Engineering and Analyzing Invoices Data

- The files were stored in a zipped format, so the zipfile library was imported into Databricks to unpack the data within
- The 2021 invoices csv was saved as a delta table in DBFS
- Throughout, SQL SELECT * was queried to ensure tables were appropriately stored and/or successfully altered
- Exploration into the data began with a query to calculate the total sales amount per invoice
- Data exploration continued with a query to determine high value customers based on their total spending
- Initial data exploration concluded with the creation of a bar chart to illustrate the total sales amount per invoice; this revealed an average sales amount of just under 500 per invoice, with outliers clearly displayed; an area for future exploration is total sales per invoice over time, which could help determine appropriate stock levels throughout the year; Python's matplotlib library – the pyplot function - was imported for chart creation
- Stored and temporary views of the data were created to view total quantity sold per item (StockCode) and total sales per country, respectively
- A new database was created (database1) to store a transformed version of the delta table (total_sales) that displayed total sales by the Europe and Oceania regions; SQL code to accomplish this included CREATE TABLE IF NOT EXISTS, UPDATE-SET, ALTER TABLE, SET TBLPROPERTIES, and INSERT OVERWRITE TABLE
- Finally, the table was partitioned on the StockCode and TotalQuantitySold (aggregate) columns, and z-ordered on TotalQuantitySold to optimize query performance on these columns
- The primary challenge during this data engineering was partitioning and z-ordering simultaneously, which was overcome by experimenting with different code versions and syntax, as well as making choices about which columns to partition by versus which columns to z-order by

Compress the selected files and load it into the DBS. Unzip batch data into a new DBFS directory.

```
%python
import zipfile
# define the path to the zip file
zip_path = "/dbfs/FileStore/project2dataset.zip"
# extract the files from the zip file
with zipfile.ZipFile(zip_path, "r") as zip_ref:
    zip_ref.extractall("/dbfs/FileStore/project2extracted")
```

Data Extraction and Preprocessing: Use Python to read the data into Spark DataFrames. Convert Spark DataFrame into stored view. Utilize Python and SQL for initial exploration into the data.

```
%python
invoices2021 = spark.read.format('csv').option('header', 'true').load('/FileStore/project2extracted/Project 2 Dataset/dataset_1/invoices_2021.
invoices2021.show(5)
```

```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|InvoiceNo|StockCode|         Description|Quantity|    InvoiceDate|UnitPrice|CustomerID|       Country|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
|   536365|   85123A|WHITE HANGING HEA...|       6|01-01-2021 8.26|     2.55|     17850|United Kingdom|
|   536365|    71053| WHITE METAL LANTERN|       6|01-01-2021 8.26|     3.39|     17850|United Kingdom|
|   536365|   84406B|CREAM CUPID HEART...|       8|01-01-2021 8.26|     2.75|     17850|United Kingdom|
|   536365|   84029G|KNITTED UNION FLA...|       6|01-01-2021 8.26|     3.39|     17850|United Kingdom|
|   536365|   84029E|RED WOOLLY HOTTIE...|       6|01-01-2021 8.26|     3.39|     17850|United Kingdom|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------+
only showing top 5 rows
```

```
%python
invoices2021.write.format("delta").mode("overwrite").saveAsTable("invoices2021")
```

```
describe detail invoices2021
```

Table

	format	id	name	description	location
1	delta	b97c5fa7-6a68-4f71-b1f2-4da185bbcdaf	spark_catalog.default.invoices2021	null	dbfs:/user/hive/warehouse/invoices2021

1 row

```
SELECT * FROM invoices2021
```

Table

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Countr
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-01-2021 8.26	2.55	17850	United
2	536365	71053	WHITE METAL LANTERN	6	01-01-2021 8.26	3.39	17850	United
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-01-2021 8.26	2.75	17850	United
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-01-2021 8.26	3.39	17850	United
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-01-2021 8.26	3.39	17850	United
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	01-01-2021 8.26	7.65	17850	United

254 rows

```
--total sales amount per invoice
SELECT invoiceno, ROUND(SUM(quantity * unitprice), 2) AS totalsalesamount
FROM invoices2021
GROUP BY invoiceno;
```

Table

	invoiceno	totalsalesamount
1	536374	350.4
2	536386	508.2
3	536366	22.2
4	536387	3193.92
5	536385	130.85
6	536375	259.86

27 rows

```
--high value customers based on total spending
SELECT customerid, ROUND(SUM(quantity * unitprice), 2) AS totalspending
FROM invoices2021
GROUP BY customerid
ORDER BY totalspending DESC;
```

Table

	customerid	totalspending
1	16029	3702.12
2	17511	1825.74
3	13408	1024.68
4	12583	801.86
5	17850	725.44
6	18074	489.6

19 rows

```
%python
# create dataframe for the plot below
graph_df = spark.sql("SELECT invoiceno, ROUND(SUM(quantity * unitprice), 2) AS totalsalesamount \
FROM invoices2021 \
GROUP BY invoiceno")
graph_df.show(5)
```

```
+---------+----------------+
|invoiceno|totalsalesamount|
+---------+----------------+
|   536374|           350.4|
|   536386|           508.2|
|   536366|            22.2|
|   536387|         3193.92|
|   536385|          130.85|
+---------+----------------+
only showing top 5 rows
```

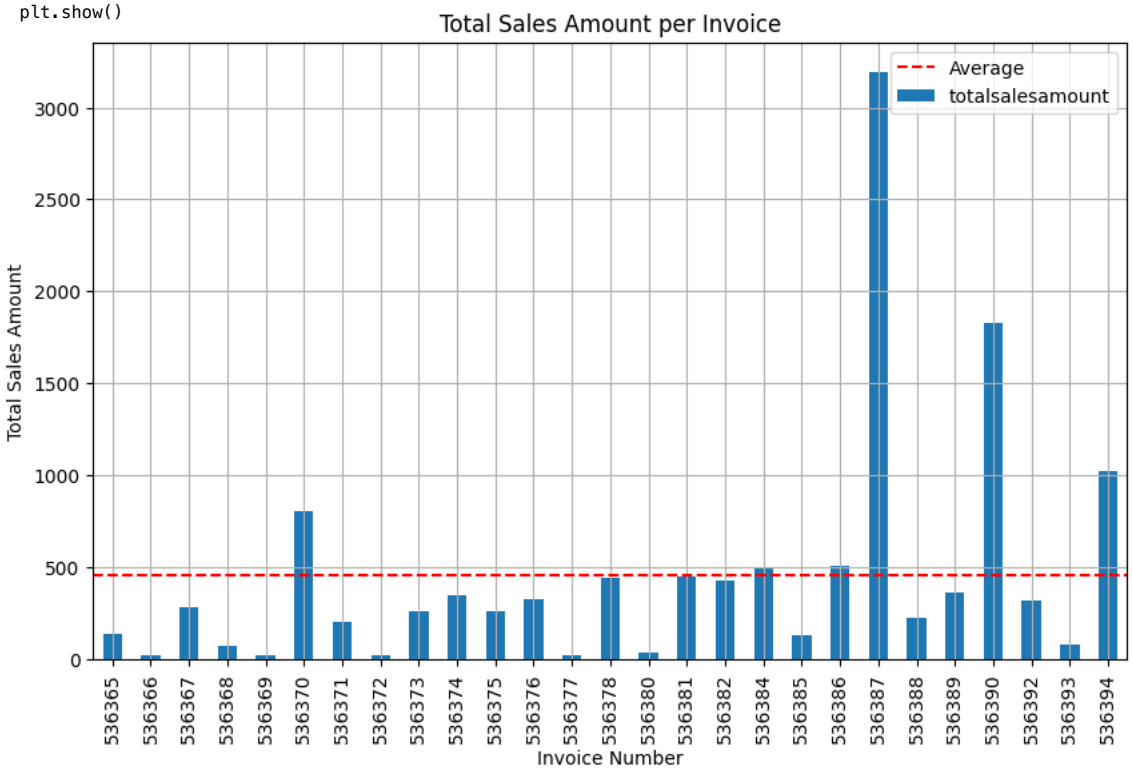
```
%python
import matplotlib.pyplot as plt
import pandas as pd

# convert the DataFrame to Pandas for plotting
invoices2021_pd = graph_df.toPandas()

# group the data by invoiceno and compute the total sales amount per invoice
total_sales_per_invoice = invoices2021_pd.groupby('invoiceno')['totalsalesamount'].sum()

# calculate the average total sales amount
average_sales = total_sales_per_invoice.mean()

# create bar chart
plt.figure(figsize=(10, 6))
total_sales_per_invoice.plot(kind='bar')
plt.axhline(y=average_sales, color='r', linestyle='--', label='Average')  # add the average line
plt.xlabel('Invoice Number')
plt.ylabel('Total Sales Amount')
plt.title('Total Sales Amount per Invoice')
plt.legend()  # show legend with the average line
plt.grid(True)
plt.show()
```



Convert Spark DataFrame into temporary view.

```
%python
invoices2021.createOrReplaceTempView("invoices2021_temp")


SELECT * FROM invoices2021_temp
LIMIT 5;
```

Table

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-01-2021 8.26	2.55	17850	United King
2	536365	71053	WHITE METAL LANTERN	6	01-01-2021 8.26	3.39	17850	United King
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-01-2021 8.26	2.75	17850	United King
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-01-2021 8.26	3.39	17850	United King
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-01-2021 8.26	3.39	17850	United King

5 rows

Use SQL statements to transform and manipulate data within the stored and temporary views.

```
--stored view transformation/manipulation
--calculate the total number of items sold per product
SELECT stockcode, description, sum(quantity) AS totalquantitysold
FROM invoices2021
GROUP BY stockcode, description;
```

Table

	stockcode	description	totalquantitysold
1	10002	INFLATABLE POLITICAL GLOBE	60
2	37444A	YELLOW BREAKFAST CUP AND SAUCER	1
3	22197	SMALL POPCORN HOLDER	100
4	22912	YELLOW COAT RACK PARIS FASHION	3
5	22310	IVORY KNITTED MUG COSY	6
6	84519A	TOMATO CHARLIE+LOLA COASTER SET	6

195 rows

```
--temporary view transformation/manipulation
--calculate the total number of sales by country
SELECT Country, SUM(Quantity) AS TotalSales
FROM invoices2021_temp
GROUP BY Country
ORDER BY TotalSales DESC;
```

Table

	Country	TotalSales
1	United Kingdom	5637
2	France	446
3	Australia	107

3 rows

Database and Delta Table Management: Create a new database/schema. Store the transformed data in Delta tables within the newly created database/schema.

```
CREATE SCHEMA database1
```

OK

```
CREATE TABLE IF NOT EXISTS database1.total_sales
USING DELTA
AS
SELECT Country, SUM(Quantity) AS TotalSales
FROM invoices2021_temp
GROUP BY Country
ORDER BY TotalSales DESC;
```

Query returned no results

```
describe detail database1.total_sales
```

Table

	format	id	name	description	location
1	delta	f7de0fe7-6b83-4e30-a767-b82959cdf04b	spark_catalog.database1.total_sales	null	dbfs:/user/hive/warehouse/database1.db/t

1 row

```
SELECT * FROM database1.total_sales;
```

Table

	Country	TotalSales
1	United Kingdom	5637
2	France	446
3	Australia	107

3 rows

Perform update operation on temporary.

```
UPDATE database1.total_sales
SET Country = 'Europe'
WHERE Country IN ('United Kingdom', 'France');
```

Table

	num_affected_rows
1	2

1 row

```
select * from database1.total_sales
```

Table

	Country	TotalSales
1	Europe	5637
2	Europe	446
3	Australia	107

3 rows

```
-- enable column mapping on delta table with the requested mode
ALTER TABLE database1.total_sales
SET TBLPROPERTIES ('delta.columnMapping.mode' = 'name');

-- rename the column
ALTER TABLE database1.total_sales
RENAME COLUMN Country TO Region;
```

OK

```
SELECT * FROM database1.total_sales
```

Table

	Region	TotalSales
1	Europe	5637
2	Europe	446
3	Australia	107

3 rows

```
-- update the existing "total_sales" table by performing an INSERT OVERWRITE operation with the aggregated results
INSERT OVERWRITE TABLE database1.total_sales
SELECT Region, SUM(TotalSales) AS TotalSales
FROM database1.total_sales
GROUP BY Region;
```

Table

	num_affected_rows	num_inserted_rows
1	2	2

1 row

```
SELECT * FROM database1.total_sales
```

Table

	Region	TotalSales
1	Europe	6083
2	Australia	107

2 rows

```
UPDATE database1.total_sales
SET Region = 'Oceania'
WHERE Region IN ('Australia', 'Polynesia', 'Melanesia');
```

Table

	num_affected_rows
1	1

1 row

```
SELECT * FROM database1.total_sales
ORDER BY TotalSales DESC;
```

Table

	Region	TotalSales
1	Europe	6083
2	Oceania	107

2 rows

Data Update and Optimization: Perform an update operation on one of the delta tables. Partition the table for performance. Choose two columns that are significant for analysis. Implement z-ordering on the delta table to optimize the query performance based on these columns.

```
describe detail invoices2021
```

Table

	format	id	name	description	location
1	delta	b97c5fa7-6a68-4f71-b1f2-4da185bbcdaf	spark_catalog.default.invoices2021	null	dbfs:/user/hive/warehouse/invoices2021

1 row

```
-- INSERT INTO database1.invoices_table
-- SELECT stockcode, description, sum(quantity) as totalquantitysold
-- FROM invoices2021
-- GROUP BY stockcode, description;
```

```
CREATE TABLE database1.invoices_table
USING delta
PARTITIONED BY (StockCode, TotalQuantitySold)
AS (
  SELECT StockCode, Description, sum(Quantity) AS TotalQuantitySold, Country
  FROM invoices2021
  GROUP BY StockCode, Description, Country
);
```

Query returned no results

```
describe detail database1.invoices_table
```

Table

	format	id	name	description	location
1	delta	a8eed39f-2789-4070-9542-446e21278cf7	spark_catalog.database1.invoices_table	null	dbfs:/user/hive/warehouse/database1

1 row

```
%python
# Load your data into DataFrame (example)
df = spark.read.format("delta").load("/user/hive/warehouse/database1.db/invoices_table")

# Create a temporary view from your DataFrame without considering the partitioning.
df.createOrReplaceTempView("temp_view")

# Create or Replace a Delta Table with Partitioning (using IF NOT EXISTS clause)
spark.sql("""
    CREATE TABLE IF NOT EXISTS delta_table
    USING delta
    PARTITIONED BY (StockCode)
    AS SELECT * FROM temp_view
""")

# Apply Z-Ordering using the OPTIMIZE command to the Delta Table
spark.sql("""OPTIMIZE delta_table ZORDER BY (TotalQuantitySold)""")
```

```
DataFrame[path: string, metrics: struct<numFilesAdded:bigint,numFilesRemoved:bigint,filesAdded:struct<min:bigint,max:bigint,avg:double,total
Files:bigint,totalSize:bigint>,filesRemoved:struct<min:bigint,max:bigint,avg:double,totalFiles:bigint,totalSize:bigint>,partitionsOptimized:
bigint,zOrderStats:struct<strategyName:string,inputCubeFiles:struct<num:bigint,size:bigint>,inputOtherFiles:struct<num:bigint,size:bigint>,i
nputNumCubes:bigint,mergedFiles:struct<num:bigint,size:bigint>,numOutputCubes:bigint,mergedNumCubes:bigint>,numBatches:bigint,totalConsidere
dFiles:bigint,totalFilesSkipped:bigint,preserveInsertionOrder:boolean,numFilesSkippedToReduceWriteAmplification:bigint,numBytesSkippedToRedu
ceWriteAmplification:bigint,startTimeMs:bigint,endTimeMs:bigint,totalClusterParallelism:bigint,totalScheduledTasks:bigint,autoCompactParalle
lismStats:struct<maxClusterActiveParallelism:bigint,minClusterActiveParallelism:bigint,maxSessionActiveParallelism:bigint,minSessionActivePa
rallelism:bigint>,deletionVectorStats:struct<numDeletionVectorsRemoved:bigint,numDeletionVectorRowsRemoved:bigint>,numTableColumns:bigint,nu
mTableColumnsWithStats:bigint,totalTaskExecutionTimeMs:bigint,skippedArchivedFiles:bigint,clusteringMetrics:struct<sizeOfTableInBytesBeforeL
azyClustering:bigint,isNewMetadataCreated:boolean,numFilesClassifiedToIntermediateNodes:bigint,sizeOfFilesClassifiedToIntermediateNodesInByt
es:bigint,logicalSizeOfFilesClassifiedToIntermediateNodesInBytes:bigint,numFilesClassifiedToLeafNodes:bigint,sizeOfFilesClassifiedToLeafNode
sInBytes:bigint,logicalSizeOfFilesClassifiedToLeafNodesInBytes:bigint,clusterThresholdStrategy:string,minFileSize:bigint,maxFileSize:bigint,
nodeMinNumFilesToCompact:bigint,numIdealFiles:bigint,numClusteringTasksPlanned:int,numCompactionTasksPlanned:int,numOptimizeBatchesPlanned:i
nt,numLeafNodesExpanded:bigint,numLeafNodesClustered:bigint,numGetFilesForNodeCalls:bigint,numSamplingJobs:bigint,numLeafNodesCompacted:bigi
nt,numIntermediateNodesCompacted:bigint,totalSizeOfDataToCompactInBytes:bigint,totalLogicalSizeOfDataToCompactInBytes:bigint,numIntermediate
NodesClustered:bigint,numFilesSkippedAfterExpansion:bigint,totalSizeOfFilesSkippedAfterExpansionInBytes:bigint,totalLogicalSizeOfFilesSkippe
dAfterExpansionInBytes:bigint,totalSizeOfDataToRewriteInBytes:bigint,totalLogicalSizeOfDataToRewriteInBytes:bigint,timeMetrics:struct<classi
fierTimeMs:bigint,optimizerTimeMs:bigint,metadataLoadTimeMs:bigint,totalGetFilesForNodeCallsTimeMs:bigint,totalSamplingTimeMs:bigint,metadat
aCreationTimeMs:bigint>>>]
```

Table

	StockCode	Description	TotalQuantitySold	Country
1	85183B	CHARLIE & LOLA WASTEPAPER BIN FLORA	48	United Kingdom
2	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	42	United Kingdom
3	21340	CLASSIC METAL BIRDCAGE PLANT HOLDER	2	United Kingdom
4	84755	COLOUR GLASS T-LIGHT HOLDER HANGING	48	United Kingdom
5	21080	SET/20 RED RETROSPOT PAPER NAPKINS	96	United Kingdom
6	22923	FRIDGE MAGNETS LES ENFANTS ASSORTED	12	United Kingdom

202 rows