Documentation: An Exploratory Data Analysis of HS Code 3542

What I Set Out to Do

When I first opened this dataset, I had a simple goal: understand where American semiconductors were going and why. The data represented 10 months of 2025 exports—specifically processors and controllers under HS Code 3542—shipped to 142 different countries. I wanted to dig deep, find patterns, spot outliers, and understand if certain countries were consistently receiving high-value shipments while others got the scraps.

This wasn't just about running a few charts. I wanted statistical proof that the patterns I was seeing were real, not just noise. I wanted to know: Are export values tied to specific countries? Do major economies dominate? How volatile are these shipments month to month? And most importantly—what story does the data tell about America's semiconductor trade relationships?

The Dataset I Was Working With

Before I dove into the analysis, I needed to understand what I had in front of me. Here's what the dataset contained:

- 950 export transactions
- January through October 2025
- 142 importing countries
- HS Code 3542: Electronic integrated circuits (processors and controllers)
- FOB values in USD (Free On Board—the value at the point of export)

Step 1: Getting My Bearings

The first thing I always do with any dataset is just look at it. I needed to know: Is this data clean? Are there any obvious problems? What does the structure look like?

Here's the code I ran to explore:

```
#Import necessary libraries for data manipulation, visualization, and stats

import pandas as pd

import numpy as np
```

```
import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats import chi2_contingency

# Load the cleaned export data

#Note: Using Latin1 encoding to handle special characters in country names

df = pd.read_csv('C:\\Users\\Aaron\\OneDrive\\Documents\\Debut projects\\2
Cleaned data\\Cleaned_HS_3542_Un_Comtrade_USA_Exports.csv',
encoding='Latin1')

# Initial data exploration

print(df.head())  # View first 5 rows

print(df.shape, df.describe(include='all'), df.info())
```

I used Latin1 encoding because some country names have special characters—like Türkiye and Curaçao. I wanted to make sure nothing broke when Python tried to read those. Everything looked good: 950 rows, 142 unique countries, and the FOB values were ready for analysis.

Step 2: Making Sure the Numbers Were Actually Numbers

You'd be surprised how often numeric columns aren't actually numeric. Maybe there's a stray text value somewhere. Maybe the CSV encoding messed something up. So I forced the conversion:

```
#Convert key columns to numeric (handling any non-numeric values)

for col in ['fobvalue','refYear', 'reporterCode']:

    df[col] = pd.to_numeric(df[col], errors='coerce')
```

The `errors='coerce'` parameter is key here. Instead of crashing when it hits something weird, Python just converts the bad value to NaN and keeps going. That way I could spot problems without breaking my whole analysis.

Step 3: Visualizing the Distribution—My First Big Insight

Now came the fun part. I wanted to see how export values were distributed. Were most shipments small? Were there a few mega-transactions skewing everything?

```
# Create histogram showing frequency distribution of export values

ax1 = sns.histplot(df['fobvalue'], bins=30)


# Add count labels on top of each bar for clarity

for p in ax1.patches:

    height = p.get_height()

    if height > 0:

        ax1.text(p.get_x() + p.get_width() / 2, height, int(height),

                ha='center', va='bottom', fontsize='8', rotation=0)


#Format the plot

plt.grid(axis='y', linestyle='--')  # Add horizontal gridlines

plt.ticklabel_format(style='plain', axis='x')  # Display x-axis numbers
without scientific notation

plt.show()

plt.close()
```

The histogram revealed something critical: the distribution was heavily right-skewed. Most exports were in the lower value ranges, but a few absolutely massive transactions pulled the tail way out to the right.

What this told me: The U.S. exports semiconductors to a lot of countries, but most of those shipments are relatively small. A handful of countries—probably the big tech manufacturing hubs—were getting the truly high-value shipments.

Step 4: The Chi-Square Test—Proving the Pattern Was Real

I could see patterns in the histogram, but I needed statistical proof. I wanted to answer the question: *Is there a real relationship between which country imports and what value they receive?* Or is it all just random variation?

To test this, I used a chi-square test of independence. First, I split the FOB values into three equal-sized groups:

```
# Create categorical variable by splitting FOB values into 3 equal-sized
groups

df['fob_cat'] = pd.qcut(df['fobvalue'], q=3, labels=['Low', 'Medium',
'High'])
```

```
# Build contingency table: rows = countries, columns = FOB categories
contingency = pd.crosstab(df['partnerISO'], df['fob_cat'])

# Perform chi-square test to see if country and export value category are independent
chi2, p, dof, expected = chi2_contingency(contingency)

#Print test results
print('Chi-Square Statistic: ', chi2)
print('P-value: ', p)
print('Degrees of Freedom: ', dof)
print('Expected Frequencies: ', expected)
```

The results were stunning:

• Chi-Square Statistic: 1431.93

• P-value: 5.14e-153 (basically zero)

• Degrees of Freedom: 282

That p-value is so small it's almost absurd. In statistical terms, this means there is a **highly significant relationship** between the importing country and the export value category. In plain English: Different countries consistently receive different value levels of semiconductors—and this is NOT random.

Why did I use `pd.qcut()`? Because it creates equal-sized groups rather than equal-width bins. This prevents the chi-square test from having empty cells, which would mess up the statistical validity. Each category (Low, Medium, High) had roughly the same number of observations, making the test more robust.

Step 5: Heatmaps—Seeing the Pattern Visually

The chi-square test gave me statistical proof. Now I wanted to *see* the pattern. I created heatmaps showing which countries clustered in which value categories.

```
#Extract all country names from the contingency table index
Countries = contingency.index.tolist()

#Loop through countries in groups of 10 to create manageable heatmaps
for i in range(0, len(Countries), 10):
    group = Countries[i:i+10]  # Select 10 countries at a time
```

```
    subset = contingency.loc[group]  # Filter contingency table for this
group


    #Create heatmap visualization

    plt.figure(figsize=(12, 6))

    sns.heatmap(subset, annot=True, fmt='d', cmap='Greens')  # annot=True
shows values, fmt='d' for integers

    plt.title('Contingency Table: Importer Country Vs FOB Category')

    plt.xlabel('FOB Category')

    plt.ylabel('Importer Country')

    plt.show()

    plt.close()
```

I processed the heatmaps in groups of 10 countries to keep them readable. Trying to cram 142 countries into one chart would've been a mess.

What I saw: Major economies like China, Germany, Japan, and South Korea showed heavy concentration in the 'High' category. Smaller economies appeared predominantly in the 'Low' category. This makes sense—big tech manufacturing hubs need high-value, high-volume semiconductor shipments. Smaller countries might only need occasional, lower-value imports.

Step 6: Boxplots—Understanding Spread and Outliers

Next, I wanted to understand the distribution *within* each country. Are shipments to China consistently high? Or are there wild swings? Are there outliers—unusually large or small transactions?

```
# Open file to save boxplot statistics

with open('boxplt_stats.txt', 'w') as f:

    # Process countries in groups of 10

    for i in range(0, len(Countries_list), 10):

        group = Countries_list[i:i+10]  # Current group of 10 countries

        subset = df[df['partnerISO'].isin(group)]  # Filter data for these
countries


        # Skip if no data exists for this group

        if subset.empty:
```

```
            continue

        # Create boxplot showing FOB value distribution for each country
        plt.figure(figsize=(12, 6))
        sns.boxplot(x='partnerISO', y='fobvalue', data=subset)
        plt.title('Distribution of FOB by Importer Country')
        plt.xlabel('Importer')
        plt.ylabel('FOB')
        plt.ticklabel_format(style='plain', axis='y')  #Avoid scientific
notation
        plt.grid(axis='y', linestyle='--')
        plt.tight_layout()
        plt.show()
        plt.close()

        # Calculate detailed statistics for each country in the group
        stats = {}
        for country in group:
            values = subset.loc[subset['partnerISO'] == country, 'fobvalue']

            if values.empty:
                continue

            # Calculate boxplot statistics manually
            q1 = np.percentile(values, 25)  #First quartile (25th percentile)
            q2 = np.percentile(values, 50)  #Median (50th percentile)
            q3 = np.percentile(values, 75)  #Third quartile (75th percentile)
            iqr = q3 - q1  #Interquartile range

            # Whiskers extend to 1.5*IQR from quartiles
            lower_whisker = values[values >= (q1 - 1.5 * iqr)].min()
```

```
            upper_whisker = values[values <= (q3 + 1.5 * iqr)].max()


            #Identify outliers (values beyond whiskers)
            outliers = values[(values < (q1 - 1.5 * iqr)) | (values > (q3 +
1.5 * iqr))]


            #Store stats in dictionary
            stats[country] = {
                'count': len(values),
                'Q1': q1,
                'Median': q2,
                'Q3': q3,
                'Lower Whisker': lower_whisker,
                'Upper Whisker': upper_whisker,
                'Outliers': outliers.tolist()
            }


        # Print stats to console
        print(f'\nGroup {i//10 + 1} boxplot stats:\n')
        for country, s in stats.items():
            print(country, s)


        # Write stats to file
        f.write(f'\nGroup {i//10 + 1} ({group})\n')
        for country, s in stats.items():
            f.write(f'{country}: {s}\n')
        f.write('\n' + '-'*60 + '\n')
```

I manually calculated the quartiles, whiskers, and outliers using the standard 1.5 × IQR rule. This gave me precise numeric values for documentation and ensured consistency with the visual boxplots.

Key findings from the boxplot statistics:

• China: Extremely high median (~$838M) with massive outliers, indicating both consistent high-value exports AND occasional mega-transactions

• Canada: High median (~$51M) with moderate variability

• Mexico: Moderate median (~$17M) with some high outliers

• Small economies: Very low medians (<$10K) with minimal variability

• Outliers common in major tech hubs (China, Singapore, South Korea), suggesting periodic large orders or special projects

Step 7: Time Series Analysis—Tracking Changes Month by Month

Finally, I wanted to see how exports changed over time. Were there growth trends? Seasonal patterns? Sudden spikes or drops?

```
#Open file to save line plot statistics

with open("lineplot_stats.txt", "w") as f:

    # Process countries in groups of 10

    for i in range(0, len(Countries_list), 10):

        group = Countries_list[i:i+10]  #Current group of 10 countries

        subset = df[df['partnerISO'].isin(group)]  #Filter data for these
countries


        # Skip if no data exists for this group

        if subset.empty:

            print(f"Skipping Group {i//10 + 1}: no data")

            continue


        # Create line plot showing FOB trends over months

        plt.figure(figsize=(12, 6))

        ax = sns.lineplot(x='refMonth', y='fobvalue', hue='partnerISO',
data=subset)


        #Calculate time series statistics for each country

        stats = {}

        for country in group:

            country_data = subset[subset['partnerISO'] == country]
```

```
            if country_data.empty:
                continue

            #Sort by month to ensure chronological order
            country_data = country_data.sort_values("refMonth")

            #Create dictionary of month-to-value mappings
            month_values = {
                str(m): int(v)
                for m, v in zip(country_data["refMonth"],
country_data["fobvalue"])
                if v > 0  #Only include non-zero values
            }

            if not month_values:
                continue

            # Store key time series stats
            stats[country] = {
                "first": int(country_data["fobvalue"].iloc[0]),  # First
month value
                "max": int(country_data["fobvalue"].max()),      #Maximum
value
                "last": int(country_data["fobvalue"].iloc[-1]),  # Last month
value
                "per_month": month_values                        # All
monthly values
            }

        # Annotate key points on each line (first, max, last)
        for line in ax.lines:
            x_data = line.get_xdata()
```

```
            y_data = line.get_ydata()


            if len(x_data) > 0:
                # Label first point
                ax.text(x_data[0], y_data[0], f'{int(y_data[0]):,}',
                        ha='center', va='bottom', fontsize=6, rotation=45)


                # Label maximum point
                max_idx = y_data.argmax()
                ax.text(x_data[max_idx], y_data[max_idx],
f'{int(y_data[max_idx]):,}',
                        ha='center', va='bottom', fontsize=6, rotation=45)


                # Label last point
                ax.text(x_data[-1], y_data[-1], f'{int(y_data[-1]):,}',
                        ha='center', va='bottom', fontsize=6, rotation=45)


        #Format and display plot
        plt.title(f'FOB Value Over Time By Importer Country (Group {i//10 +
1})')
        plt.ticklabel_format(style='plain', axis='y')  # Avoid scientific
notation
        plt.grid(axis='y', linestyle='--')
        plt.xlabel('Month')
        plt.ylabel('FOB')
        plt.tight_layout()
        plt.show()
        plt.close()


        #Print stats to console
        print(f"\nGroup {i//10 + 1} lineplot stats:")
        for country, s in stats.items():
```

```
            print(country)
            print("  First:", s["first"])
            print("  Max:", s["max"])
            print("  Last:", s["last"])
            print("  Per month:", s["per_month"])

        #write stats to file
        f.write(f"\nGroup {i//10 + 1} ({group})\n")
        for country, s in stats.items():
            f.write(f"{country}\n")
            f.write(f"  First: {s['first']}\n")
            f.write(f"  Max: {s['max']}\n")
            f.write(f"  Last: {s['last']}\n")
            f.write("  Per month:\n")
            for month, val in s["per_month"].items():
                f.write(f"    {month}: {val}\n")
        f.write("\n" + "-"*60 + "\n")
```

I annotated the first, maximum, and last values on each trend line. This let me quickly identify starting points, peaks, and current status without having to consult the raw data.

What the time series revealed:

• China dominated throughout all 10 months, with values ranging from $400M-$900M per month

• High volatility: Several countries showed significant month-to-month variation, suggesting project-based or cyclical ordering patterns

• Growth trends: Some countries (Brazil, India) showed upward trajectories, indicating growing semiconductor demand

• Limited seasonality: With only 10 months of data, I couldn't definitively establish seasonal patterns, but some countries showed Q2-Q3 peaks

Why I Made the Choices I Did

1. Three-Category Split with pd.qcut()

I used `pd.qcut()` instead of `pd.cut()` because it creates equal-sized groups rather than equal-width bins. This ensures each category has roughly the same number of

observations, making the chi-square test more statistically robust and preventing empty cells that would invalidate the test.

2. Groups of 10 Countries

I processed visualizations in batches of 10 countries to keep the charts readable. Trying to show all 142 countries at once would've created an incomprehensible mess. It also made the file output and console review manageable.

3. Manual Boxplot Statistics

I calculated quartiles, whiskers, and outliers manually using NumPy rather than relying solely on seaborn's internal calculations. This gave me precise numeric values for documentation and ensured perfect consistency with the visual boxplots.

4. 1.5 × IQR Rule for Outliers

I used the standard 1.5 × IQR criterion for identifying outliers. This is the industry-standard approach that flags genuinely unusual values without being overly sensitive to normal variation. It's the same rule used by default in most statistical software.

5. Line Plot Annotations

I labeled the first, maximum, and last values on each trend line. This helps viewers quickly identify starting points, peaks, and current status without having to consult raw data or squint at the axes.

What I Learned: The Big Picture

1. Extreme Concentration of Export Value

China absolutely dominates U.S. semiconductor exports. It accounts for the vast majority of high-value transactions. The top 10 countries likely represent 80%+ of total export value—a classic Pareto distribution. Most countries (100+) receive relatively small, consistent shipments.

2. Statistically Significant Country Patterns

The chi-square test ($p < 0.001$) proved that export value categories are NOT randomly distributed across countries. This validates using country as a key segmentation variable for trade strategy. Different countries have fundamentally different semiconductor demand profiles.

3. High Variability in Major Markets

Large economies like China, Japan, South Korea, and Germany show high medians but also massive outliers. This suggests a mix of routine orders and occasional mega-projects or bulk purchases. Smaller economies show much more consistent (but lower) order patterns.

4. Limited Temporal Patterns

With only 10 months of data, I couldn't definitively establish strong seasonality trends. However, month-to-month volatility was evident, particularly in major markets. Year-over-year analysis would require additional data from previous years.

5. Strategic Implications

If I were advising a semiconductor company, I'd recommend focusing resources on maintaining and growing relationships with the top 10-15 countries. Smaller markets provide diversification but limited revenue impact. The outlier transactions in major markets warrant investigation: Are they one-off deals or recurring patterns? The time series data suggests potential for predictive modeling if extended to multi-year datasets.

What This Analysis Couldn't Tell Me

Every analysis has limitations, and I want to be transparent about what this EDA could and couldn't answer:

- Only 10 months of data limits trend analysis and seasonality detection
- FOB value doesn't account for quantity, making per-unit pricing analysis impossible
- No data on rejected orders, delayed shipments, or trade barriers

Files I Generated During This Analysis

1. boxplt_stats.txt: Comprehensive quartile, whisker, and outlier statistics for all 142 countries

2. lineplot_stats.txt: First, maximum, last, and month-by-month values for temporal analysis

3. Multiple visualizations: Histograms, heatmaps, boxplots, and line plots saved during execution

Final Thoughts

This EDA gave me exactly what I was looking for: a deep understanding of U.S. semiconductor export patterns across 142 countries. I proved statistically that export values are strongly tied to specific countries. I identified extreme concentration in major markets and high volatility in key relationships. And I documented every step so the analysis could be replicated, extended, or challenged.

The dataset is now ready for SQL-based aggregation, strategic reporting, and decision-making. More importantly, I learned a lot about how global semiconductor trade actually works—and that's what makes data analysis worth doing.