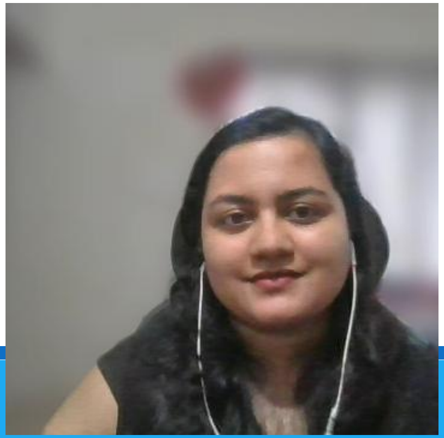
[AAI-500-IN1] Group 3 Presentation

E-commerce Sales Insights & Strategic Recommendations

Objective

Understanding e-commerce sales and customer behaviour through statistical analysis



Dataset

Source: E-commerce transaction dataset (UK Retail)

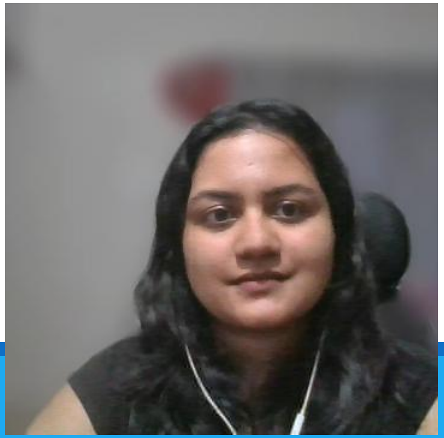
Time Period: December 2018 to December 2019

Number of Entries: ~500,000 transactions

Features:

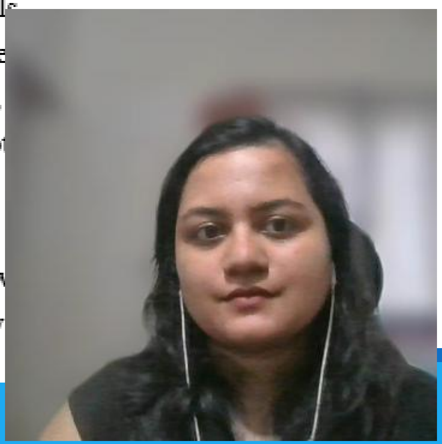
- TransactionNo
- Date
- ProductNo
- Product
- Price
- Quantity
- CustomerNo
- Country

Goal: Identify patterns, trends, and actionable insights in the sales data

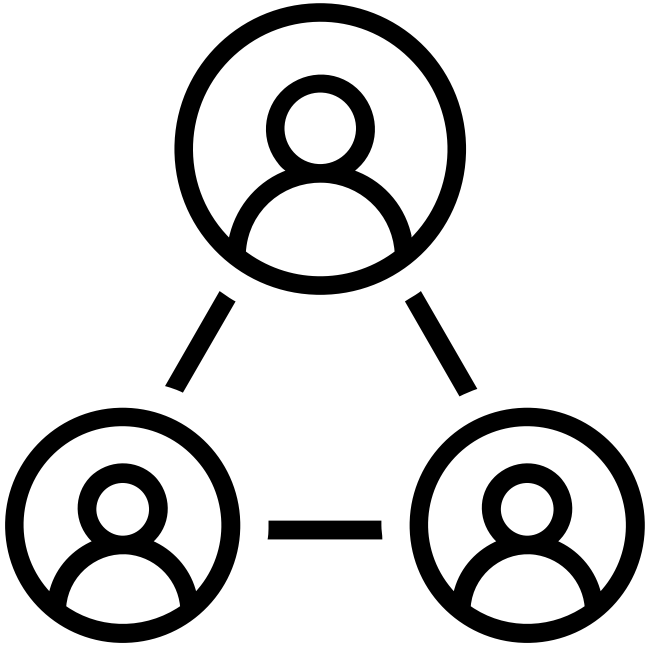


Project Structure

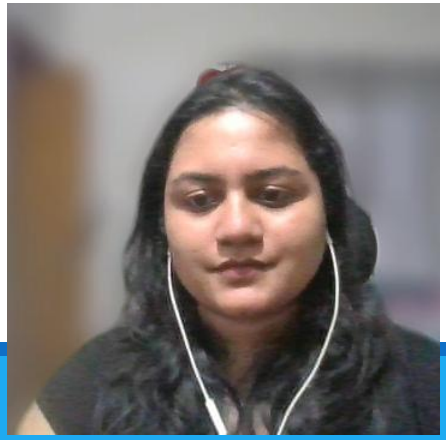
```
ecommerce-sales-project/
│
├── 📁 data/
│   ├── raw/                          # Original dataset from Kaggle
│   ├── processed/                    # Cleaned & transformed datasets
│   └── data_dictionary.md            # Notes about variables and schema
├── 📁 notebooks/
│   ├── 01_data_cleaning.ipynb        # Data wrangling and missing value handling
│   ├── 02_eda_part1.ipynb            # Visualizations, outlier detection, stats
│   ├── 02_eda_part1.ipynb            # Visualizations, outlier detection, stats
│   ├── 03_modeling.ipynb             # Modeling and evaluation
├── 📁 visuals/
  ├── data_cleansing_charts/          # Histogram, boxplot, pie chart, etc.
│   ├── eda_charts/                   # Histogram, boxplot, pie chart, etc.
│   └── model_charts/                 # Forecasts etc.
├── 📁 reports/
│   ├── Final-Project-Report-Group3.pdf  # Final technical report
├── 📁 presentation/
│   ├── Final-Project-Presentation-Group3.mp4
│   └── slides.pptx                   # Slide deck used in video
├── 📁 meta/
│   ├── team_contacts.txt             # Team contact details
│   ├── meeting_notes.md              # Weekly sync-up meeting note
│   ├── role_assignment.md            # Roles: prep, EDA, modeling,
│   └── ai_usage_notes.md             # Notes on ChatGPT or Copilo
│
├── requirements.txt                  # Libraries used
├── README.md                         # Project intro & how
└── .gitignore                        # Ignore unnecessary
```



Role Distribution



- Yogesh Sangwikar
  - Data Cleaning & Preparation
- Meghesh Saini
  - EDA part 1
- Aishwarya Gulhane
  - EDA Part 2 ,Modeling & Reporting



Data Cleaning:

File reading and preliminary observations

Yogesh

- The first part was to load and read the data file, using pandas function in Python
- The first few and last few records were as under:

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
0	581482	12/9/2019	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom
1	581475	12/9/2019	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom
2	581475	12/9/2019	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom
3	581475	12/9/2019	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom
4	581475	12/9/2019	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom

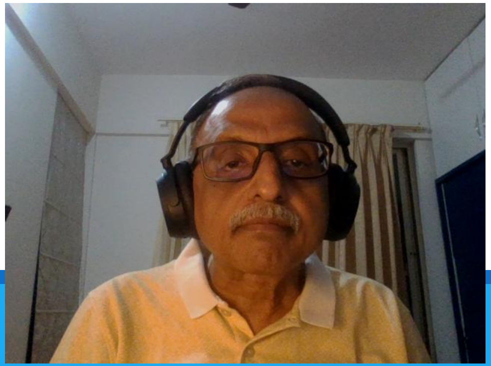
```
sales_data.tail(5)
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
536345	C536548	12/1/2018	22168	Organiser Wood Antique White	18.96	-2	12472.0	Germany
536346	C536548	12/1/2018	21218	Red Spotty Biscuit Tin	14.09	-3	12472.0	Germany
536347	C536548	12/1/2018	20957	Porcelain Hanging Bell Small	11.74	-1	12472.0	Germany
536348	C536548	12/1/2018	22580	Advent Calendar Gingham Sack	16.35	-4	12472.0	Germany
536349	C536548	12/1/2018	22767	Triple Photo Frame Cornice	20.45	-2	12472.0	Germany

This raw data without any cleaning activity has 536350 rows and 8 columns

Observations:

- Few values in Quantity column seem to have negative values !
- On reading about this data , we got to know that negative quantity values are corresponding to the sales cancellation !
- This can also be identified, as we see a "C" alphabet in front of the transaction no data
- These peculiarities of data need to be considered while processing!



Data Cleaning:
Understanding Datatypes

Yogesh

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 536350 entries, 0 to 536349
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TransactionNo  536350 non-null  object
 1   Date           536350 non-null  object
 2   ProductNo      536350 non-null  object
 3   ProductName    536350 non-null  object
 4   Price          536350 non-null  float64
 5   Quantity       536350 non-null  int64
 6   CustomerNo     536295 non-null  float64
 7   Country        536350 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 32.7+ MB
```

- **Observations:**

- ProductNo and TransactionNo columns have been kept as String/ object data, we need to find out why ?
- The date column is object type, which is incorrect , it needs to be changed to date type !
- The customerNo column is float, which can be changed to integer , as there are no need of any decimal places
- The Quantity is in integers , so there is no qty in fractions

```
Yes, there are non-numeric values in the 'TransactionNo' column.
Yes, there are non-numeric values in the 'ProductNo' column.
```

Further exploration revealed that the Product No and Transaction No (though supposed to be numeric), contained Non numeric values , which justifies it to be an ”object” type.

Data Cleaning:
Checking Null values and Duplicate values

Yogesh

On checking , it was observed that there were no NULL values , in any column !

However when checked for DUPLICATE records , we did get 5,200 duplicate records !

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
985	581497	2019-12-09	21481	Fawn Blue Hot Water Bottle	7.24	1	17497.0	United Kingdom
1365	581538	2019-12-09	23275	Set Of 3 Hanging Owls Ollie Beak	6.19	1	14446.0	United Kingdom
1401	581538	2019-12-09	22992	Revolver Wooden Ruler	6.19	1	14446.0	United Kingdom
1406	581538	2019-12-09	22694	Wicker Star	6.19	1	14446.0	United Kingdom
1409	581538	2019-12-09	23343	Jumbo Bag Vintage Christmas	6.19	1	14446.0	United Kingdom
...	...	...	...	...	...	...	...	...
535227	536559	2018-12-01	51014L	Feather Pen Light Pink	11.12	12	17873.0	United Kingdom
535310	536569	2018-12-01	22111	Scottie Dog Hot Water Bottle	15.32	1	16274.0	United Kingdom
535327	536569	2018-12-01	21809	Christmas Hanging Tree With Bell	11.53	1	16274.0	United Kingdom
535960	536592	2018-12-01	82613A	Metal Sign Cupcake Single Hook	12.82	1	16592.0	United Kingdom
536190	536528	2018-12-01	22839	3 Tier Cake Tin Green And Cream	25.57	1	15525.0	United Kingdom

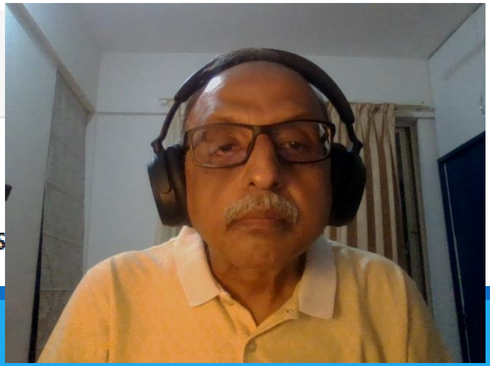
5200 rows × 8 columns

All these duplicate records were deleted and confirmed , and the total records came down to 531,095 !

```
# Drop duplicate rows and keep the first occurrence
sales_data = sales_data.drop_duplicates()
sales_data.shape
```

```
(531095, 8)
```

We can see that now the records have come down to 531095 (after deleting 5200 rows from earlier file which had 536,295 records



Yogesh

Data Cleaning:
Feature Engineering

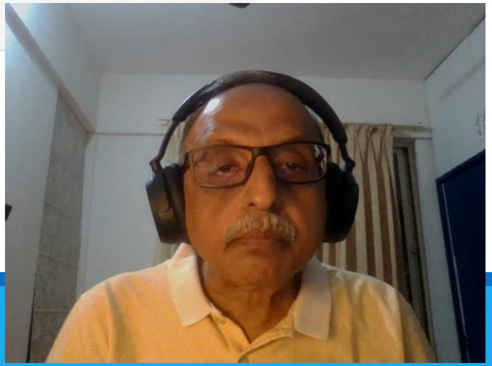
Created New columns for Year, Month, Week and Weekdays and the data now looks like this :

```
# Assuming 'Date' is already a datetime column, add year month and week columns
sales_data['Year'] = sales_data['Date'].dt.year
sales_data['Month'] = sales_data['Date'].dt.month
sales_data['Week'] = sales_data['Date'].dt.isocalendar().week
sales_data['weekday'] = sales_data['Date'].dt.day_name()
sales_data.head()
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country	Year	Month	Week	weekday
0	581482	2019-12-09	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom	2019	12	50	Monday
1	581475	2019-12-09	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom	2019	12	50	Monday
2	581475	2019-12-09	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom	2019	12	50	Monday
3	581475	2019-12-09	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom	2019	12	50	Monday
4	581475	2019-12-09	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom	2019	12	50	Monday

```
sales_data.shape
```

(531095, 12)

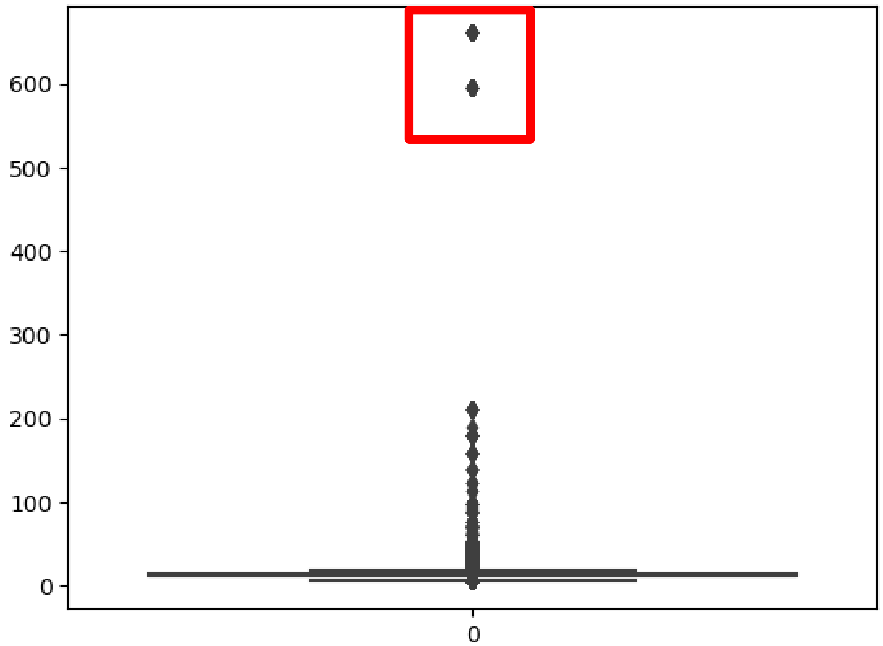


This time period data was derived from Date column, after which the columns increased from 8 to 12

Data Cleaning:
Finding Outliers

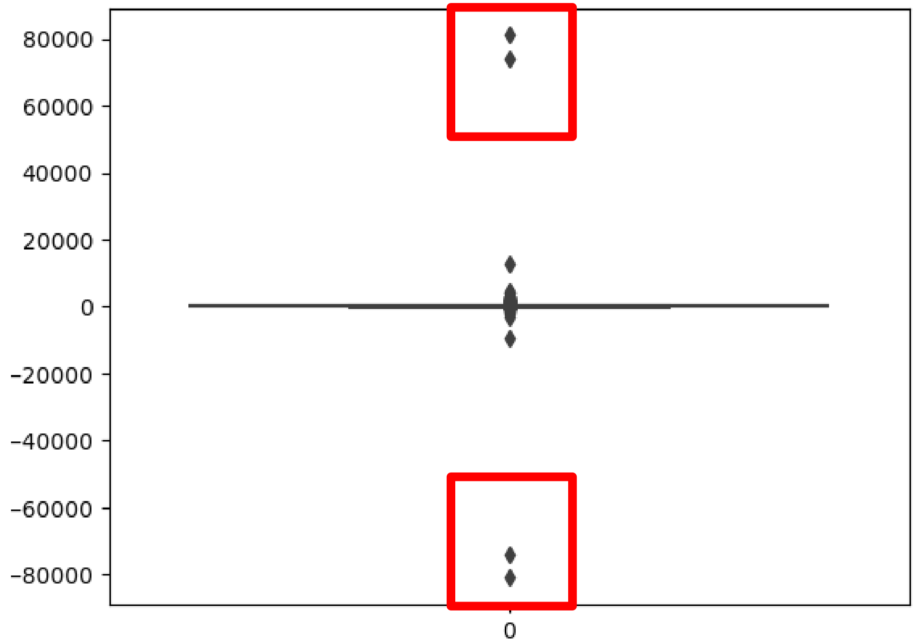
The numerical columns , namely the price and Quantity were checked for presence of any OUTLIER figures and the BOX plots were plotted as under :

Price

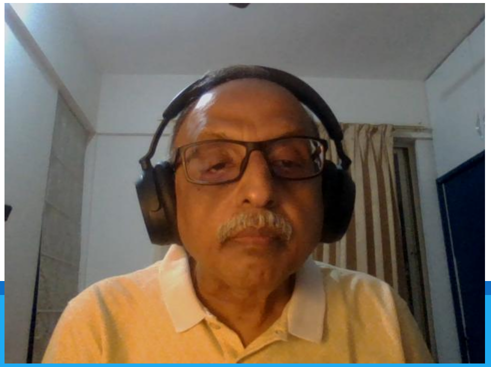


<Axes: >

Quantity



- There are a few outliers on price - higher side and on Quantity on each side
- Even though there quite a few outliers in price and quantity , we have taken a call to retain them as it is, because they are not erroneous and may change the statistics and total amount

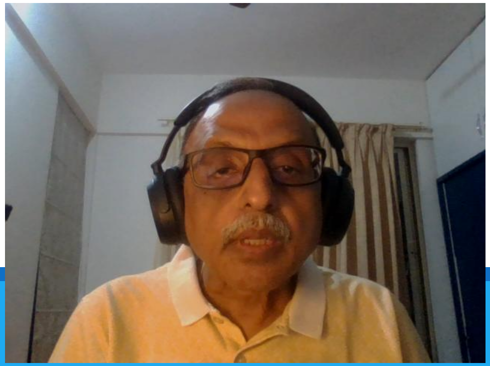


Data Cleaning:
Basic Statistical analysis

Yogesh

```
##### Confirm using descibe also
sales_data.describe()
```

	Date	Price	Quantity	CustomerNo	Year	Month	Week
count	531095	531095.000000	531095.000000	531095.000000	531095.000000	531095.000000	531095.0
mean	2019-07-04 00:29:26.409399296	12.669635	9.993146	15222.612241	2018.921743	7.552238	30.995059
min	2018-12-01 00:00:00	5.130000	-80995.000000	12004.000000	2018.000000	1.000000	1.0
25%	2019-03-28 00:00:00	10.990000	1.000000	13798.000000	2019.000000	5.000000	19.0
50%	2019-07-20 00:00:00	11.940000	3.000000	15146.000000	2019.000000	8.000000	34.0
75%	2019-10-19 00:00:00	14.090000	10.000000	16727.000000	2019.000000	11.000000	45.0
max	2019-12-09 00:00:00	660.620000	80995.000000	18287.000000	2019.000000	12.000000	51.0
std	NaN	8.526181	217.710261	1716.633588	0.268576	3.508959	15.163434



Yogesh

Data Cleaning:
Adding “Regions” to the datafile

The corresponding “Regions” for each country were identified in another data file and this file was merged to get regions information in a new column.

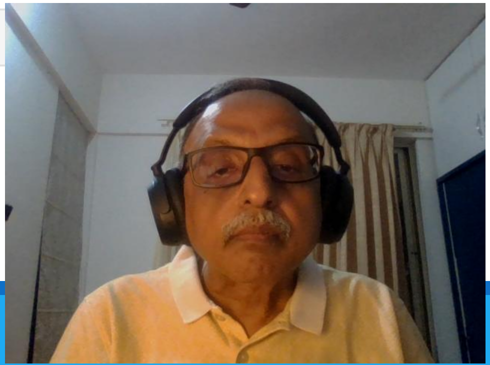
```
# all these countries belong to some regions, these regions are listed in anothe data file "Country_to_Region_Mapping.csv"
# we decided to add regional information to this database by merging these two files
country_region_df = pd.read_csv(r"C:\Users\yogsa\OneDrive\Documents\GitHub\AAI-500-IN1_PROJECT\data\Raw\Country_to_Region_Mapping.csv")
sales_data_with_region = sales_data.merge(country_region_df, on="Country", how="left")
```

```
sales_data_with_region.head()
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country	Year	Month	Week	weekday	Region
0	581482	2019-12-09	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom	2019	12	50	Monday	Europe
1	581475	2019-12-09	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom	2019	12	50	Monday	Europe
2	581475	2019-12-09	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom	2019	12	50	Monday	Europe
3	581475	2019-12-09	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom	2019	12	50	Monday	Europe
4	581475	2019-12-09	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom	2019	12	50	Monday	Europe

```
sales_data_with_region.shape
```

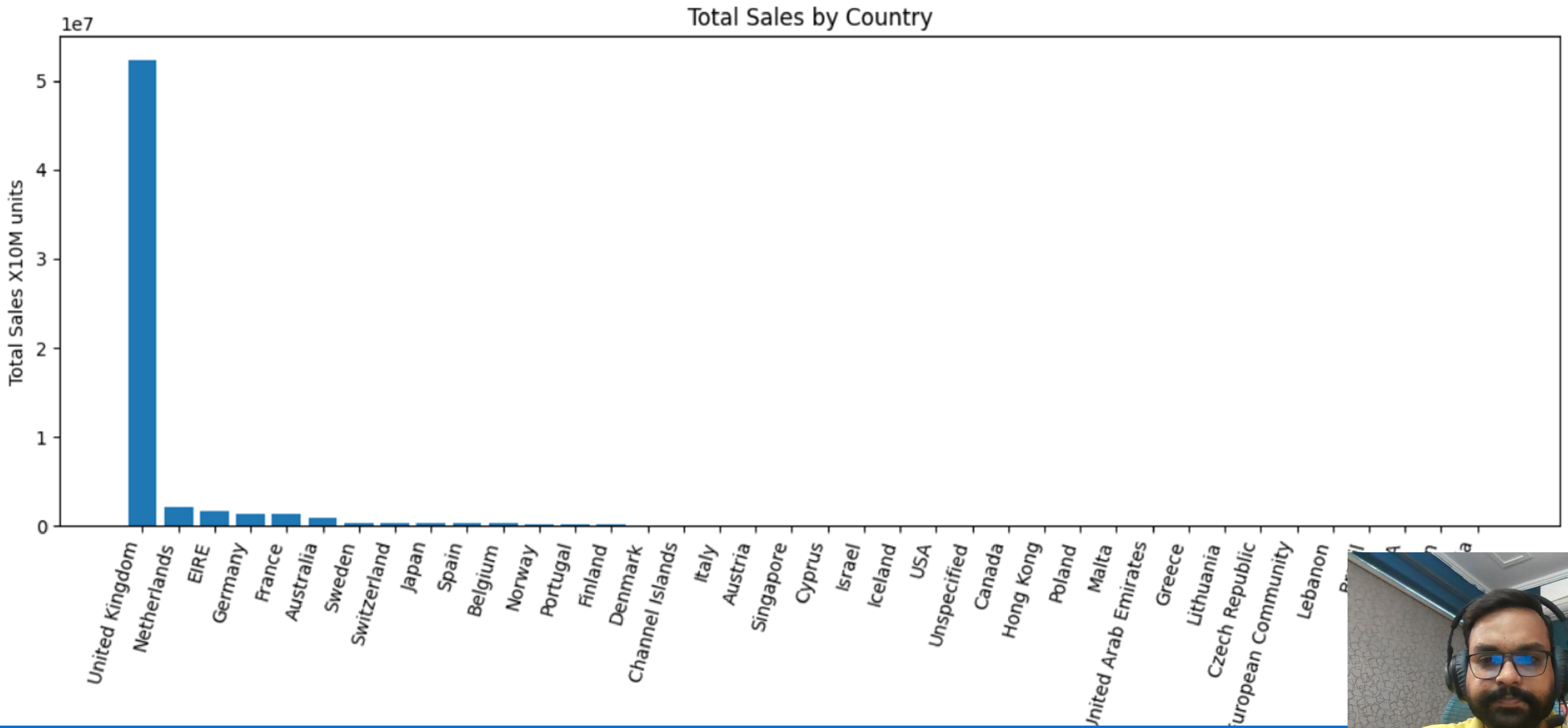
```
(531095, 13)
```



Exploratory Data Analysis

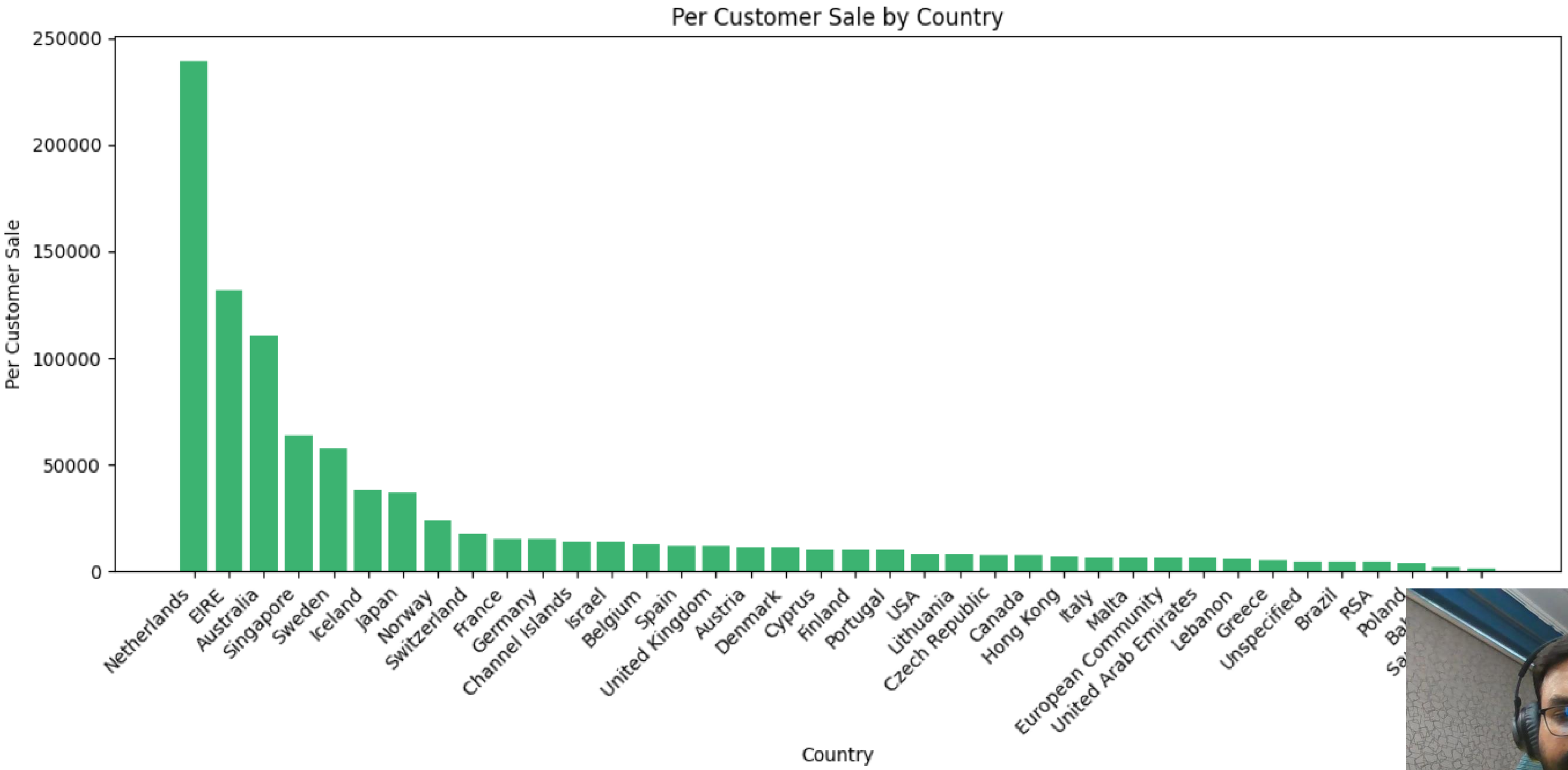
- Total sale value in units is 62,781,304 pounds.
- The total value is obtained by multiplying the price and the quantity sold.
- Adding up these values to get total sale There are a total of 531,095 rows in the dataset.

Total sales Country-wise

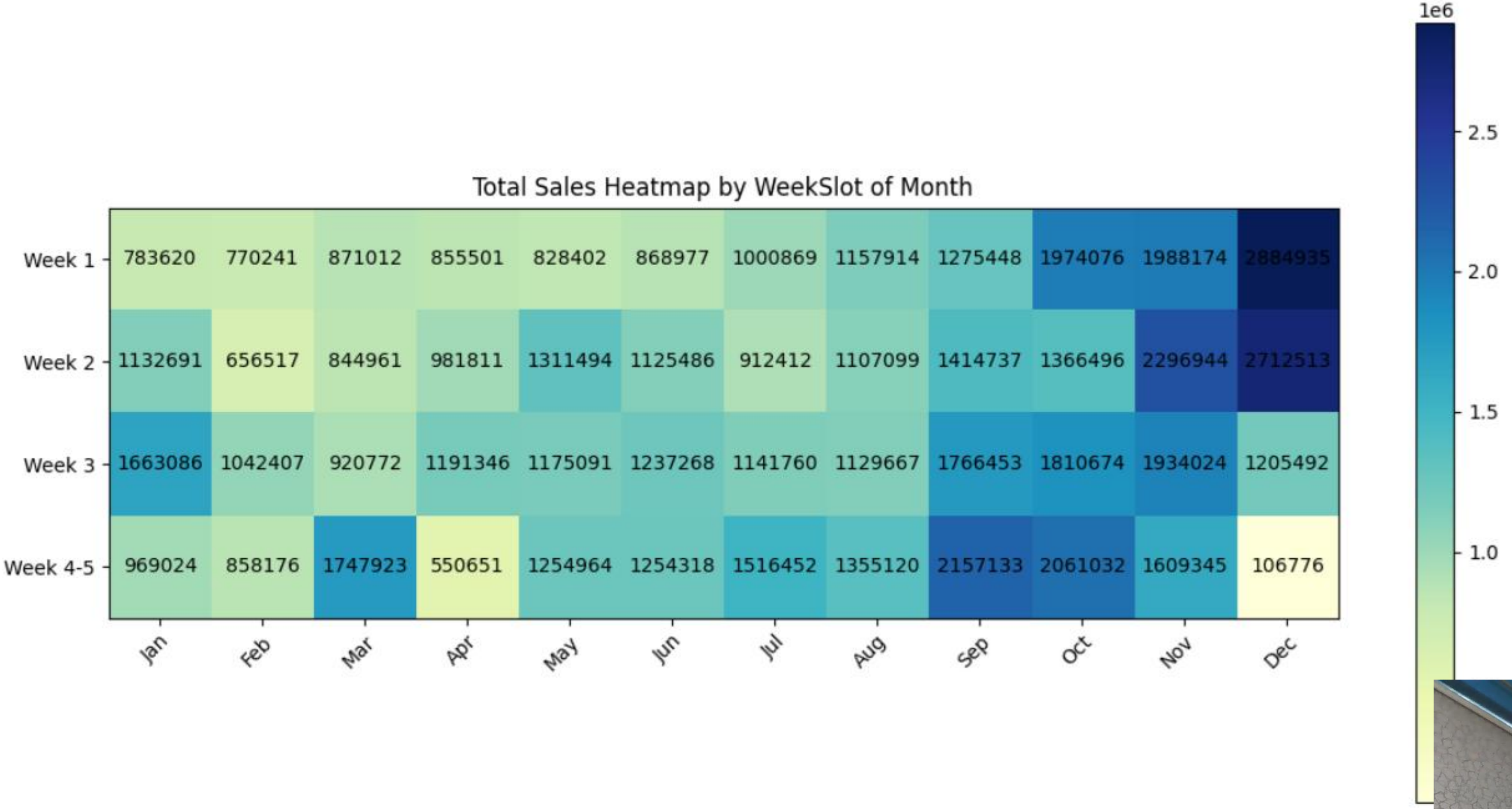


Per Customer Sales

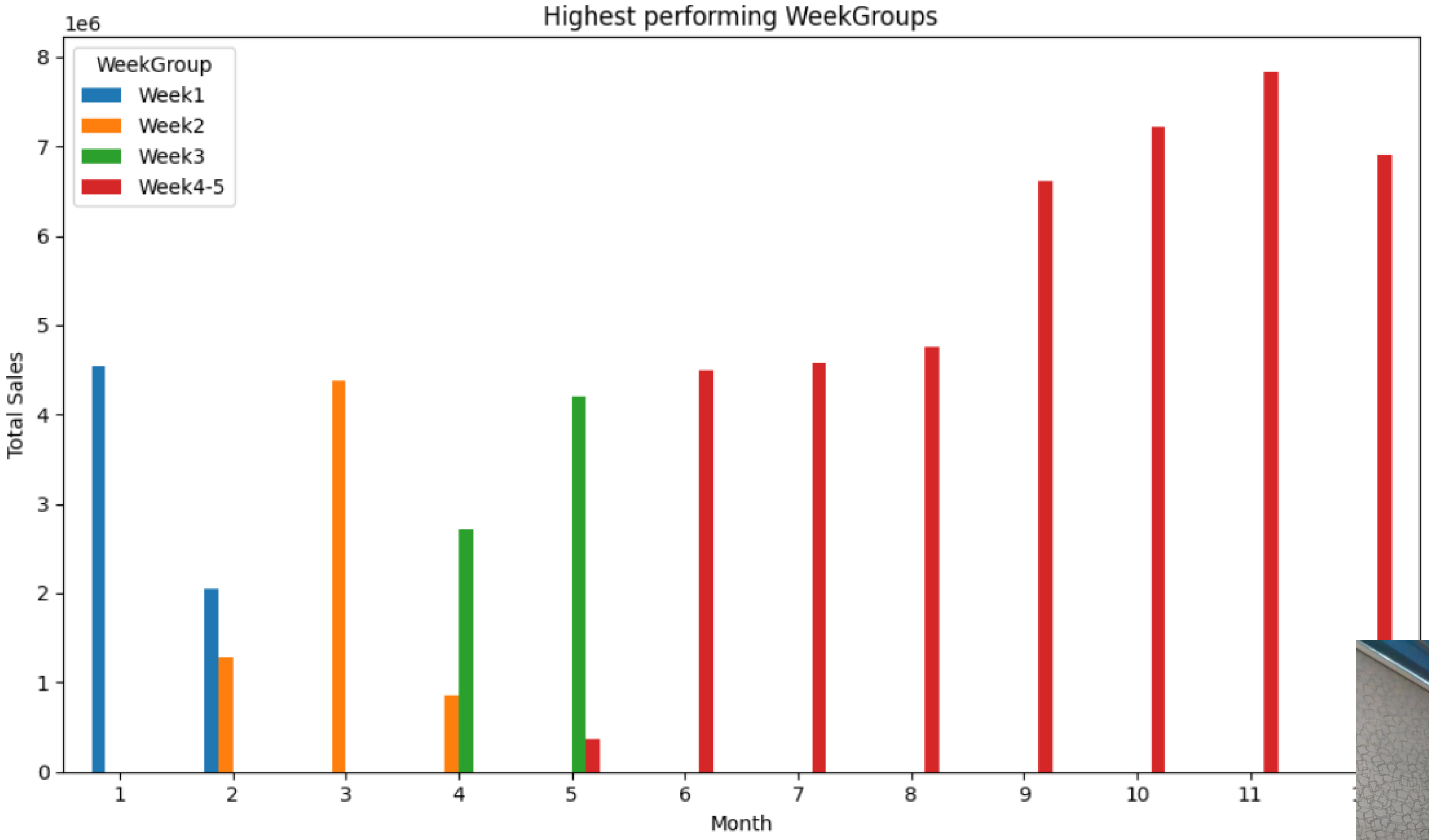
The graph shows per customer sales by country



Sales heatmap by week



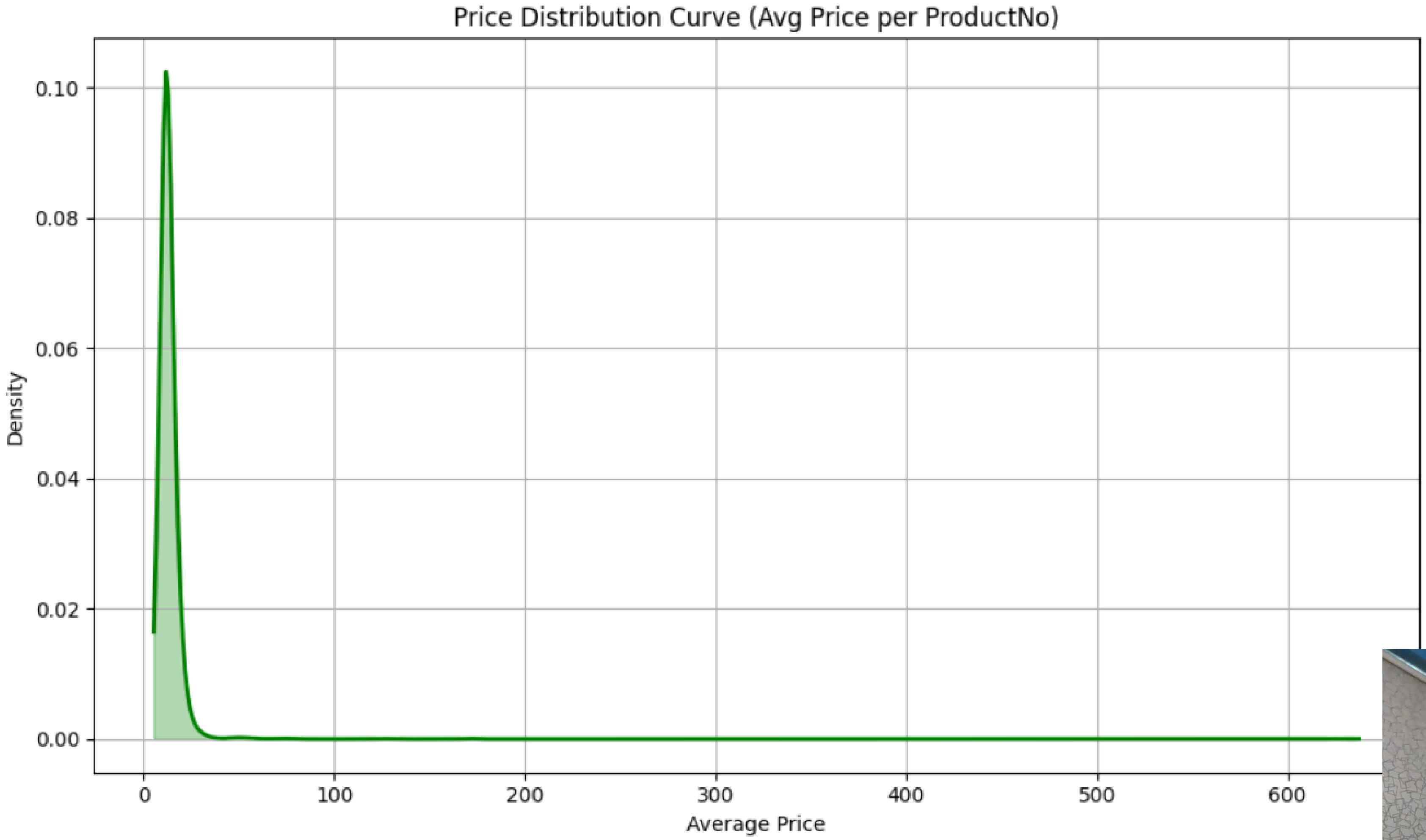
Dominating weeks in Total Sales



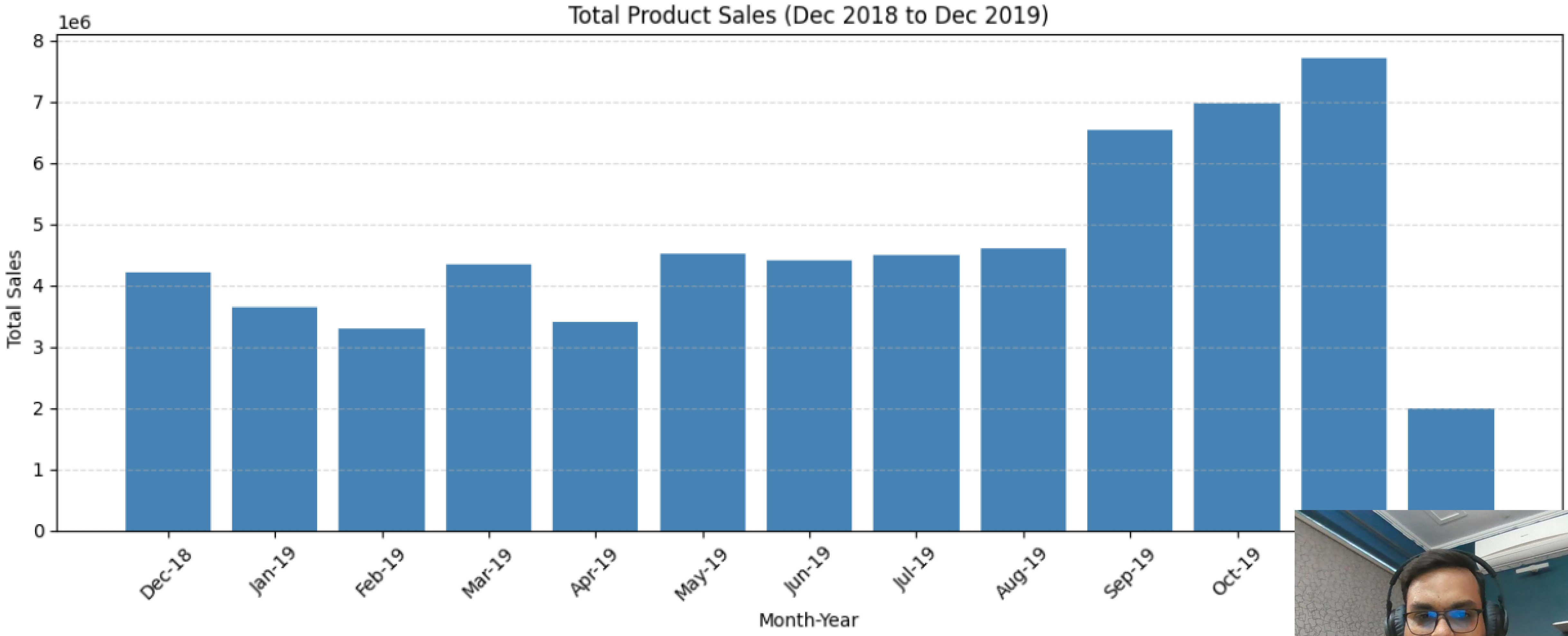
Heatmap of Total Sales

Price Distribution

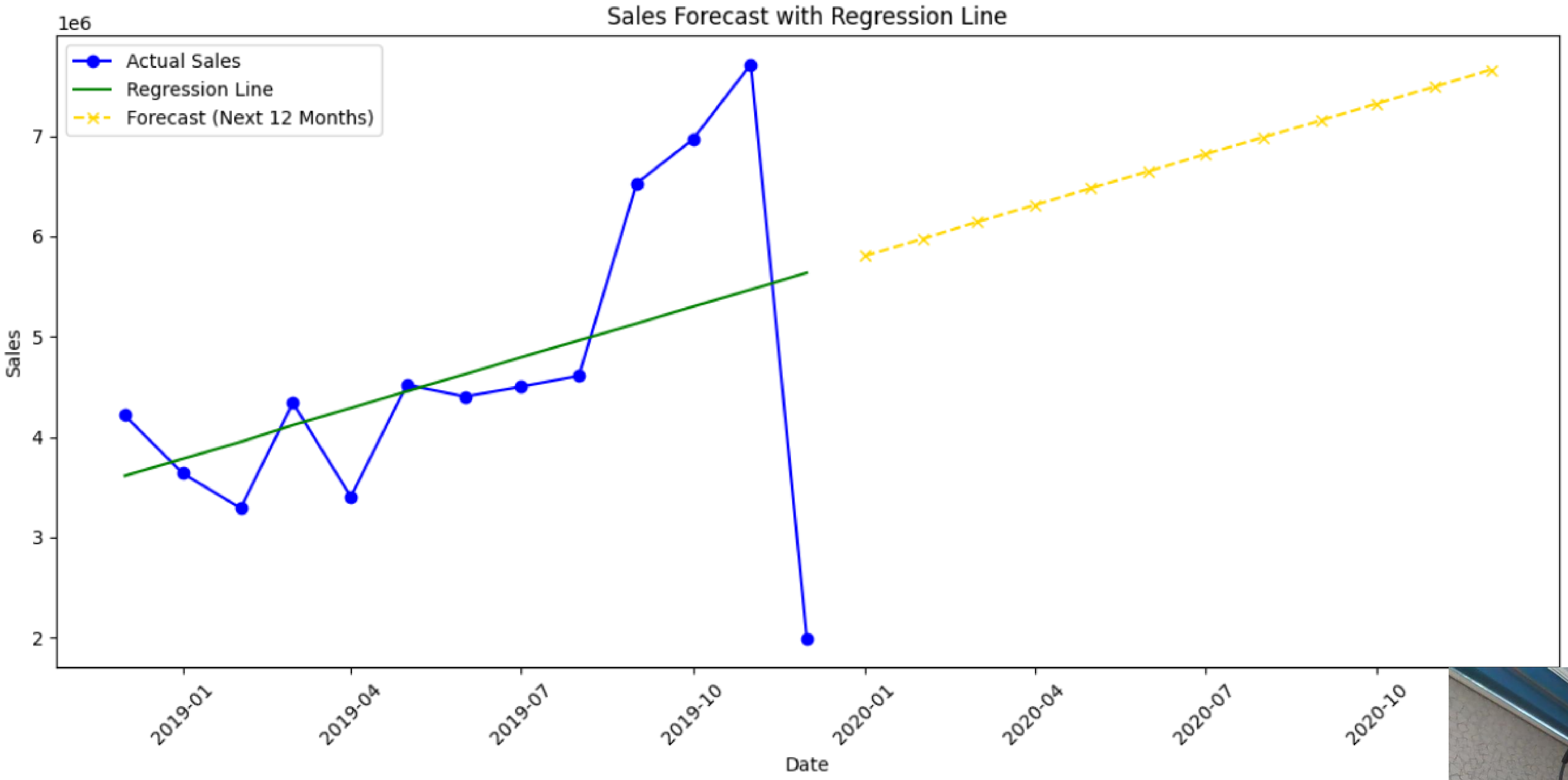
The following graphs shows price distribution in the data



Monthly Sales

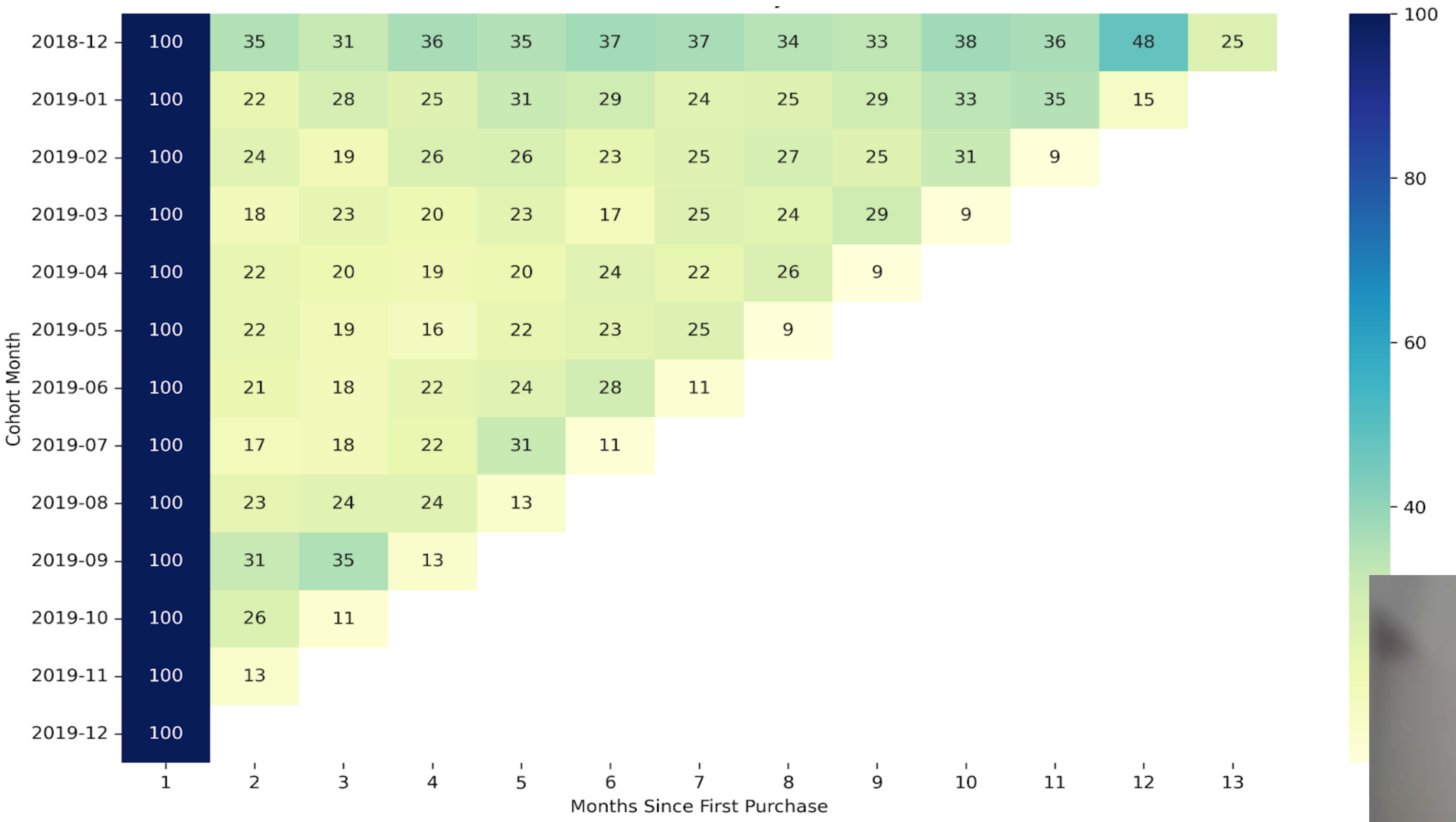


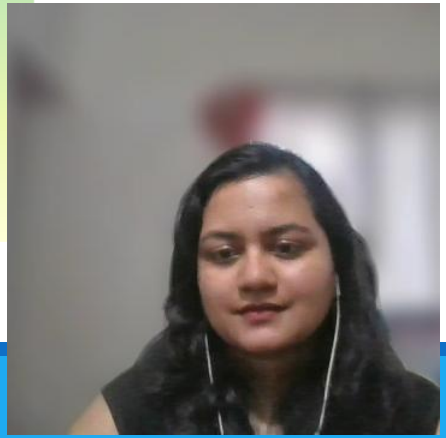
Linear Regression Line (Forcast)



We can see that the trend of sales is linear with positive growth

Cohort Analysis-Customer Retention





Model Selection and Analysis: **ARIMA**

Why ARIMA?

Designed for forecasting time-ordered data (e.g., sales trends).

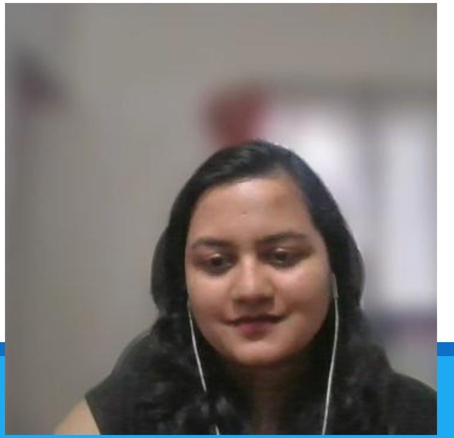
Captures:

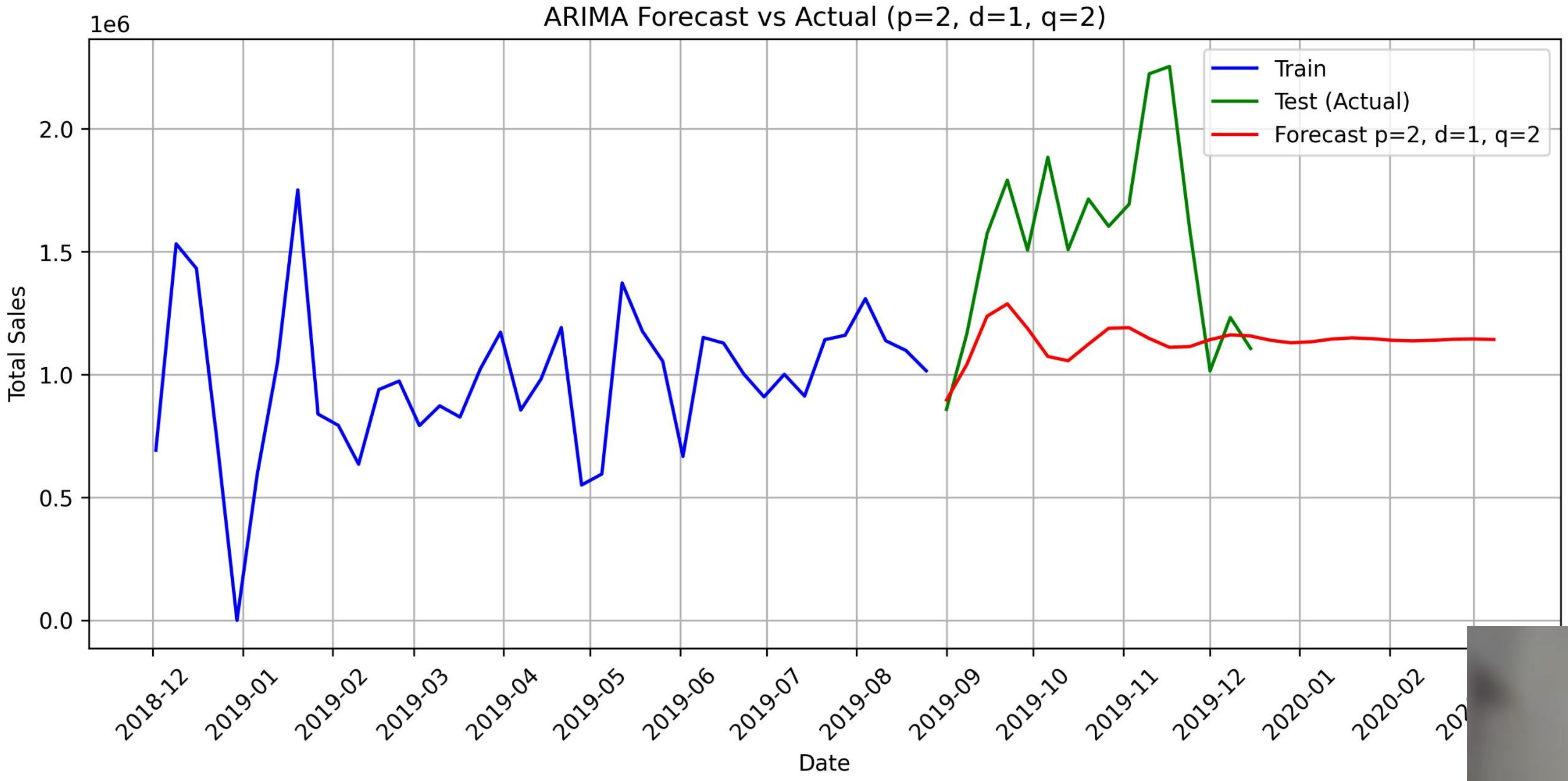
- ❑ **AR**: Uses prior observations for trend detection.
- ❑ **I**: Removes trend for stationarity.
- ❑ **MA**: Smooths random fluctuations.

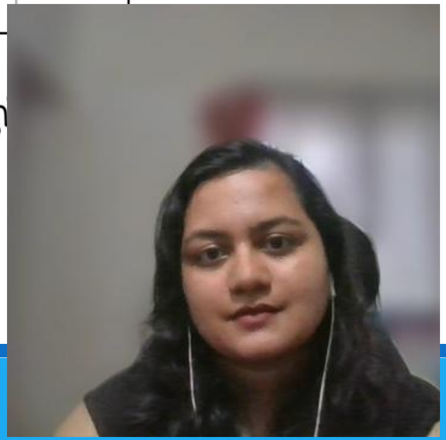
Ideal for **weekly sales trend prediction**.

ARIMA (Weekly Sales Forecast)

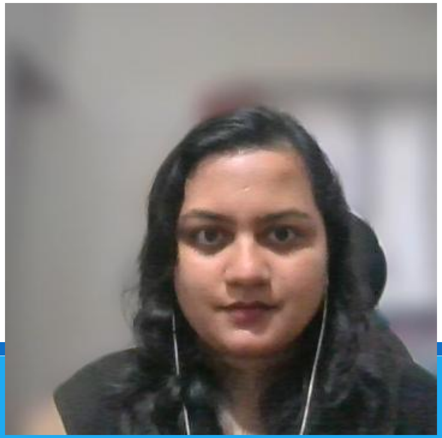
- **Best Parameters**: ARIMA (p=2, d=1, q=2)
- Achieved roughly **25% error rate** – satisfactory given sales variability.




ARIMA Forecast vs Actual (p=2, d=1, q=2)
1e6
Train
Test (Actual)
Forecast p=2, d=1, q=2
Total Sales
Date
2.0
1.5
1.0
0.5
0.0
2018-12
2019-01
2019-02
2019-03
2019-04
2019-05
2019-06
2019-07
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02



```
-------------------------------
Model: p=2, d=1, q=2 | AIC=1057.00 , MAE=439540.42, MAPE=25.03
Top 5 Actual vs Predicted:
                 Actual     Predicted     Difference
2019-09-01   858515.00  8.963537e+05  -37838.697680
2019-09-08  1164535.99  1.040567e+06  123969.311642
2019-09-15  1572890.84  1.237107e+06  335783.563235
2019-09-22  1790987.50  1.287607e+06  503380.984672
2019-09-29  1505963.44  1.187349e+06  318614.677185
```



Conclusion

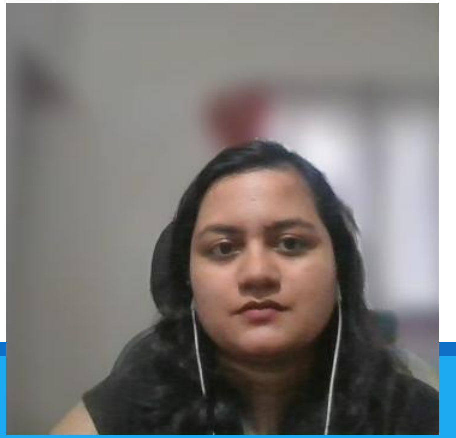
- Analyzed e-commerce sales data (1 year) to understand trends and customer behavior.

- Created a robust **end-to-end workflow**:
  Data Cleaning & Preparation ➔ EDA ➔ Modeling (ARIMA) ➔ Insights & Recommendations

- **Key Findings**
  – Weekly sales patterns captured well by ARIMA (25% error rate).

- **Limitations**:
  – Limited data (1 year) impacts long-term forecasting precision.
  – Daily forecasts using ARIMA prone to noise and higher error.

- **Future Scope**:
  – Incorporate multi-year data for improved trend and seasonality detection.
  – Refine forecasting and feature engineering for greater accuracy.



THANK YOU