Data Scientist's Workflow with Python & SQL

- Reading the Data from Google Sheet : Sheet Link
- Basic Exploratory Data Analysis (EDA).
- Data Cleaning.
- Creating a Denormalized Table by Merging/Joining Multple Sheets
- Save the final Consolidated Data in BigQuery.

Note: Please use the left side panel of the Table of Contents to navigate through the sections.

Neccessary Libraries

In [53]:
```python
# Neccessary Libraries

# for storing data into BigQuery
from google.cloud import bigquery
from google.colab import auth

# for authenticate
# auth.authenticate_user()

# initialize the client for Bigquery
project_id = 'tutorial-data-441413'
client = bigquery.Client(project_id, location='US')

# for Cleaning, Analyzing & Charts
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re


from google.auth.transport.requests import Request
from google.oauth2.service_account import Credentials
```

Google Sheet URLs for CSV export

To convert a google gheet file into csv and directly read by pandas here is the structure we need to follow, https://docs.google.com/spreadsheets/d/{spreadsheet_id}/export?format=csv&gid={sheet_gid}

For example;

this https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/edit?gid=1531479241#gid=1531479241

will be converted to;

this https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/export?format=csv&gid=1531479241

Data Explorations

In [54]:
```python
# @title Google Sheet URLs for CSV export

# file path

orders = "https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/export?format=csv&gid=1531479241"
customers = "https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/export?format=csv&gid=2099175586"
returns = "https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/export?format=csv&gid=1158708900"
users = "https://docs.google.com/spreadsheets/d/1M0u00wa4A6CqTA8xImd90nDtF86OwhR2ESgQjUItfaY/export?format=csv&gid=531959115"

# Read directly into Pandas DataFrame
df_orders = pd.read_csv(orders, index_col='Row ID')
df_customers = pd.read_csv(customers)
df_returns = pd.read_csv(returns)
df_users = pd.read_csv(users)
```

In [55]:
```python
# @title Orders

# Display the first few rows
df_orders.head()
```

Out[55]:

Row ID	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Postal Code	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Sh
18606	88525	2	Corporate	Office Supplies	Labels	Small Box	Avery 49	Not Specified	Regular Air	Central	...	60101	5/28/2012	5/30/2012	2	2.88	0.01	0.36	
20847	88522	3	Corporate	Office Supplies	Pens & Art Supplies	Wrap Bag	SANFORD Liquid Accent™ Tank-Style Highlighters	High	Express Air	West	...	98221	7/7/2010	7/8/2010	4	2.84	0.01	0.54	
23086	88523	3	Corporate	Office Supplies	Paper	Small Box	Xerox 1968	Not Specified	Express Air	West	...	98221	7/27/2011	7/28/2011	7	6.68	0.03	0.37	
23087	88523	3	Corporate	Office Supplies	Scissors, Rulers and Trimmers	Small Pack	Acme® Preferred Stainless Steel Scissors	Not Specified	Regular Air	West	...	98221	7/27/2011	7/28/2011	7	5.68	0.01	0.56	
23088	88523	3	Corporate	Technology	Telephones and Communication	Small Box	V70	Not Specified	Express Air	West	...	98221	7/27/2011	7/27/2011	8	205.99	0.00	0.59	

5 rows × 22 columns

In [56]:
```python
# @title Customers

# Display the first few rows
df_customers.head()
```

Out[56]:

	Customer ID	Customer Name
0	2	Janice Fletcher
1	3	Bonnie Potter
2	5	Ronnie Proctor
3	6	Dwight Hwang
4	7	Leon Gill

In [57]:
```python
# @title Returns

# Display the first few rows
df_returns.head()
```

Out[57]:

	Order ID	Status
0	65	Returned
1	612	Returned
2	614	Returned
3	678	Returned
4	710	Returned

In [58]:
```python
# @title Users

# Display the first few rows
df_users.head()
```

Out[58]:

	Region	Manager
0	Central	Chris
1	East	Erin
2	South	Sam
3	West	William

Basic Data Cleaning

In [59]:
```python
# @title Rows & Columns

print("Rows:", df_orders.shape[0])
print("Columns:", df_orders.shape[1])
```

```
Rows: 9427
Columns: 22
```

In [60]:
```python
# @title Dataset Columns

# Let's print the columns (features) names.
df_orders.columns
```

Out[60]:
```
Index(['Order ID', 'Customer ID', 'Customer Segment', 'Product Category',
       'Product Sub-Category', 'Product Container', 'Product Name',
       'Order Priority', 'Ship Mode', 'Region', 'State or Province', 'City',
       'Postal Code', 'Order Date', 'Ship Date', 'Quantity Ordered',
       'Unit Price', 'Discount', 'Product Base Margin', 'Shipping Cost',
       'Sales', 'Profit'],
      dtype='object')
```

In [61]:
```python
# @title Columns Data Type

# Let's print the columns data types.
df_orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9427 entries, 18606 to 24492
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Order ID              9427 non-null   int64
 1   Customer ID           9427 non-null   int64
 2   Customer Segment      9427 non-null   object
 3   Product Category      9427 non-null   object
 4   Product Sub-Category  9427 non-null   object
 5   Product Container     9427 non-null   object
 6   Product Name          9427 non-null   object
 7   Order Priority        9427 non-null   object
 8   Ship Mode             9427 non-null   object
 9   Region                9427 non-null   object
 10  State or Province     9427 non-null   object
 11  City                  9427 non-null   object
 12  Postal Code           9427 non-null   int64
 13  Order Date            9427 non-null   object
 14  Ship Date             9427 non-null   object
 15  Quantity Ordered      9427 non-null   int64
 16  Unit Price            9427 non-null   float64
 17  Discount              9427 non-null   float64
 18  Product Base Margin   9355 non-null   float64
 19  Shipping Cost         9427 non-null   float64
 20  Sales                 9427 non-null   float64
 21  Profit                9427 non-null   float64
dtypes: float64(6), int64(4), object(12)
memory usage: 1.7+ MB
```

In [62]:
```python
# @title Columns Data Type Transformation

# Let's try to change the datatypes of the following column in the dataset.
df_orders['Order Date'] = df_orders['Order Date'].astype('datetime64[ns]')
df_orders['Ship Date'] = df_orders['Ship Date'].astype('datetime64[ns]')
df_orders['Postal Code'] = df_orders['Postal Code'].astype('object')


# Let's print the columns data types.
df_orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9427 entries, 18606 to 24492
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Order ID              9427 non-null   int64
 1   Customer ID           9427 non-null   int64
 2   Customer Segment      9427 non-null   object
 3   Product Category      9427 non-null   object
 4   Product Sub-Category  9427 non-null   object
 5   Product Container     9427 non-null   object
 6   Product Name          9427 non-null   object
 7   Order Priority        9427 non-null   object
 8   Ship Mode             9427 non-null   object
 9   Region                9427 non-null   object
 10  State or Province     9427 non-null   object
 11  City                  9427 non-null   object
 12  Postal Code           9427 non-null   object
 13  Order Date            9427 non-null   datetime64[ns]
 14  Ship Date             9427 non-null   datetime64[ns]
 15  Quantity Ordered      9427 non-null   int64
 16  Unit Price            9427 non-null   float64
 17  Discount              9427 non-null   float64
 18  Product Base Margin   9355 non-null   float64
 19  Shipping Cost         9427 non-null   float64
 20  Sales                 9427 non-null   float64
 21  Profit                9427 non-null   float64
dtypes: datetime64[ns](2), float64(6), int64(3), object(11)
memory usage: 1.7+ MB
```

In [63]:
```python
# @title Summary Statistics

# Describing statistical information on the dataset
df_orders.describe().round(2)
```

Out[63]:

	Order ID	Customer ID	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Profit
count	9427.00	9427.00	9427	9427	9427.00	9427.00	9427.00	9355.00	9427.00	9427.00	9427.00
mean	82319.33	1738.24	2012-03-05 19:19:32.759096064	2012-03-07 20:01:51.509494016	13.80	88.30	0.05	0.51	12.79	949.60	139.22
min	6.00	2.00	2010-01-01 00:00:00	2010-01-02 00:00:00	1.00	0.99	0.00	0.35	0.49	1.32	-16476.84
25%	86737.50	898.00	2011-03-07 12:00:00	2011-03-09 00:00:00	5.00	6.48	0.02	0.38	3.22	61.10	-74.00
50%	88345.00	1750.00	2012-04-08 00:00:00	2012-04-09 00:00:00	10.00	20.99	0.05	0.52	6.05	203.42	2.54
75%	89988.50	2578.50	2013-03-26 00:00:00	2013-03-28 00:00:00	17.00	85.99	0.08	0.59	13.99	776.36	140.21
max	91591.00	3403.00	2013-12-31 00:00:00	2014-01-17 00:00:00	170.00	6783.02	0.25	0.85	164.73	100119.16	16332.41
std	19148.61	979.28	NaN	NaN	15.11	281.53	0.03	0.14	17.18	2597.90	998.43

In [64]:
```python
# Describing more statistical information on the dataset
# Describing statistical information on the numerical columns only
df_orders.describe(include=[np.number]).round(2)
```

Out[64]:

	Order ID	Customer ID	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Profit
count	9427.00	9427.00	9427.00	9427.00	9427.00	9355.00	9427.00	9427.00	9427.00
mean	82319.33	1738.24	13.80	88.30	0.05	0.51	12.79	949.60	139.22
std	19148.61	979.28	15.11	281.53	0.03	0.14	17.18	2597.90	998.43
min	6.00	2.00	1.00	0.99	0.00	0.35	0.49	1.32	-16476.84
25%	86737.50	898.00	5.00	6.48	0.02	0.38	3.22	61.10	-74.00
50%	88345.00	1750.00	10.00	20.99	0.05	0.52	6.05	203.42	2.54
75%	89988.50	2578.50	17.00	85.99	0.08	0.59	13.99	776.36	140.21
max	91591.00	3403.00	170.00	6783.02	0.25	0.85	164.73	100119.16	16332.41

In [65]:
```python
# prompt: But, I want the above descriptive statistics without the Order ID and Customer ID.
# Make the code simpler

# Describing statistical information on the numerical columns only, excluding specified columns
df_orders.drop(columns=['Order ID', 'Customer ID']).describe(include=[np.number]).round(2)
```

Out[65]:

	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Profit
count	9427.00	9427.00	9427.00	9355.00	9427.00	9427.00	9427.00
mean	13.80	88.30	0.05	0.51	12.79	949.60	139.22
std	15.11	281.53	0.03	0.14	17.18	2597.90	998.43
min	1.00	0.99	0.00	0.35	0.49	1.32	-16476.84
25%	5.00	6.48	0.02	0.38	3.22	61.10	-74.00
50%	10.00	20.99	0.05	0.52	6.05	203.42	2.54
75%	17.00	85.99	0.08	0.59	13.99	776.36	140.21
max	170.00	6783.02	0.25	0.85	164.73	100119.16	16332.41

In [66]:
```python
# @title Exporting the modified Dataset

df_orders.to_csv('df_orders_exported.csv')
# index =False)
```

Data Cleaning

In [67]:
```python
# @title Reading Data
# Let's try to read from the new order dataset
df_cleaned = pd.read_csv('/content/df_orders_exported.csv', index_col='Row ID')
df_cleaned.head()
```

Out[67]:

Row ID	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Postal Code	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost
18606	88525	2	Corporate	Office Supplies	Labels	Small Box	Avery 49	Not Specified	Regular Air	Central	...	60101	2012-05-28	2012-05-30	2	2.88	0.01	0.36	0.50
20847	88522	3	Corporate	Office Supplies	Pens & Art Supplies	Wrap Bag	SANFORD Liquid Accent™ Tank-Style Highlighters	High	Express Air	West	...	98221	2010-07-07	2010-07-08	4	2.84	0.01	0.54	0.93
23086	88523	3	Corporate	Office Supplies	Paper	Small Box	Xerox 1968	Not Specified	Express Air	West	...	98221	2011-07-27	2011-07-28	7	6.68	0.03	0.37	6.15
23087	88523	3	Corporate	Office Supplies	Scissors, Rulers and Trimmers	Small Pack	Acme® Preferred Stainless Steel Scissors	Not Specified	Regular Air	West	...	98221	2011-07-27	2011-07-28	7	5.68	0.01	0.56	3.60
23088	88523	3	Corporate	Technology	Telephones and Communication	Small Box	V70	Not Specified	Express Air	West	...	98221	2011-07-27	2011-07-27	8	205.99	0.00	0.59	2.50

5 rows × 22 columns

In [68]:
```python
# @title Duplicate Checking

df_cleaned.duplicated().sum()
```

Out[68]: 1

In [69]:
```python
# Print the duplicated rows
df_cleaned[df_cleaned.duplicated()].head()
```

Out[69]:

Row ID	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Postal Code	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Prof
24563	90200	6	Home Office	Office Supplies	Paper	Small Box	Xerox 21	Critical	Regular Air	West	...	95123	2012-12-29	2012-12-31	4	6.48	0.07	0.37	6.6	28.61	-13.8

1 rows × 22 columns

In [70]:
```python
# Display records where 'Row ID' is 24563
df_cleaned.loc[[24563]]
```

Out[70]:

Row ID	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Postal Code	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Prof
24563	90200	6	Home Office	Office Supplies	Paper	Small Box	Xerox 21	Critical	Regular Air	West	...	95123	2012-12-29	2012-12-31	4	6.48	0.07	0.37	6.6	28.61	-13.8
24563	90200	6	Home Office	Office Supplies	Paper	Small Box	Xerox 21	Critical	Regular Air	West	...	95123	2012-12-29	2012-12-31	4	6.48	0.07	0.37	6.6	28.61	-13.8

2 rows × 22 columns

In [71]:
```python
# @title Removing Duplicate

df_cleaned.drop_duplicates(inplace=True)
```

In [72]:
```python
df_cleaned.duplicated().sum()
```

Out[72]: 0

In [73]:
```python
# Let's Display records where 'Row ID' is 24563
df_cleaned.loc[[24563]]
```

Out[73]:

Row ID	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Postal Code	Order Date	Ship Date	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Prof
24563	90200	6	Home Office	Office Supplies	Paper	Small Box	Xerox 21	Critical	Regular Air	West	...	95123	2012-12-29	2012-12-31	4	6.48	0.07	0.37	6.6	28.61	-13.8

1 rows × 22 columns

Merge & Consolidated Data

In [74]:
```python
# @title Merge with Customers

# Merge orders with customers on 'Customer ID'
df_consolidated = pd.merge(df_cleaned, df_customers, on='Customer ID', how='left')
```

In [75]:
```python
# @title Merge with Regions

# Assuming there is a 'Region' column in df_orders to join with df_users
df_consolidated = pd.merge(df_consolidated, df_users, left_on='Region', right_on='Region', how='left')
```

In [76]:
```python
# @title Merge with Returns

# Merge the result with returns on 'Order ID' to include order status
df_consolidated = pd.merge(df_consolidated, df_returns, on='Order ID', how='left')
```

In [77]:
```python
# @title Consolidated Data

# The final df_consolidated will contain merged data
df_consolidated.head()
```

Out[77]:

	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Profit	Custor Na
0	88525	2	Corporate	Office Supplies	Labels	Small Box	Avery 49	Not Specified	Regular Air	Central	...	2	2.88	0.01	0.36	0.50	5.90	1.3200	Jar Fletc
1	88522	3	Corporate	Office Supplies	Pens & Art Supplies	Wrap Bag	SANFORD Liquid Accent™ Tank-Style Highlighters	High	Express Air	West	...	4	2.84	0.01	0.54	0.93	13.01	4.5600	Bon Po
2	88523	3	Corporate	Office Supplies	Paper	Small Box	Xerox 1968	Not Specified	Express Air	West	...	7	6.68	0.03	0.37	6.15	49.92	-47.6400	Bon Po
3	88523	3	Corporate	Office Supplies	Scissors, Rulers and Trimmers	Small Pack	Acme® Preferred Stainless Steel Scissors	Not Specified	Regular Air	West	...	7	5.68	0.01	0.56	3.60	41.64	-30.5100	Bon Po
4	88523	3	Corporate	Technology	Telephones and Communication	Small Box	V70	Not Specified	Express Air	West	...	8	205.99	0.00	0.59	2.50	1446.67	998.2023	Bon Po

5 rows × 25 columns

In [78]:
```python
# @title EDA of Consolidated Data

df_consolidated['Status'].value_counts(dropna=False)
```

Out[78]:

Status	count
NaN	9328
Returned	98

dtype: int64

Lets Fill the NaN as 'Order Complete'

In [79]:
```python
df_consolidated['Status'] = df_consolidated['Status'].fillna('Order Not Returned')
```

In [80]:
```python
df_consolidated['Status'].value_counts(dropna=False)
```

Out[80]:

Status	count
Order Not Returned	9328
Returned	98

dtype: int64

Store the Data in BigQuery

In [81]:
```python
# @title BigQuery

df_consolidated.to_gbq('ecommerce_data.superstore_sales_denormalized_table',
                       project_id,
                       chunksize=None,
                       if_exists='replace'
                       )
```

```
<ipython-input-81-c613fdb31fd2>:3: FutureWarning: Starting with pandas version 3.0 all arguments of to_gbq except for the argument 'destination_table' will be keyword-only.
  df_consolidated.to_gbq('ecommerce_data.superstore_sales_denormalized_table',
<ipython-input-81-c613fdb31fd2>:3: FutureWarning: to_gbq is deprecated and will be removed in a future version. Please use pandas_gbq.to_gbq instead: https://pandas-gbq.readthedocs.io/en/latest/api.html#pandas_gbq.to_gbq
  df_consolidated.to_gbq('ecommerce_data.superstore_sales_denormalized_table',
100%|██████████| 1/1 [00:00<00:00, 8630.26it/s]
```

Reading Back the Data into Python from BigQuery

In [83]:
```python
# If you ever want to read the data back in Python from that Bigquery Table:

from google.cloud import bigquery

# Construct a BigQuery client object.
project_id = 'tutorial-data-441413'
client = bigquery.Client(project_id, location='US')

# Dataset ID and Table ID is required
dataset_id = 'ecommerce_data'
table_id = 'superstore_sales_denormalized_table'
table_ref = f"{project_id}.{dataset_id}.{table_id}"

# Perform a query.
QUERY = (
    f'SELECT * FROM `{table_ref}` LIMIT 100'
)
query_job = client.query(QUERY)  # API request
rows = query_job.result()  # Waits for query to finish

# Convert the results to a pandas DataFrame
df = rows.to_dataframe()
df
```

Out[83]:

	Order ID	Customer ID	Customer Segment	Product Category	Product Sub-Category	Product Container	Product Name	Order Priority	Ship Mode	Region	...	Quantity Ordered	Unit Price	Discount	Product Base Margin	Shipping Cost	Sales	Pr
0	88525	2	Corporate	Office Supplies	Labels	Small Box	Avery 49	Not Specified	Regular Air	Central	...	2	2.88	0.01	0.36	0.50	5.90	1.32
1	88522	3	Corporate	Office Supplies	Pens & Art Supplies	Wrap Bag	SANFORD Liquid Accent™ Tank-Style Highlighters	High	Express Air	West	...	4	2.84	0.01	0.54	0.93	13.01	4.56
2	88523	3	Corporate	Office Supplies	Paper	Small Box	Xerox 1968	Not Specified	Express Air	West	...	7	6.68	0.03	0.37	6.15	49.92	-47.64
3	88523	3	Corporate	Office Supplies	Scissors, Rulers and Trimmers	Small Pack	Acme® Preferred Stainless Steel Scissors	Not Specified	Regular Air	West	...	7	5.68	0.01	0.56	3.60	41.64	-30.51
4	88523	3	Corporate	Technology	Telephones and Communication	Small Box	V70	Not Specified	Express Air	West	...	8	205.99	0.00	0.59	2.50	1446.67	998.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	89208	33	Corporate	Office Supplies	Paper	Small Box	Staples Copy Paper (20Lb. and 84 Bright)	Critical	Regular Air	West	...	9	4.98	0.05	0.38	4.70	46.63	-14.99
96	91256	35	Small Business	Technology	Office Machines	Large Box	Polycom ViewStation™ ISDN Videoconferencing Unit	Not Specified	Regular Air	West	...	3	6783.02	0.00	0.39	24.49	21366.51	2396.87
97	91257	35	Small Business	Technology	Telephones and Communication	Small Box	StarTAC 7760	Low	Regular Air	West	...	7	65.99	0.07	0.59	3.99	394.37	165.50
98	91256	36	Small Business	Office Supplies	Paper	Wrap Bag	Wirebound Message Books, Four 2 3/4" x 5" Form...	Not Specified	Express Air	West	...	29	9.27	0.02	0.38	4.39	272.98	71.43
99	91256	36	Small Business	Office Supplies	Paper	Small Box	Xerox 1968	Not Specified	Regular Air	West	...	10	6.68	0.03	0.37	6.15	73.81	5.65

100 rows × 25 columns

In [ ]:
```python
# prompt: Can you run a stored procedures in BigQuery to remove the records where profit is negative?

from google.cloud import bigquery

def remove_negative_profit(project_id, dataset_id, table_id):
    """Removes records with negative profit from a BigQuery table using a stored procedure.

    Args:
        project_id: Your Google Cloud project ID.
        dataset_id: The ID of the BigQuery dataset containing the table.
        table_id: The ID of the BigQuery table to modify.
    """

    client = bigquery.Client(project=project_id, location='US')

    # Define the SQL query for the stored procedure
    query = f"""
        CREATE OR REPLACE PROCEDURE `{project_id}.{dataset_id}.remove_negative_profits`(
            IN table_name STRING
        )
        BEGIN
          DELETE FROM `{project_id}.{dataset_id}.{table_id}`
          WHERE Profit < 0;
        END;
    """

    # Execute the query to create the stored procedure
    query_job = client.query(query)
    query_job.result()  # Wait for the query to complete

    # Call the stored procedure to remove records
    procedure_ref = f"{project_id}.{dataset_id}.remove_negative_profits"
    procedure_job_config = bigquery.QueryJobConfig(
        query_parameters=[
            bigquery.ScalarQueryParameter("table_name", "STRING", table_id),
        ]
    )
    procedure_job = client.query(
        f"CALL `{procedure_ref}`('{table_id}')", job_config=procedure_job_config
    )
    procedure_job.result()


# Example usage (replace with your project ID, dataset ID, and table ID)
project_id = 'tutorial-data-441413'
dataset_id = 'ecommerce_data'
table_id = 'superstore_sales_denormalized_table'

remove_negative_profit(project_id, dataset_id, table_id)

print(f"Records with negative profit removed from table {table_id}")
```