India’s Foreign Trade Trends: 25 Years of Imports and Exports

---

1. Project Overview

This project provides a comprehensive analysis of India's import and export data from 1997 to 2022. The data was sourced, cleaned, and transformed using Python (Pandas and NumPy), stored in an SQLite database, and finally visualized using Power BI for interactive insights. The entire project, including the data and scripts, is managed using Git and hosted on GitHub.

Technologies Used:

- **Python:** Pandas, NumPy, SQLite3
- **Database:** SQLite
- **Business Intelligence:** Microsoft Power BI
- **Version Control:** Git, GitHub

2. Data Source and Initial State

The raw data was provided in a CSV format, named `EAI.csv`. This dataset contains information on country-wise exports, imports, total trade, and trade balance across various financial years.

Initial Data Snippet (`EAI.csv`):

Country	Export	Import	Total Trade	Trade Balance	Financial Year (start)	Financial Year (end)
AFGH…	21.25	10.7	31.95	10.55	1997	1998
AFGH…	12.81	28.14	40.95	-15.33	1998	1999
AFGH…	33.2	21.06	54.26	12.15	1999	2000
AFGH…	25.86	26.59	52.45	-0.73	2000	2001
AFGH…	24.37	17.52	41.89	6.85	2001	2002

3.Data Cleaning and Preprocessing (Python - Pandas/NumPy)

The raw data required significant cleaning to handle missing values, inconsistent data types, and prepare it for analysis. The `EAI_Cleaned.py` script was used for this purpose.

Key Cleaning Steps:

- **Load Data:** The `EAI.csv` file was loaded into a Pandas DataFrame.
- **Handle Missing Country Names:** Rows where the 'Country' column was entirely missing were dropped.

- **Convert Numeric Columns:** 'Export', 'Import', 'Total Trade', and 'Trade Balance' columns contained commas and were of object type. They were converted to string, commas were removed, and then coerced to numeric type. Any errors during conversion were replaced with `NaN` and then filled with `0`.
- **Fill Remaining NaNs:** All other remaining `NaN` values across the DataFrame were filled with `0`.
- **Create 'Year' Column:** A new 'Year' column was extracted from 'Financial Year(start)', ensuring it was an integer type.
- **Add India's Perspective Columns:** To provide a clear perspective from India's viewpoint, the following columns were added:`India_Imports`: Represents other countries' exports to India (which are India's imports). This was derived from the 'Export' column in the original dataset.

`India_Exports`: Represents other countries' imports from India (which are India's exports). This was derived from the 'Import' column in the original dataset.

`IndiaTrade_Balance`: Calculated as `India_Exports - India_Imports`.

Save Cleaned Data: The processed data was saved to a new CSV file named `Complete_India_Trade_Data_All_Countries_Filled.csv`.

Python Code for Data Cleaning (`EAI_Cleaned.py`):

Python

```
import pandas as pd
import numpy as np
# Load the complete dataset
df = pd.read_csv('EAI.csv')
# Data Cleaning - keep all rows, only remove if Country is completely missing
df = df.dropna(subset=['Country'], how='all')
# Convert numeric columns - fill blanks with 0 and handle commas
numeric_cols = ['Export', 'Import', 'Total Trade', 'Trade Balance']
for col in numeric_cols:
# Convert to string, replace commas, then to numeric
 df[col] = df[col].astype(str).str.replace(',', '')
# Convert to numeric, forcing errors to NaN, then fill with 0
df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)

# Fill any remaining NaN values with 0 (for non-numeric columns if any)
df = df.fillna(0)

# Create Year column from Financial Year(start) and fill blanks with 0
df['Year'] = df['Financial Year(start)'].astype(str).str.extract('(\\d+)')[0].fillna(0).astype(int)

# Add India's perspective columns
df['India_Imports'] = df['Export']  # Other countries' exports to India (India's imports)
df['India_Exports'] = df['Import']  # Other countries' imports from India (India's exports)
df['India_Trade_Balance'] = df['India_Exports'] - df['India_Imports']

# Save the complete dataset with all countries and all years
output_filename = 'Complete_India_Trade_Data_All_Countries_Filled.csv'
df.to_csv(output_filename, index=False)
```

```
# Verification
print(f"Total records processed: {len(df)}")
print(f"Total unique countries: {df['Country'].nunique()}")
print(f"Years covered: {df['Year'].min()} to {df['Year'].max()}")
```

Cleaned Data Snippet (`Complete_India_Trade_Data_All_Countries_Filled.csv`):

Country	Export	Import	Total Trade	Trade Balance	Financial Year (start)	Financial Year (end)	Year	India_Imports	India_Exports	India_Trade_Balance
A...	21.25	10.7	31.95	10.55	1997	1998	1997	21.25	10.7	-10.55
A...	12.81	28.14	40.95	-15.33	1998	1999	1998	12.81	28.14	15.33
A...	33.2	21.06	54.26	12.15	1999	2000	1999	33.2	21.06	-12.14
A...	25.86	26.59	52.45	-0.73	2000	2001	2000	25.86	26.59	0.73
A...	24.37	17.52	41.89	6.85	2001	2002	2001	24.37	17.52	-6.85

4.Database Creation (Python - SQLite)

The cleaned data was then loaded into an SQLite database (`india_trade.db`) for efficient querying and management. The `EAI_DATABASE.py` script handles the database creation and data insertion.

Key Database Steps:

Connect to Database: An SQLite connection was established, creating the `india_trade.db` file if it didn't exist.

Create Table: A table named `trade_data` was created with appropriate columns to store the cleaned trade information.

Insert Data: The DataFrame containing the cleaned data was directly inserted into the `trade_data` table.

Create Views (for easier querying):

`india_yearly_summary`: Aggregates India's total exports, imports, and trade balance per year.

`top_trading_partners`: Provides a summary of total trade volume, exports, imports, and trade balance for each country across all years, ordered by total trade volume.

Python Code for Database Creation (`EAI_DATABASE.py`):

```
Python
import pandas as pd
import numpy as np
```

```
import sqlite3
from pathlib import Path

# Load and preprocess the data
print("Loading and preprocessing data...")
df = pd.read_csv('EAI.csv')

# Data Cleaning (as described in section 3)
df = df.dropna(subset=['Country'], how='all')

numeric_cols = ['Export', 'Import', 'Total Trade', 'Trade Balance']
for col in numeric_cols:
    df[col] = df[col].astype(str).str.replace(',', '')
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)

df = df.fillna(0)

df['Year'] = df['Financial Year(start)'].astype(str).str.extract('(\\d+)')[0].fillna(0).astype(int)

df['India_Imports'] = df['Export']
df['India_Exports'] = df['Import']
df['India_Trade_Balance'] = df['India_Exports'] - df['India_Imports']

# SQLite Database Setup
db_path = 'india_trade.db'
print(f"\nCreating SQLite database at {db_path}...")

# Connect to SQLite (this will create the database if it doesn't exist)
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Drop table if it already exists (for clean runs)
cursor.execute("DROP TABLE IF EXISTS trade_data")

# Create table with all necessary columns
cursor.execute('''
CREATE TABLE trade_data (
    Country TEXT,
    Export REAL,
    Import REAL,
    "Total Trade" REAL,
    "Trade Balance" REAL,
    "Financial Year(start)" TEXT,
    "Financial Year(end)" TEXT,
    Year INTEGER,
    India_Imports REAL,
    India_Exports REAL,
    India_Trade_Balance REAL
)
''')
```

```
# Insert data from DataFrame into the table
df.to_sql('trade_data', conn, if_exists='append', index=False)

# Create views for easier analysis

# 1. India Yearly Summary
cursor.execute('''
CREATE VIEW IF NOT EXISTS india_yearly_summary AS
SELECT
    year,
    SUM(india_exports) AS total_india_exports,
    SUM(india_imports) AS total_india_imports,
    SUM(india_trade_balance) AS total_india_trade_balance
FROM trade_data
GROUP BY year
ORDER BY year;
''')

# 2. Top Trading Partners
cursor.execute('''
CREATE VIEW IF NOT EXISTS top_trading_partners AS
SELECT
    country,
    SUM(india_exports) AS total_exports_to_country,
    SUM(india_imports) AS total_imports_from_country,
    SUM(india_trade_balance) as total_trade_balance,
    SUM("Total Trade") as total_trade_volume,
    COUNT(*) as years_of_data
FROM trade_data
GROUP BY country
ORDER BY total_trade_volume DESC;
''')

# Commit changes and close connection
conn.commit()
conn.close()

print(f"\nDatabase successfully created with:")
print(f"- {len(df):,} total trade records")
print(f"- {df['Country'].nunique()} unique countries")
print(f"- Data from {df['Year'].min()} to {df['Year'].max()}")

print("\nSample queries you can run:")
print("1. Get India's trade with China:")
print('   sqlite> SELECT * FROM trade_data WHERE country = "CHINA P RP" ORDER BY year;')

print("\n2. View India's yearly trade summary:")
print('   sqlite> SELECT * FROM india_yearly_summary;')
print("\n3. View top 10 trading partners:")
print('   sqlite> SELECT * FROM top_trading_partners LIMIT 10;')
```

print("\n4. Find countries with trade surplus with India:")
print(' sqlite> SELECT country, total_trade_balance FROM top_trading_partners WHERE total_trade_balance > 0 ORDER BY total_trade_balance DESC;')

Sample SQLite Queries:Retrieve all data for a specific country (e.g., CHINA P RP):
SQL
SELECT * FROM trade_data WHERE Country = 'CHINA P RP' ORDER BY Year;
View India's yearly trade summary:
SQL
SELECT * FROM india_yearly_summary;
Get top 10 trading partners by total trade volume:
SQL
SELECT * FROM top_trading_partners LIMIT 10;
Find countries with a trade surplus with India:
SQL
SELECT Country, total_trade_balance FROM top_trading_partners WHERE total_trade_balance > 0 ORDER BY total_trade_balance DESC;

Extract Data from SQLite to CSV (using Python is common):

Install necessary libraries: If you don't have them, you'll need `pandas` and `sqlite3`.

```
pip install pandas
```

Python Script: Write a Python script to connect to your SQLite database, query the data from the desired tables or views, and then save that data to a CSV file.

Python

```
import sqlite3
import pandas as pd

db_path = 'india_trade.db' # Replace with your .db file name
output_csv_path = 'trade_data_from_db.csv' # Desired output CSV file name

conn = sqlite3.connect(db_path)

# Example: Querying a specific table or view
# You can change 'trade_data' to 'india_yearly_summary' or 'top_trading_partners'
query = "SELECT * FROM trade_data"

try:
    df = pd.read_sql_query(query, conn)
    df.to_csv(output_csv_path, index=False)
    print(f"Data successfully exported to {output_csv_path}")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
conn.close()
```
this script: Execute the Python file. This will create a CSV file containing your database's data.

5. Data Visualization and Analysis (Power BI)

Power BI was used to create interactive dashboards and visualizations to explore India's import and export trends. The `trade_data.xls` file contains the Power BI report.

Steps to Connect Power BI to SQLite Database:

Open Power BI Desktop: Launch Power BI Desktop application.

Get Data: Click on "Get Data" from the Home tab.

Browse for excel File: In the SQLite database dialog box, browse and select the `trade_data.xls` file created earlier. Click "Open".

Navigator: In the Navigator window, you will see all the tables and views from your SQLite database. Select `trade_data`, `india_yearly_summary`, and `top_trading_partners`.

Load Data: Click "Load" to import the selected tables/views into Power BI.

Data Modeling (if needed): Ensure relationships between tables are correctly identified (though for this setup, direct loading of views might be sufficient for most analyses).

Power BI Visualizations and Screenshots:

Here are some screenshots from the Power BI dashboard, showcasing key insights into India's trade:

Overview of India's Trade (Imports, Exports, Trade Balance over Time)

This slide provides a high-level overview of India's import and export performance and the resulting trade balance across the years.

LINE CHARTS:

📊 Power BI Line Chart – India Import vs Export (1997–2022)

🔹 1. Purpose

- To analyze trade trends over 25 years.
- Compare the growth and gaps between imports and exports.
- Identify trade surplus or deficit patterns.

🔹 2. Data Setup

- X-axis: Year (1997 to 2022).
- Y-axis: Value in USD Billion (or INR Crore).
- Legend: Import, Export (two lines).

🔹 3. Key Insights You Can Show

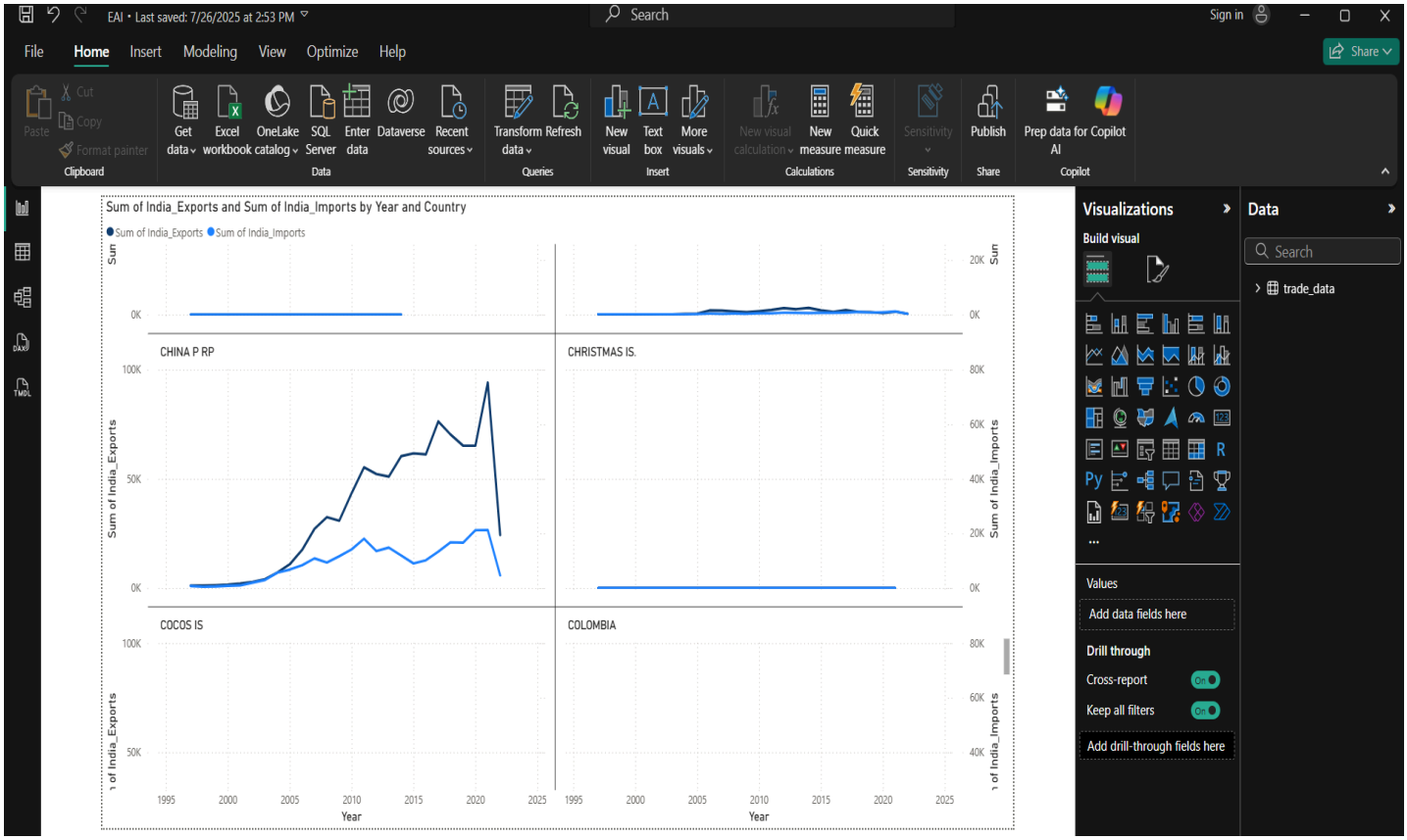
- 📈 Trend Analysis: How trade volume evolved.
- 🟥 Trade Deficit Periods: When imports > exports.
- 🟩 Trade Surplus/Improvement: Years when the gap reduced.
- 📌 Important Events Impacting Trade:
  - 2008 Global Recession
  - 2016 Demonetization
  - 2020 COVID-19
  - 2022 Russia-Ukraine war impact on oil import bills

🔹 4. Visual Features to Use

- Dual-line chart: One for Exports, one for Imports.
- Markers: Highlight major events (tooltip or annotation).
- Data labels (optional): For peak values.
- Custom tooltip: Show year-wise details on hover.

5. Customizations

- Format Y-axis with currency (₹ or $).
- Color Exports (e.g., green) and Imports (e.g., red).
- Add slicer for filtering by decade or government era.



TREEMAP CHARTS:

🧩 Treemap Chart – Power BI Overview (India Trade Data)

🔹 1. Purpose

- Visualize hierarchical or categorical data.
- Show proportions within a whole (e.g., top exporting sectors).
- Quickly compare the size of each category.

🔹 2. Data Setup (India Trade Example)

Use Case	Treemap Setup Example
📦 **Top Export Commodities**	Category: CommodityValues: Export Value
🌍 **Top Importing Countries**	Category: CountryValues: Import Value
📊 **Sector-wise Trade Share**	Group: SectorCategory: SubsectorValues: Trade value

🔹 3. Key Insights You Can Sho

- 🏭 Which sectors dominate India's exports (e.g., Petroleum, Textiles, IT).
- 🌎 Which countries are India's biggest trade partners.
- 🔄 Compare import vs export share by category.

🔹 4. Features

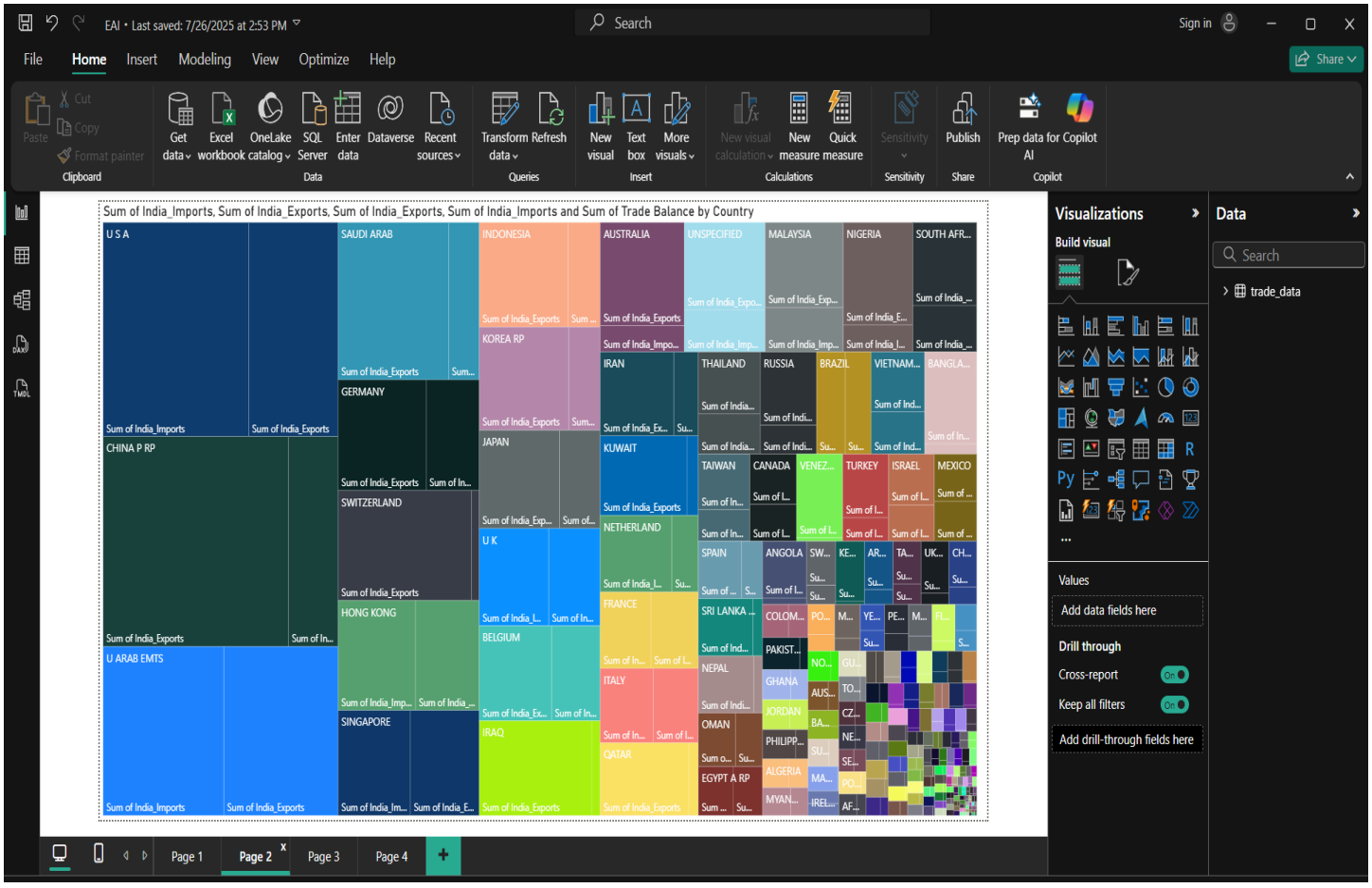
- Size of rectangle = value (e.g., export amount).
- Color can show:
    - Different categories (by default)
    - Conditional formatting (e.g., trade surplus in green, deficit in red).
- Tooltips show detailed data on hover.

5. Best Practices

- Use for relative comparison, not for trends.
- Combine with slicers (e.g., Year, Type: Import/Export).
- Limit the number of categories shown to avoid clutter.

6. Example Analysis Titles

- ✅ *"Country-wise Share of India's Imports (1997–2022)"*
- ✅ *"Sectoral Contribution to India's Export Growth"*



WATERFALL CHARTS:

🌊 Waterfall Chart – Power BI Overview (India Trade Data)

🔹 1. Purpose

- Shows incremental changes in values over time or across categories.
- Helps identify how a value increased or decreased step-by-step.
- Great for showing contributions to a total (e.g., trade balance).

🔹 2. Structure

- Start column: Beginning value (e.g., trade balance in 1997).
- Rising bars: Positive changes (e.g., export growth).
- Falling bars: Negative changes (e.g., import spikes).
- End column: Final total (e.g., trade balance in 2022).

🔹 3. Use Cases for India's Trade

Use Case	Description
📈 **Year-over-Year Trade Balance**	how trade surplus/deficit changed yearly.
🔄 **Import Cost Change Breakdown**	total change into sectors (e.g., oil, gold, electronics).
💰 **Export Revenue Growth Breakdown**	Identify which sectors contributed most to export increase.

🔹 4. Key Features

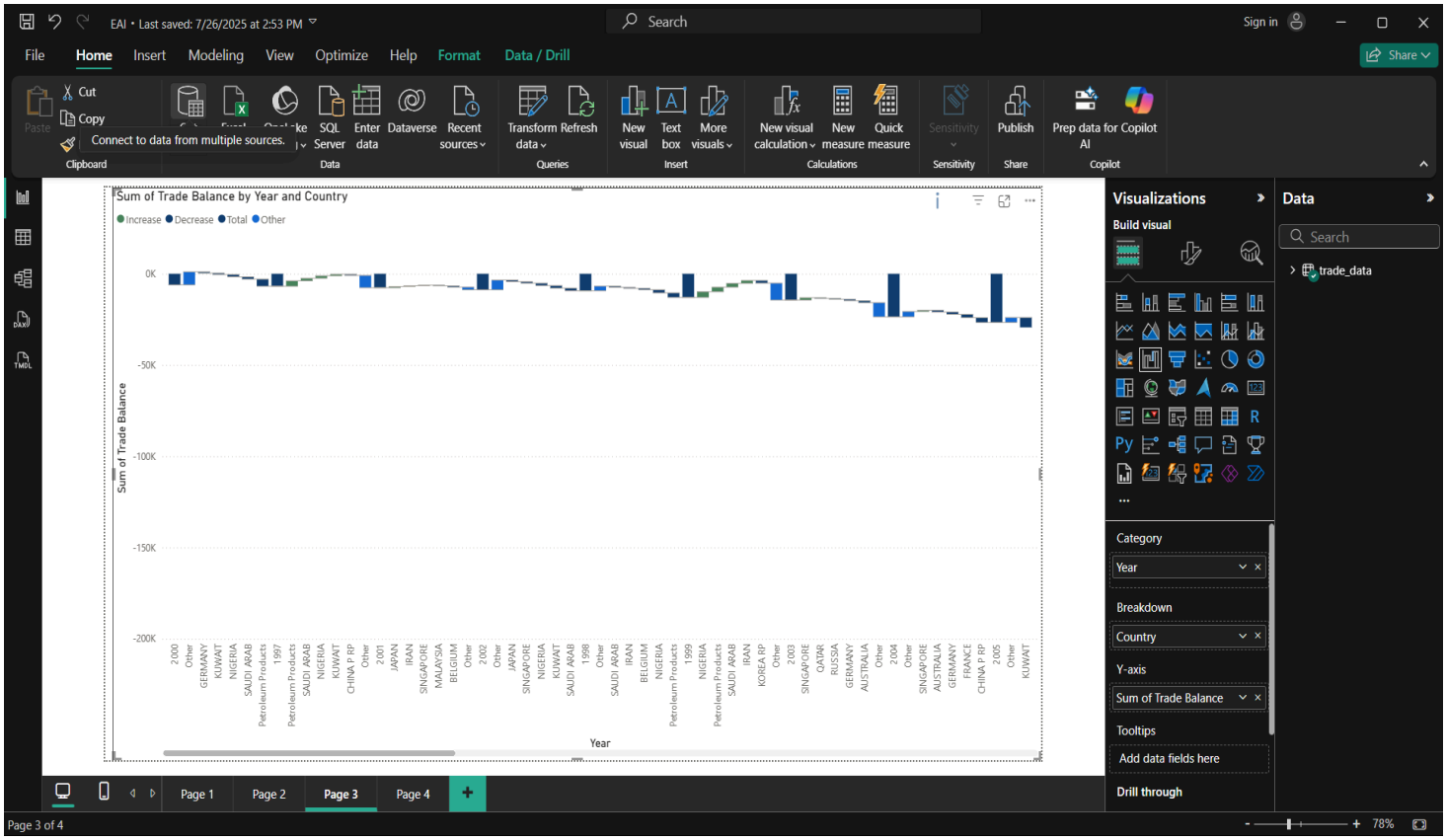
- Automatically calculates increases and decreases.
- Breakdown field: Allows category-wise contribution (e.g., sector-wise).
- Tooltips: Show exact values and deltas on hover.
- Total bars: Can be toggled on/off.

5. Best Practices

- Keep category names short for clarity.
- Use distinct colors for increase (green) and decrease (red).
- Highlight key categories or years with data labels.

6. Example Analysis Titles

- ✅ *"How India's Trade Deficit Widened from 1997 to 2022"*
- ✅ *"Contribution of Import Categories to Total Cost – 2022"*
- ✅ *"Year-wise Impact on Export Revenue Growth"*



MATRIX CHARTS:

🧮 Matrix Chart – Power BI Overview (India Trade Data)

🔹 1. Purpose

- Displays data in tabular format with the power of pivot tables.
- Allows you to show hierarchies, row/column groups, and aggregates.
- Ideal for detailed comparisons across multiple dimensions.

🔹 2. Structure

- Rows: Typically a category like *Year* or *Country*.
- Columns: Another category (e.g., Import/Export, Sector).
- Values: Numerical data (e.g., trade amount, growth %, etc.).

Use Case	Setup Example
Year-wise Export/Import Summary	YearColumns: Trade TypeValues: USD
🌍 **Country-wise Trade with India**	CountryColumns: Import/ExportValues: Amount
🏭 **Sector Contribution to Exports (by year)**	SectorColumns: YearValues: Export Value
📊 **Trade Balance Comparison**	YearColumns: Import vs ExportValues: Difference (Calculated Measure)

🔹 4. Key Features

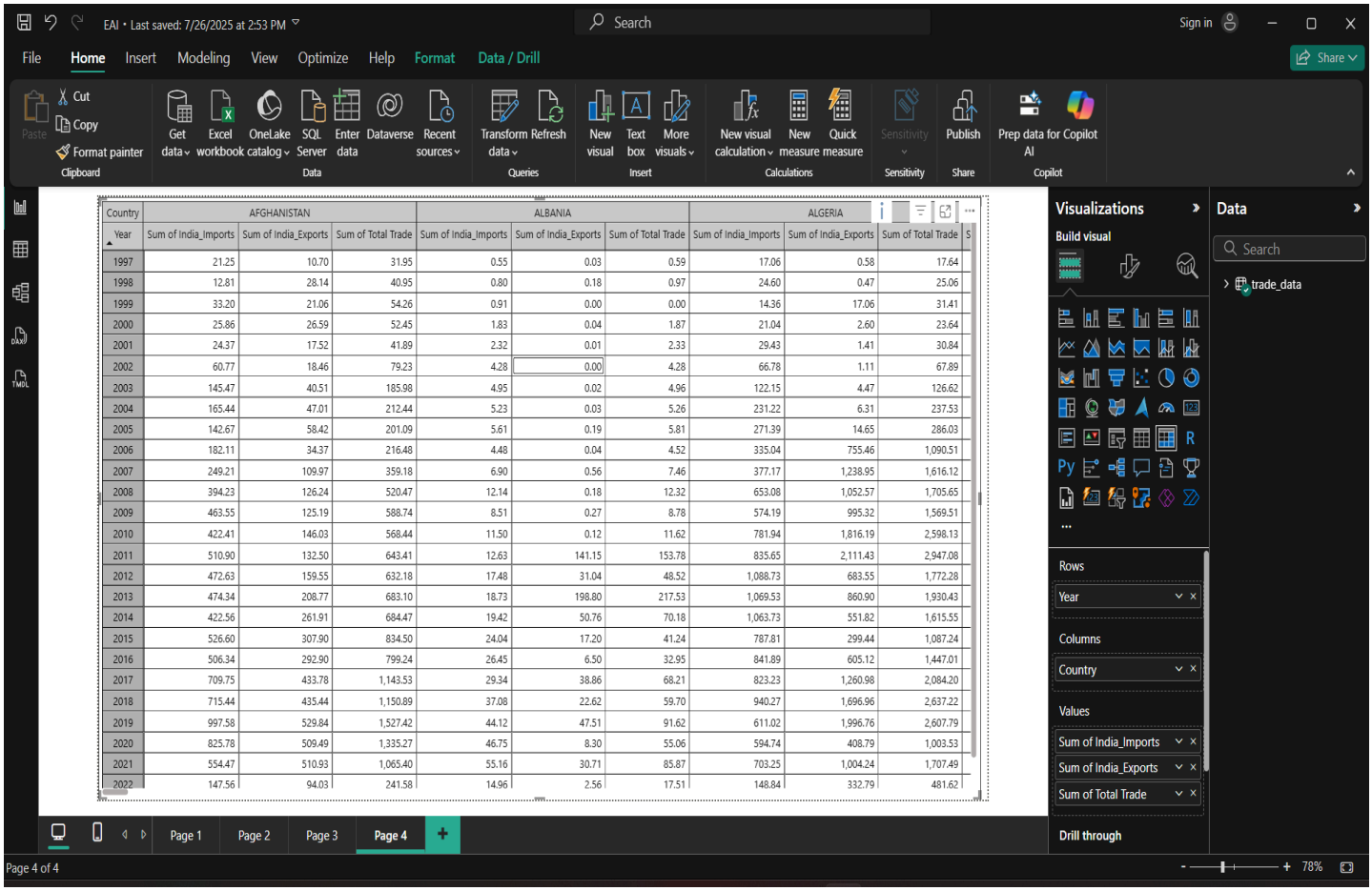
- Drill-down: Expand/collapse rows or columns for hierarchy (e.g., Sector → Subsector).
- Conditional Formatting: Add color scales or data bars.
- Totals/Subtotals: Automatically displayed and customizable.
- Sorting: By rows, columns, or values.
- Tooltips: Show detailed info on hover.

5. Best Practices

- Use column/row formatting to highlight key data.
- Hide unnecessary totals for clarity.
- Use hierarchical fields to reduce clutter.
- Avoid overcrowding with too many columns—consider slicers or filters.

6. Example Titles

- ✅ *"India's Year-wise Import & Export Matrix (1997–2022)"*
- ✅ *"Country-Wise Trade with India: A Matrix View"*
- ✅ *"Sector-Wise Export Trends Over Time"*



Country	AFGHANISTAN			ALBANIA			ALGERIA		
Year	Sum of India_Imports	Sum of India_Exports	Sum of Total Trade	Sum of India_Imports	Sum of India_Exports	Sum of Total Trade	Sum of India_Imports	Sum of India_Exports	Sum of Total Trade
1997	21.25	10.70	31.95	0.55	0.03	0.59	17.06	0.58	17.64
1998	12.81	28.14	40.95	0.80	0.18	0.97	24.60	0.47	25.06
1999	33.20	21.06	54.26	0.91	0.00	0.00	14.36	17.06	31.41
2000	25.86	26.59	52.45	1.83	0.04	1.87	21.04	2.60	23.64
2001	24.37	17.52	41.89	2.32	0.01	2.33	29.43	1.41	30.84
2002	60.77	18.46	79.23	4.28	0.00	4.28	66.78	1.11	67.89
2003	145.47	40.51	185.98	4.95	0.02	4.96	122.15	4.47	126.62
2004	165.44	47.01	212.44	5.23	0.03	5.26	231.22	6.31	237.53
2005	142.67	58.42	201.09	5.61	0.19	5.81	271.39	14.65	286.03
2006	182.11	34.37	216.48	4.48	0.04	4.52	335.04	755.46	1,090.51
2007	249.21	109.97	359.18	6.90	0.56	7.46	377.17	1,238.95	1,616.12
2008	394.23	126.24	520.47	12.14	0.18	12.32	653.08	1,052.57	1,705.65
2009	463.55	125.19	588.74	8.51	0.27	8.78	574.19	995.32	1,569.51
2010	422.41	146.03	568.44	11.50	0.12	11.62	781.94	1,816.19	2,598.13
2011	510.90	132.50	643.41	12.63	141.15	153.78	835.65	2,111.43	2,947.08
2012	472.63	159.55	632.18	17.48	31.04	48.52	1,088.73	683.55	1,772.28
2013	474.34	208.77	683.10	18.73	198.80	217.53	1,069.53	860.90	1,930.43
2014	422.56	261.91	684.47	19.42	50.76	70.18	1,063.73	551.82	1,615.55
2015	526.60	307.90	834.50	24.04	17.20	41.24	787.81	299.44	1,087.24
2016	506.34	292.90	799.24	26.45	6.50	32.95	841.89	605.12	1,447.01
2017	709.75	433.78	1,143.53	29.34	38.86	68.21	823.23	1,260.98	2,084.20
2018	715.44	435.44	1,150.89	37.08	22.62	59.70	940.27	1,696.96	2,637.22
2019	997.58	529.84	1,527.42	44.12	47.51	91.62	611.02	1,996.76	2,607.79
2020	825.78	509.49	1,335.27	46.75	8.30	55.06	594.74	408.79	1,003.53
2021	554.47	510.93	1,065.40	55.16	30.71	85.87	703.25	1,004.24	1,707.49
2022	147.56	94.03	241.58	14.96	2.56	17.51	148.84	332.79	481.62