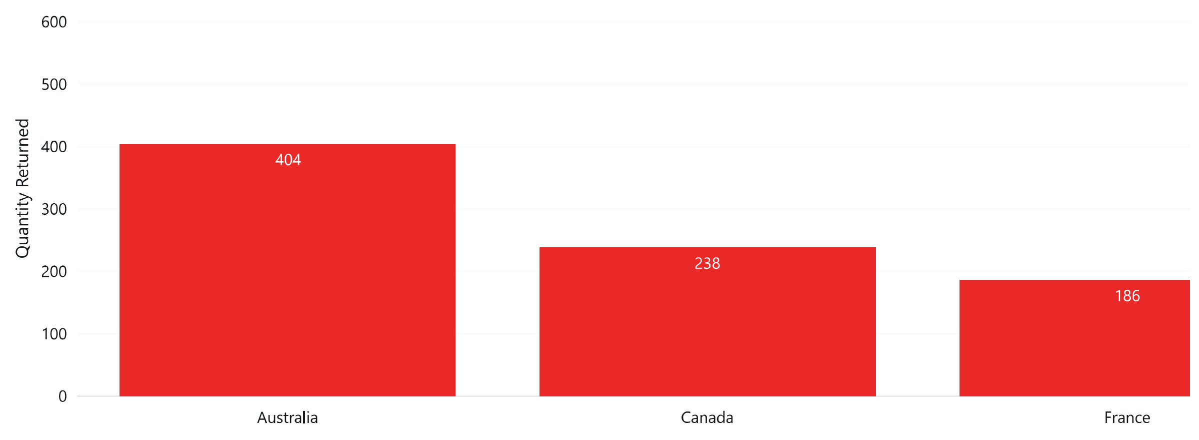
5 rows

Quantity Returned by Country

```
%sql

select Country, SUM(ReturnQuantity) as quantityreturned from returns group by Country;
```

Table Visualization 1



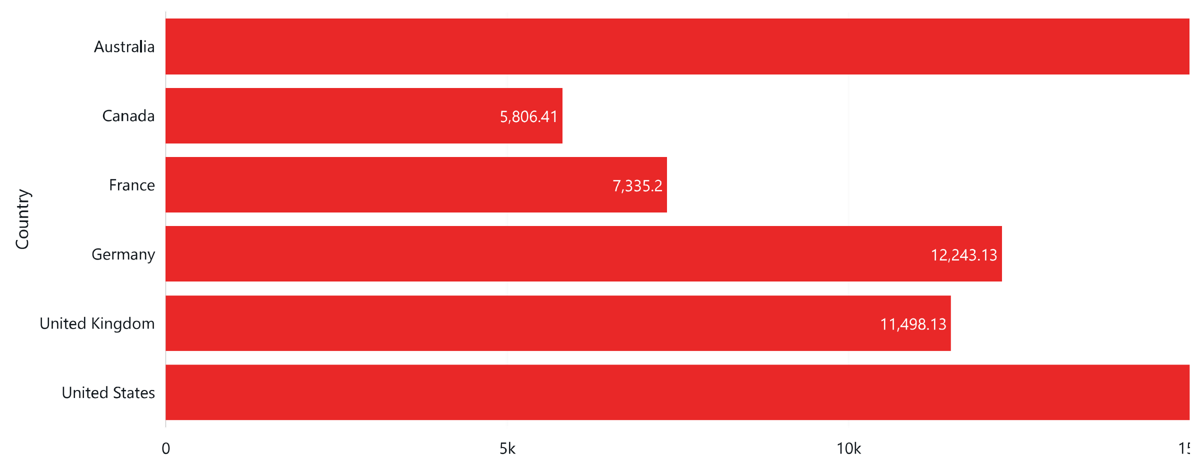
6 rows

Costs of Returned Products

By Country:

```
%sql
select R.Country, ROUND(SUM(P.ProductCost * R.ReturnQuantity),2) AS TotalCostsReturns from salesdb.returns R join
salesdb.products P
on R.ProductKey=P.ProductKey
group by R.Country order by TotalCostsReturns desc;
```

Table Visualization 1

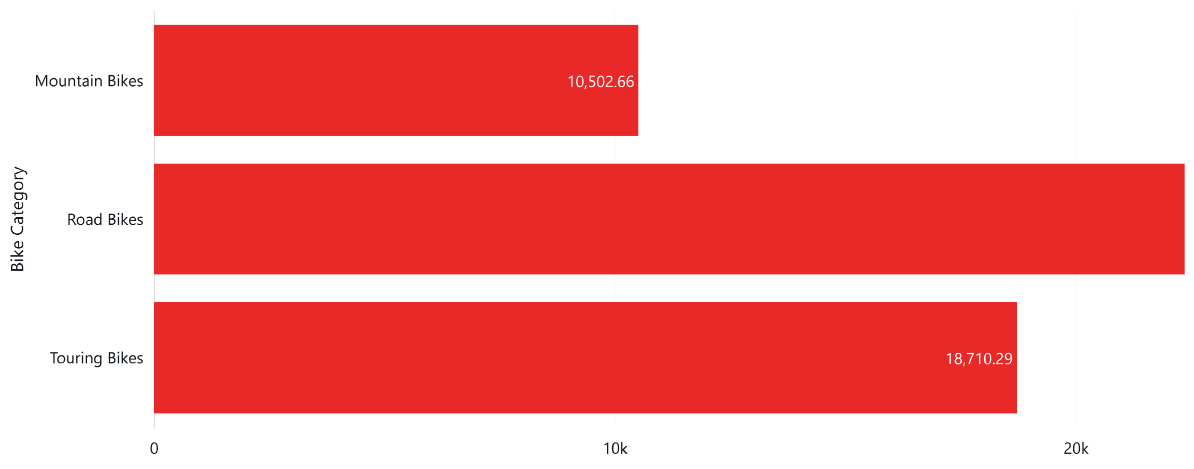


6 rows

By Product category Bikes:

```
%sql
select P.ProductSubcategory, ROUND(SUM(P.ProductCost * R.ReturnQuantity),2) AS TotalCostsReturns from salesdb.returns R join
salesdb.products P
on R.ProductKey=P.ProductKey
where P.ProductSubcategory like '%Bikes%' group by P.ProductSubcategory order by TotalCostsReturns desc;
```

Table | Visualization 1

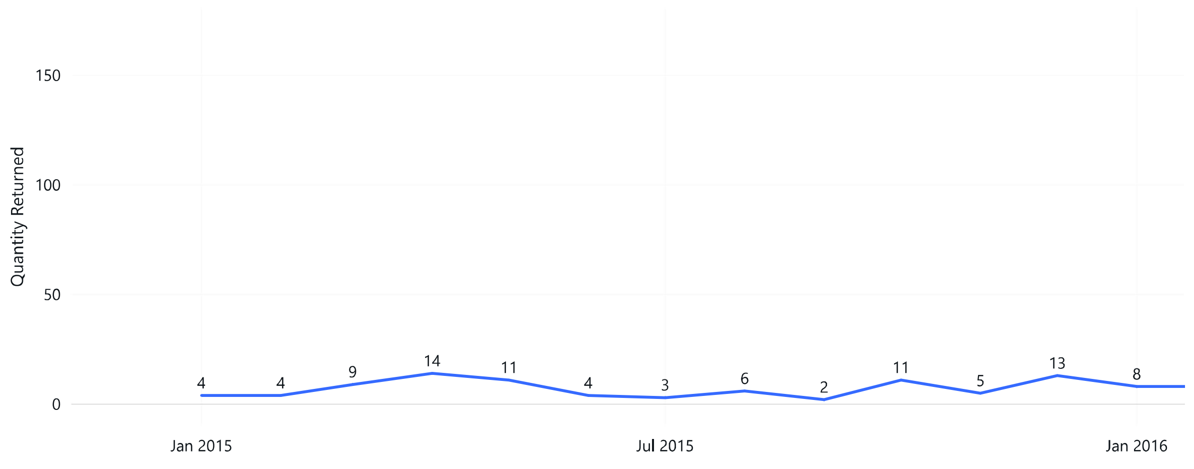


3 rows

Returned Products by Date

```
%sql
select date_trunc('month',ReturnDate) as MONTH, SUM(ReturnQuantity) as quantityreturned from returns group by 1 order by 1 asc
```

Table | Visualization 1



30 rows

End of this Notebook