Shipments & Compliance Analytics System

Prepared by:
Veman S Chippa

TABLE OF CONTENTS

1. **Introduction**
2. **Pivot Tables (Excel)**
3. **Dataset Description**
4. **Computed Fields**
5. **Power BI Dashboard**
6. **Database Design (MySQL DBMS)**
7. **Docker-Based Data Processing**
8. **Selenium Web Scraping**
9. **Python Data Manipulation**
10. **Visualization**
11. **Conclusion**

SECTION 1: Introduction

This project presents a complete end-to-end data engineering and analytics workflow designed to process, clean, analyze, and visualize export shipment data. The workflow integrates multiple tools and technologies, enabling efficient handling of raw data and converting it into meaningful insights.

The end-to-end workflow includes:

- Excel preprocessing and pivot table generation
- Creation of computed fields to standardize attributes
- Interactive visualization using Power BI
- Database modeling and SQL operations in MySQL
- Docker-based automation of the ETL process
- Selenium-based web scraping for external dataset extraction
- Python data cleaning, transformation, and statistical analysis
- Final insights through charts and visual summaries

This system demonstrates the complete data lifecycle:
Raw Data → Processing → Transformation → Modeling → Automation → Visualization.

Section 2: Pivot Tables (Excel)

Pivot Tables were created in Excel to summarize key shipment attributes such as **PORT Code**, **Gateway**, **IEC Extracted**, **GST**, and **SB Status**. These pivot summaries provide quick insights into shipment distribution, exporter activity, and compliance patterns.

Pivot Tables Prepared

1. **PORT Code vs Status – SB Count**
2. **Gateway vs GST – SB Count**
3. **IEC Extracted vs PORT Code – SB Count**
4. **PORT Code vs Gateway Extracted – SB Count**

Each pivot table uses:

- Row fields for grouping
- "Count of SB no" as the value metric
- Filters (Status, IEC, Gateway) where applicable
- Clean tabular layout for readability

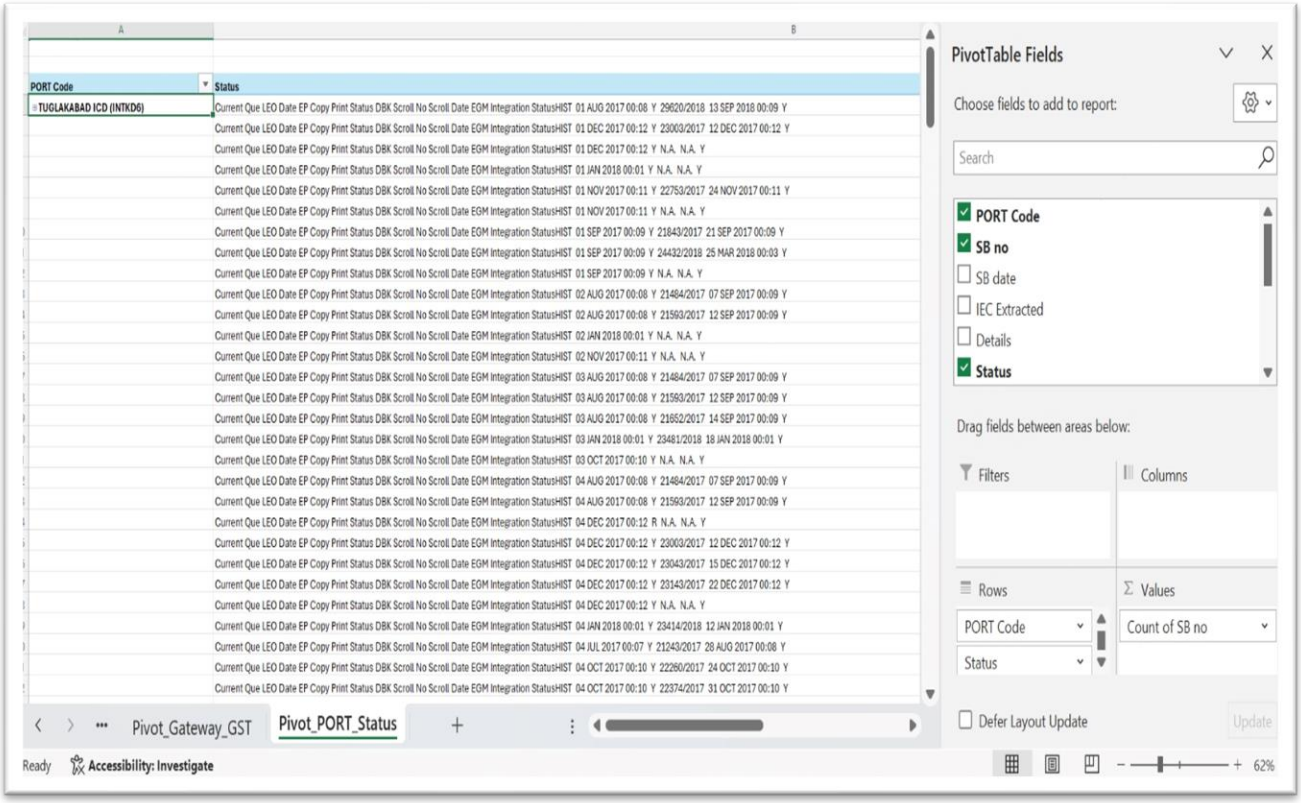


Figure 2.1: Pivot Table summarizing SB count by PORT Code and Status

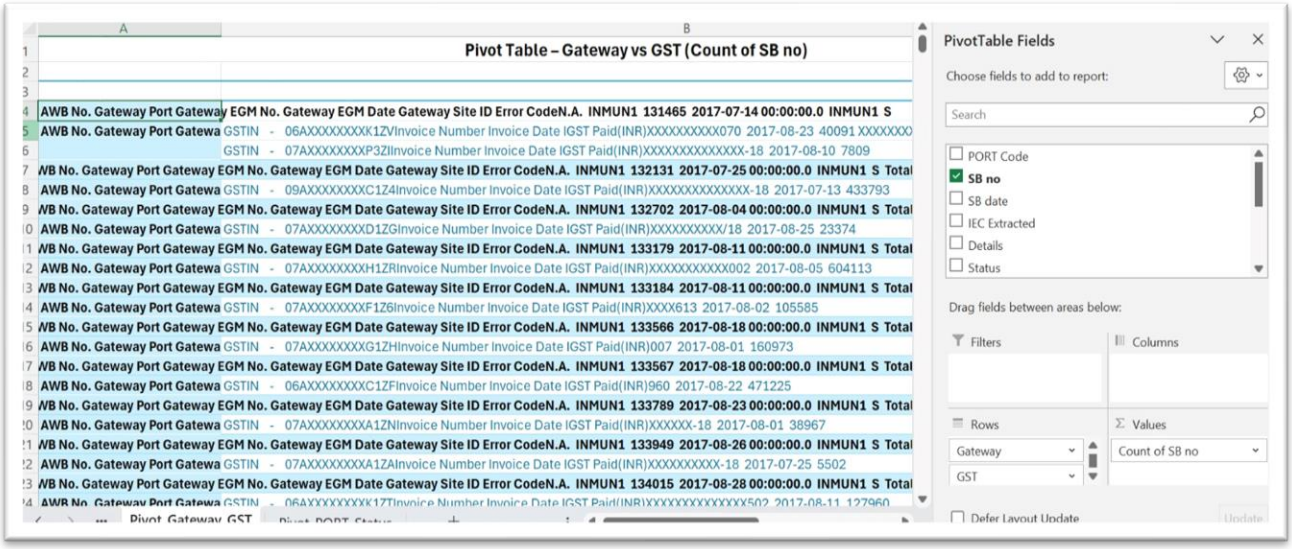
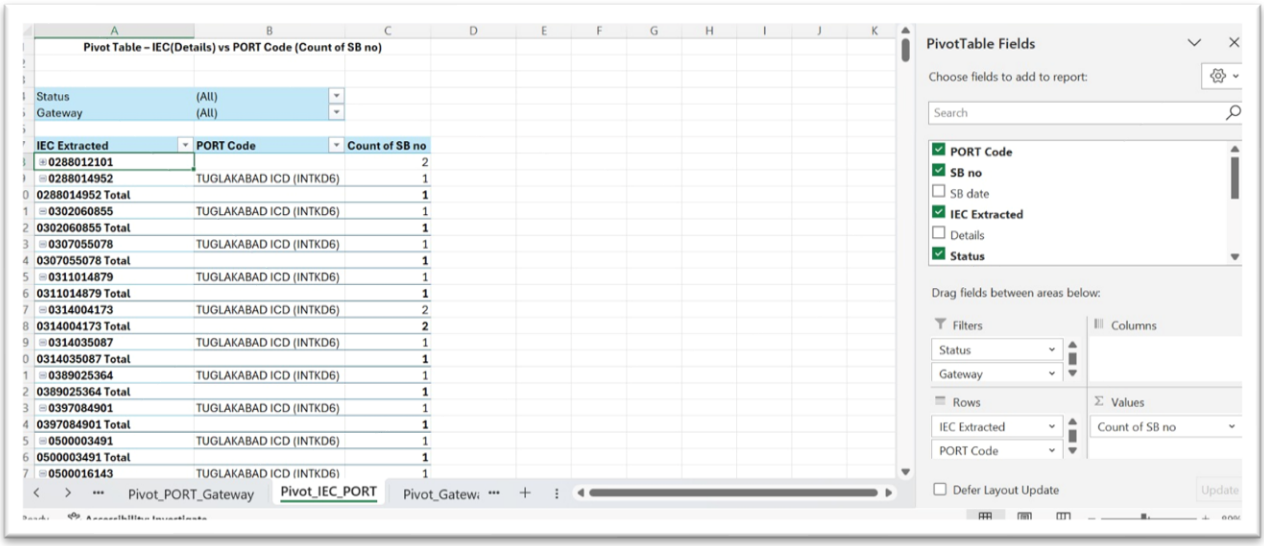


Figure 2.2: Pivot Table showing SB count grouped by Gateway and GST status

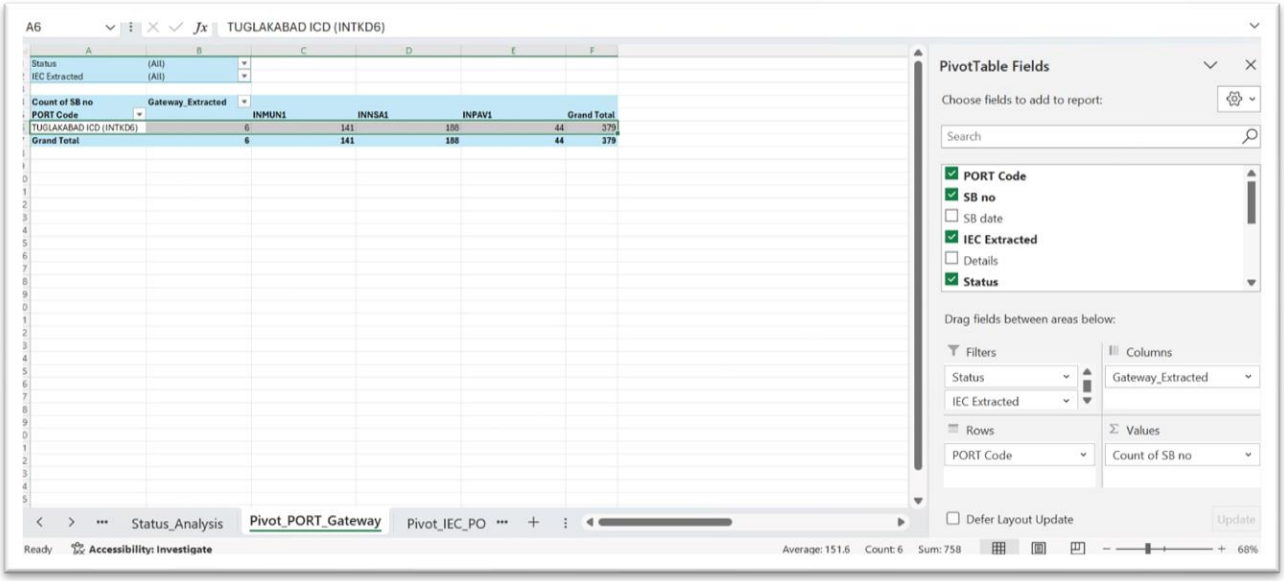


Pivot Table – IEC(Details) vs PORT Code (Count of SB no)

Status	(All)
Gateway	(All)

IEC Extracted	PORT Code	Count of SB no
0288012101		2
0288014952	TUGLAKABAD ICD (INTKD6)	1
0288014952 Total		1
0302060855	TUGLAKABAD ICD (INTKD6)	1
0302060855 Total		1
0307055078	TUGLAKABAD ICD (INTKD6)	1
0307055078 Total		1
0311014879	TUGLAKABAD ICD (INTKD6)	1
0311014879 Total		1
0314004173	TUGLAKABAD ICD (INTKD6)	2
0314004173 Total		2
0314035087	TUGLAKABAD ICD (INTKD6)	1
0314035087 Total		1
0389025364	TUGLAKABAD ICD (INTKD6)	1
0389025364 Total		1
0397084901	TUGLAKABAD ICD (INTKD6)	1
0397084901 Total		1
0500003491	TUGLAKABAD ICD (INTKD6)	1
0500003491 Total		1
0500016143	TUGLAKABAD ICD (INTKD6)	1

Figure 2.3: Pivot Table summarizing SB count by IEC Extracted and PORT Code



A6 TUGLAKABAD ICD (INTKD6)

Status	(All)
IEC Extracted	(All)

Count of SB no	Gateway_Extracted				
PORT Code	INMUN1	INNSA1	INPAV1		Grand Total
TUGLAKABAD ICD (INTKD6)	6	141	188	44	379
Grand Total	6	141	188	44	379

Figure 2.4: Pivot Table showing SB count by PORT Code and Gateway Extracted

SECTION 3: Dataset Description

The dataset contains shipment information exported from the customs system.

Main columns:

- PORT Code
- SB no
- SB date
- IEC Extracted
- GST
- Status
- EGM no
- ROSL
- Gateway Extracted / Clean

	A	B	C	D	E	F	G	H	I	J	K	L
1	PORT Code	SB no	SB date	Details	Current status	EGM no	ROSL	GST	Gateway			
2	TUGLAKABAD ICD (INT	7096937	2017/07/01	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXC1ZHInv	AWB No. Gateway Port Gateway EGM No. Gat			
3	TUGLAKABAD ICD (INT	7100605	2017/07/01	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXC1ZHInv	AWB No. Gateway Port Gateway EGM No. Gat			
4	TUGLAKABAD ICD (INT	7183827	2017/07/06	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07AXXXXXXXXM1Z2Inv	AWB No. Gateway Port Gateway EGM No. Gat			
5	TUGLAKABAD ICD (INT	7603758	2017/07/26	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXJ1ZGInvc	AWB No. Gateway Port Gateway EGM No. Gat			
6	TUGLAKABAD ICD (INT	7616933	2017/07/26	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXH1ZOInv	AWB No. Gateway Port Gateway EGM No. Gat			
7	TUGLAKABAD ICD (INT	7623801	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 09AXXXXXXXXC1Z4Inv	AWB No. Gateway Port Gateway EGM No. Gat			
8	TUGLAKABAD ICD (INT	7638661	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXL1ZEInvc	AWB No. Gateway Port Gateway EGM No. Gat			
9	TUGLAKABAD ICD (INT	7640213	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXH2ZJInvc	AWB No. Gateway Port Gateway EGM No. Gat			
10	TUGLAKABAD ICD (INT	7641404	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXH1ZOInv	AWB No. Gateway Port Gateway EGM No. Gat			
11	TUGLAKABAD ICD (INT	7641415	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXH1ZOInv	AWB No. Gateway Port Gateway EGM No. Gat			
12	TUGLAKABAD ICD (INT	7642279	2017/07/27	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXH1ZOInv	AWB No. Gateway Port Gateway EGM No. Gat			
13	TUGLAKABAD ICD (INT	7646561	2017/07/28	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07DXXXXXXXXB1ZHInv	AWB No. Gateway Port Gateway EGM No. Gat			
14	TUGLAKABAD ICD (INT	7649257	2017/07/28	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXK1ZTInvc	AWB No. Gateway Port Gateway EGM No. Gat			
15	TUGLAKABAD ICD (INT	7662086	2017/07/28	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXK1Z7Invc	AWB No. Gateway Port Gateway EGM No. Gat			
16	TUGLAKABAD ICD (INT	7690771	2017/07/29	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXQ1ZCInv	AWB No. Gateway Port Gateway EGM No. Gat			
17	TUGLAKABAD ICD (INT	7692707	2017/07/29	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 09AXXXXXXXXC2Z0Inv	AWB No. Gateway Port Gateway EGM No. Gat			
18	TUGLAKABAD ICD (INT	7703453	2017/07/31	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07AXXXXXXXXE1Z5Invc	AWB No. Gateway Port Gateway EGM No. Gat			
19	TUGLAKABAD ICD (INT	7712075	2017/07/31	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 08AXXXXXXXXR1ZRInvc	AWB No. Gateway Port Gateway EGM No. Gat			
20	TUGLAKABAD ICD (INT	7718281	2017/07/31	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07AXXXXXXXXQ1ZKInv	AWB No. Gateway Port Gateway EGM No. Gat			
21	TUGLAKABAD ICD (INT	7721241	2017/07/31	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 06AXXXXXXXXG1ZXInv	Please Wait....			
22	TUGLAKABAD ICD (INT	7742310	2017/08/01	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07AXXXXXXXXF1Z8Invc	AWB No. Gateway Port Gateway EGM No. Gat			
23	TUGLAKABAD ICD (INT	7747150	2017/08/01	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 09AXXXXXXXXK2ZTInvc	AWB No. Gateway Port Gateway EGM No. Gat			
24	TUGLAKABAD ICD (INT	7751210	2017/08/01	IEC CHA No. Job	Current Que LEO Dat	EGM No. EGM	ROSL Availed Amount(INR) Scr	GSTIN - 07AXXXXXXXXM1ZPInv	AWB No. Gateway Port Gateway EGM No. Gat			

Sheet1

Figure 3.1:: Raw provided Data

Section 4: Computed Fields

Five computed fields were created to standardize inconsistent dataset values and derive meaningful analytical categories.

1. **Clean_Status** – Processing / Pending / Other
2. **ROSL_Flag** – Released / Not Released
3. **EGM_Flag** – Submitted / Pending
4. **Gateway_Flag** – Valid / Invalid
5. **GST_StateCode** – First two digits extracted from GSTIN

Computed Field 1: Clean_Status	Computed Field 2:EGM_Flag	Computed Field 3:ROSL_Flag	Computed Field 4:EGM_Flag	Computed Field 5:Gateway_Flag
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid
Processed	EGM Submitted	Released	Submitted	Valid

Figure 4.1: Computed fields added to the dataset after preprocessing

Section 5: Power BI Dashboard

Multiple interactive visuals were created in Power BI to analyze shipment and compliance performance.

✓ KPI Cards:
Total SBs, Distinct IECs, Distinct PORTs, Distinct Gateways, Total EGMs

✓ Visuals:
Pie chart (Clean Status), Bar chart (GST State Code), PORT summary matrix, EGM distribution chart

Figure 5.1: Power BI dashboard showcasing shipment KPIs and compliance summaries

Section 6: Database Design (MySQL DBMS)

A structured relational database model was created in MySQL with four normalized tables:

1. shipment_info
2. exporter_details
3. compliance_details
4. status_details

✓ Entities & Relationships

- SB_no is the primary key for the main shipment table
- All supporting tables reference SB_no through foreign key constraints
- Referencing and relational integrity were validated using SQL queries
- The ER diagram visually represents table connectivity and schema logic

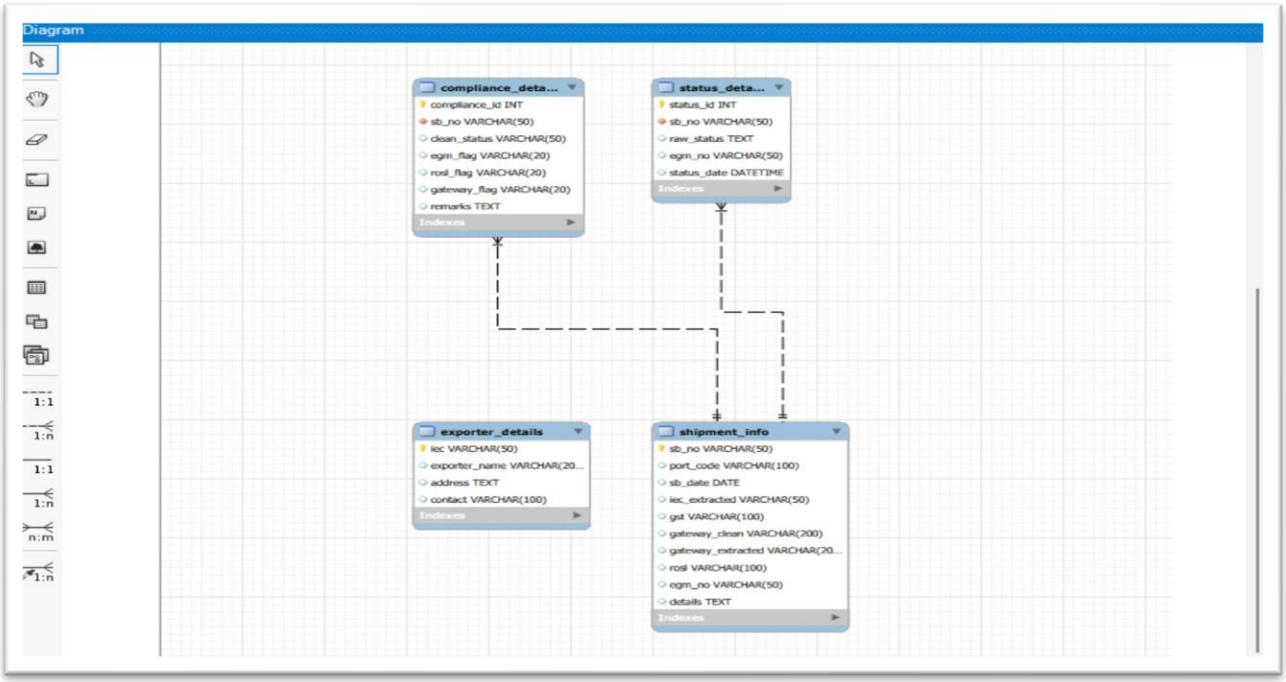


Figure 6.1: Entity–Relationship (ER) diagram designed in MySQL Workbench

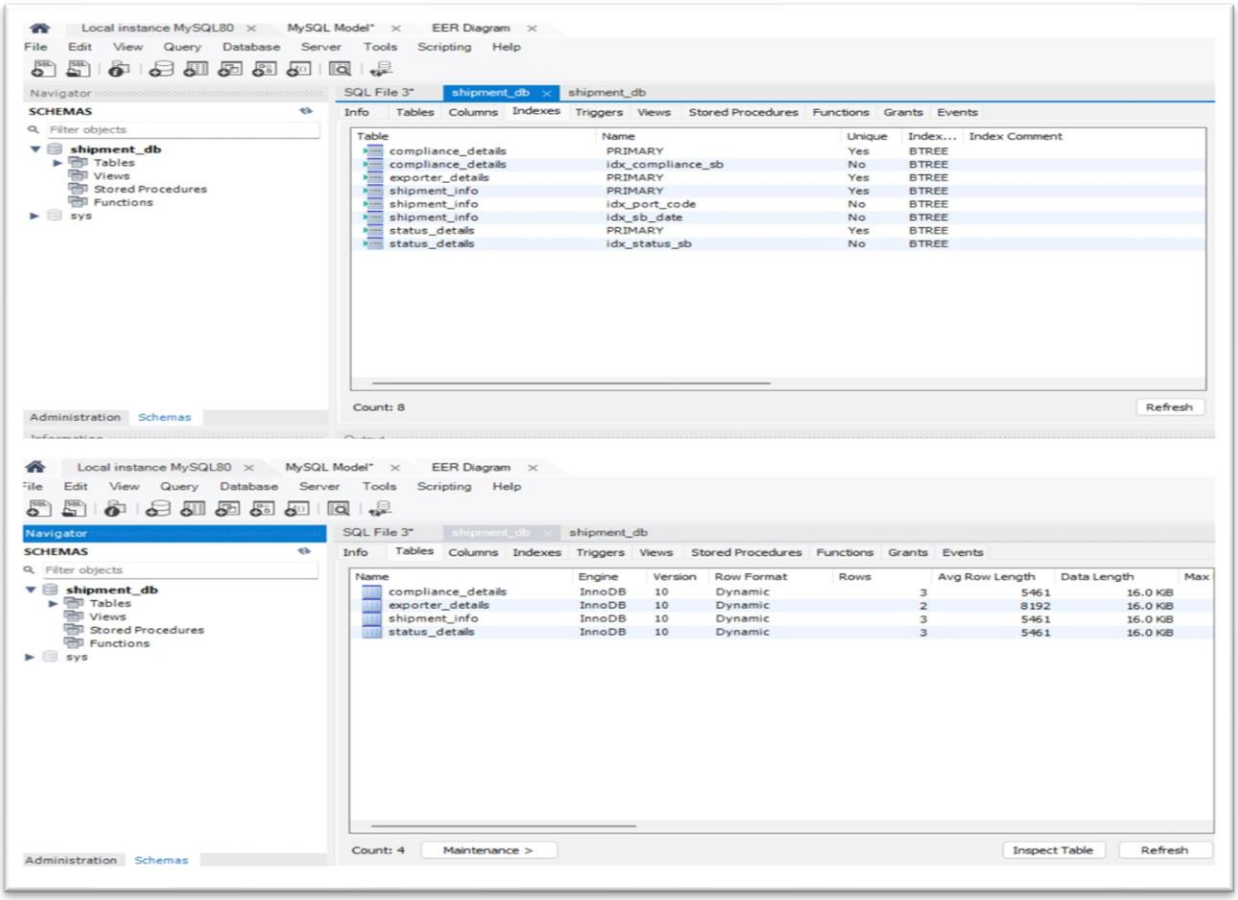
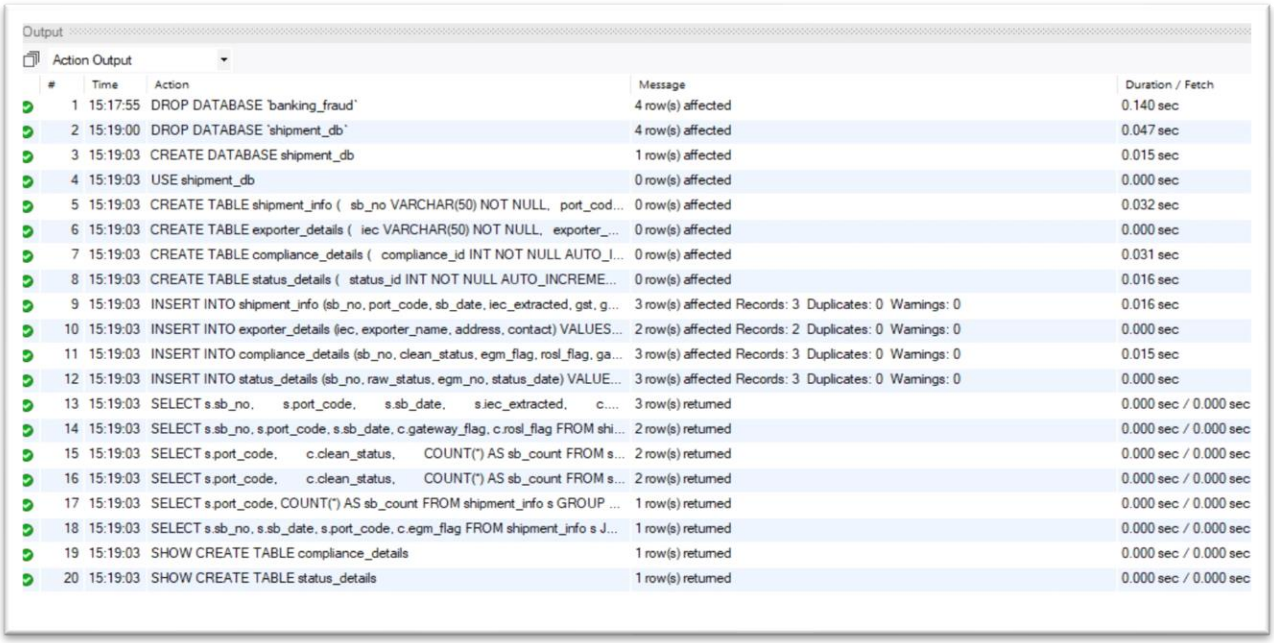


Figure 6.2: MySQL Workbench showing table creation, indexes, and storage details for all four database tables



Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:17:55	DROP DATABASE `banking_fraud`	4 row(s) affected	0.140 sec
2	15:19:00	DROP DATABASE `shipment_db`	4 row(s) affected	0.047 sec
3	15:19:03	CREATE DATABASE shipment_db	1 row(s) affected	0.015 sec
4	15:19:03	USE shipment_db	0 row(s) affected	0.000 sec
5	15:19:03	CREATE TABLE shipment_info (sb_no VARCHAR(50) NOT NULL, port_cod...	0 row(s) affected	0.032 sec
6	15:19:03	CREATE TABLE exporter_details (iec VARCHAR(50) NOT NULL, exporter_...	0 row(s) affected	0.000 sec
7	15:19:03	CREATE TABLE compliance_details (compliance_id INT NOT NULL AUTO_I...	0 row(s) affected	0.031 sec
8	15:19:03	CREATE TABLE status_details (status_id INT NOT NULL AUTO_INCREME...	0 row(s) affected	0.016 sec
9	15:19:03	INSERT INTO shipment_info (sb_no, port_code, sb_date, iec_extracted, gst, g...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
10	15:19:03	INSERT INTO exporter_details (iec, exporter_name, address, contact) VALUES...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
11	15:19:03	INSERT INTO compliance_details (sb_no, clean_status, egm_flag, rosl_flag, ga...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.015 sec
12	15:19:03	INSERT INTO status_details (sb_no, raw_status, egm_no, status_date) VALUE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
13	15:19:03	SELECT s.sb_no, s.port_code, s.sb_date, s.iec_extracted, c....	3 row(s) returned	0.000 sec / 0.000 sec
14	15:19:03	SELECT s.sb_no, s.port_code, s.sb_date, c.gateway_flag, c.rosl_flag FROM shi...	2 row(s) returned	0.000 sec / 0.000 sec
15	15:19:03	SELECT s.port_code, c.clean_status, COUNT(*) AS sb_count FROM s...	2 row(s) returned	0.000 sec / 0.000 sec
16	15:19:03	SELECT s.port_code, c.clean_status, COUNT(*) AS sb_count FROM s...	2 row(s) returned	0.000 sec / 0.000 sec
17	15:19:03	SELECT s.port_code, COUNT(*) AS sb_count FROM shipment_info s GROUP ...	1 row(s) returned	0.000 sec / 0.000 sec
18	15:19:03	SELECT s.sb_no, s.sb_date, s.port_code, c.egm_flag FROM shipment_info s J...	1 row(s) returned	0.000 sec / 0.000 sec
19	15:19:03	SHOW CREATE TABLE compliance_details	1 row(s) returned	0.000 sec / 0.000 sec
20	15:19:03	SHOW CREATE TABLE status_details	1 row(s) returned	0.000 sec / 0.000 sec

Figure 6.3: SQL action log showing successful database creation, table creation, data insertion, and query execution in MySQL Workbench

Section 7: Docker-Based Data Processing

A fully containerized data-cleaning workflow was developed using Docker to automate the ETL process.

Files Used

- **Dockerfile**
- **process_data.py** (Python cleaning script)
- **requirements.txt**
- **input.xlsx** stored in /data directory

Key Features

- Loads raw Excel file inside the container
- Cleans and transforms the dataset using Python
- Generates cleaned_shipments.csv
- Saves the output automatically to the /output directory

Commands Executed

```
docker build -t shipments-cleaner:latest .

docker run --rm -v "${PWD}/data:/data" -v "${PWD}/output:/output" shipments-cleaner:latest
```

```
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments> ^C
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments> ^C
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments> docker build -t shipments-cleaner:latest .
[+] Building 41.5s (11/11) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                 0.0s
 => => transferring dockerfile: 478B                                                                                 0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                  3.7s
 => [internal] load .dockerignore                                                                                    0.0s
 => => transferring context: 2B                                                                                      0.0s
 => [1/6] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d  12.3s
 => => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c47474444d  0.0s
 => => sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55 14.36MB / 14.36MB                     4.3s
 => => sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da 251B / 251B                           1.0s
 => => sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974 1.29MB / 1.29MB                       4.5s
 => => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB                    10.7s
 => => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db                            0.8s
 => => extracting sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974                            0.1s
 => => extracting sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55                            0.5s
 => => extracting sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da                            0.0s
 => [internal] load build context                                                                                    0.1s
 => => transferring context: 1.26kB                                                                                  0.0s
 => [2/6] WORKDIR /app                                                                                               0.1s
 => [3/6] COPY requirements.txt .                                                                                    0.0s
 => [4/6] RUN pip install --no-cache-dir -r requirements.txt                                                        16.4s
 => [5/6] COPY process_data.py .                                                                                     0.1s
 => [6/6] RUN mkdir -p /data /output                                                                                 0.4s
 => exporting to image                                                                                               8.2s
 => => exporting layers                                                                                              6.4s
 => => exporting manifest sha256:c3449c6c7aa331c7835d1ebcc02c0becff46cf6f4e0024de478c633cb04c9aed                    0.0s
 => => exporting config sha256:090b82400b99ec3462c087224aca8d8d0b4ad0ed7a8b50d1fa8f21ae85049ec3                      0.0s
 => => exporting attestation manifest sha256:f2362a248e2f8a0bde1b79db2bf078ec375f62f669e8f76cf5ddbd953f3c1fef        0.0s
 => => exporting manifest list sha256:037f23897abe354fb15b9705430c182f7bc482dee9963255a9c4f1a0b75df1d0               0.0s
 => => naming to docker.io/library/shipments-cleaner:latest                                                          0.0s
 => => unpacking to docker.io/library/shipments-cleaner:latest                                                       1.7s
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments>
```

Figure 7.1: Docker image built successfully using the provided Dockerfile

```
PS C:\Users\Veman S Chippa> cd "C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments"
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments> docker run --rm -v "${PWD}/data:/data" -v "${PWD}/output:/output" shipm
ents-cleaner:latest
Saved cleaned file to: /output/cleaned_shipments.csv
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments> docker images
                                                                                                  Info →   U  In Use
IMAGE                      ID             DISK USAGE   CONTENT SIZE   EXTRA
shipments-cleaner:latest   037f23897abe        418MB           95MB
PS C:\Users\Veman S Chippa\OneDrive\Desktop\docker_shipments>
```

Figure 7.2: Docker container run showing "Saved cleaned file" message

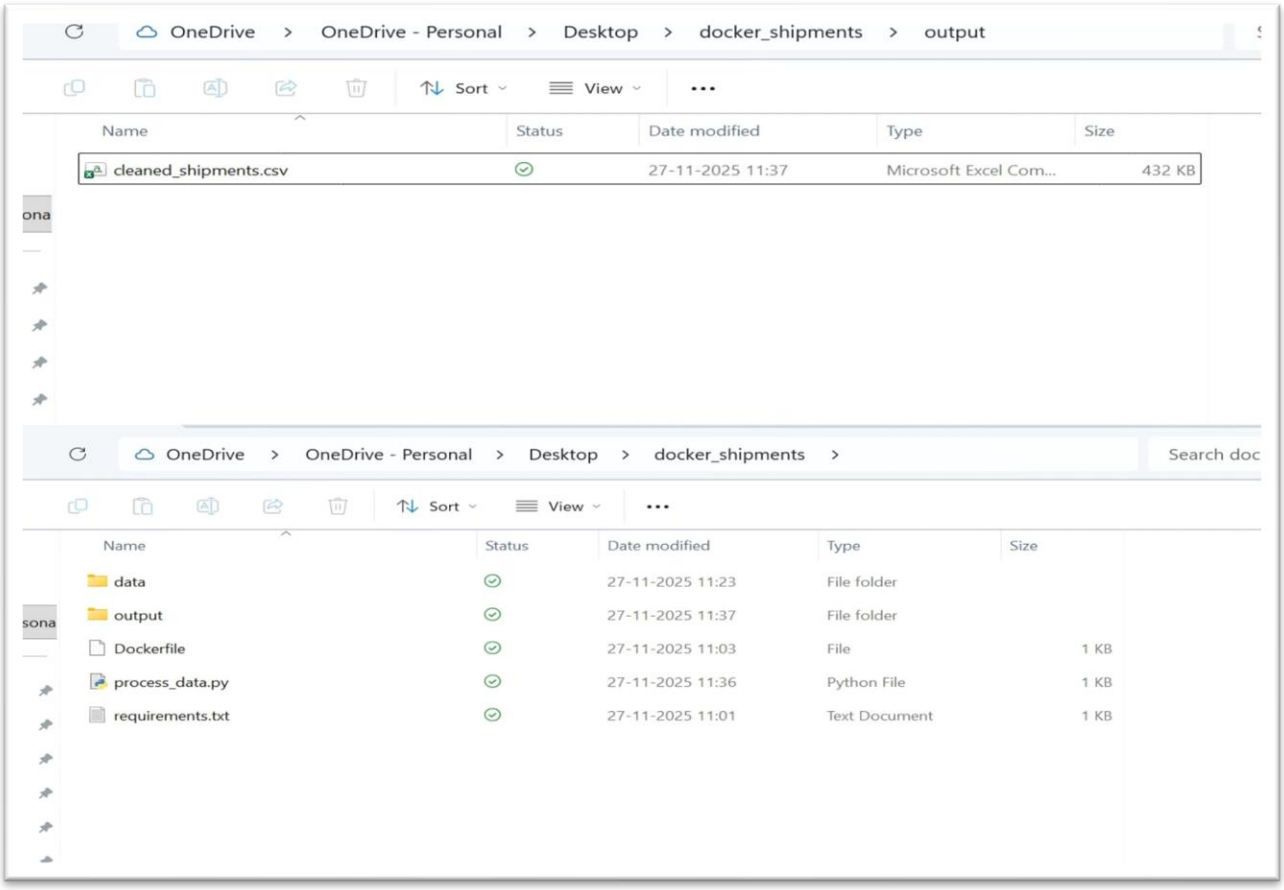


Figure 7.3: Generated cleaned_shipments.csv created inside the /output folder

Section 8: Selenium Web Scraping

A Selenium automation script was developed to extract structured tabular data from the Wikipedia page **"List of Countries by Population."**

Steps Performed

- Opened the webpage using Selenium WebDriver
- Located and extracted all table rows
- Parsed the table into structured columns
- Converted the extracted data into a Pandas DataFrame
- Saved the final output as: **scraped_wikipedia_population.csv**

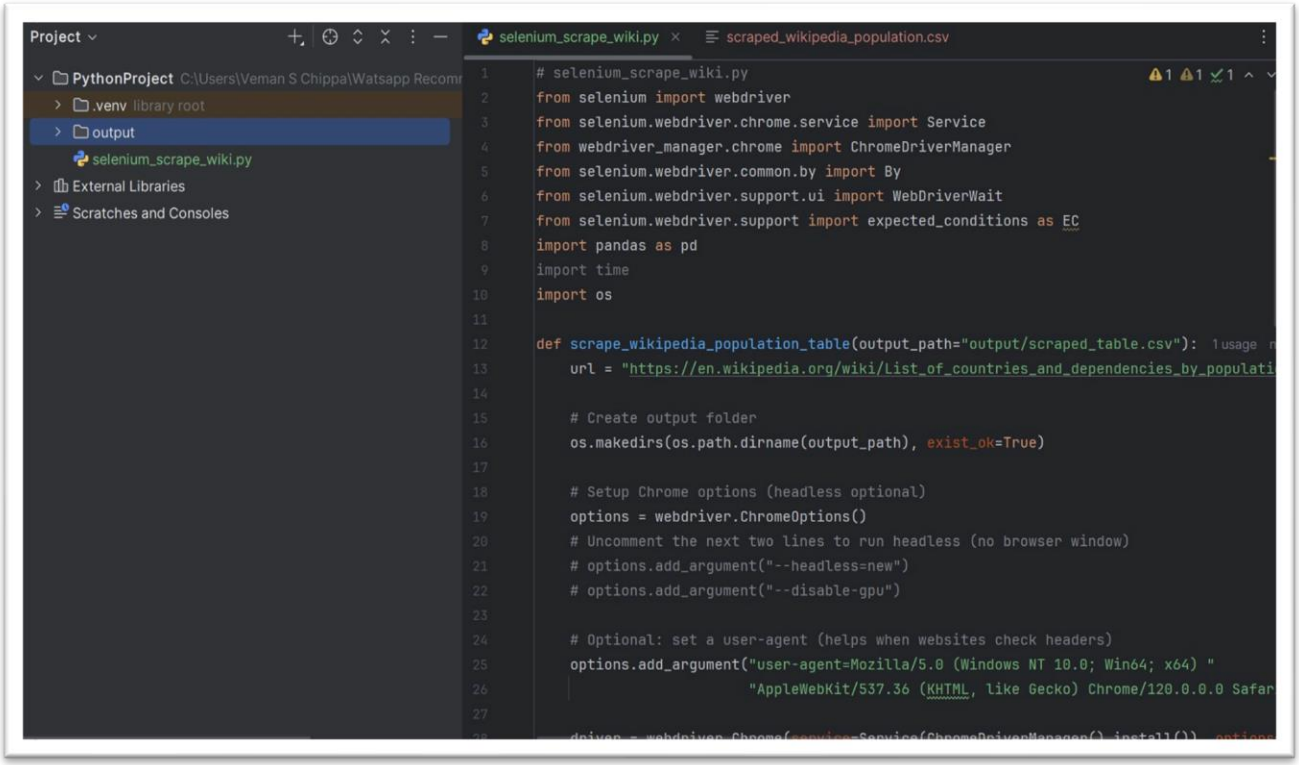
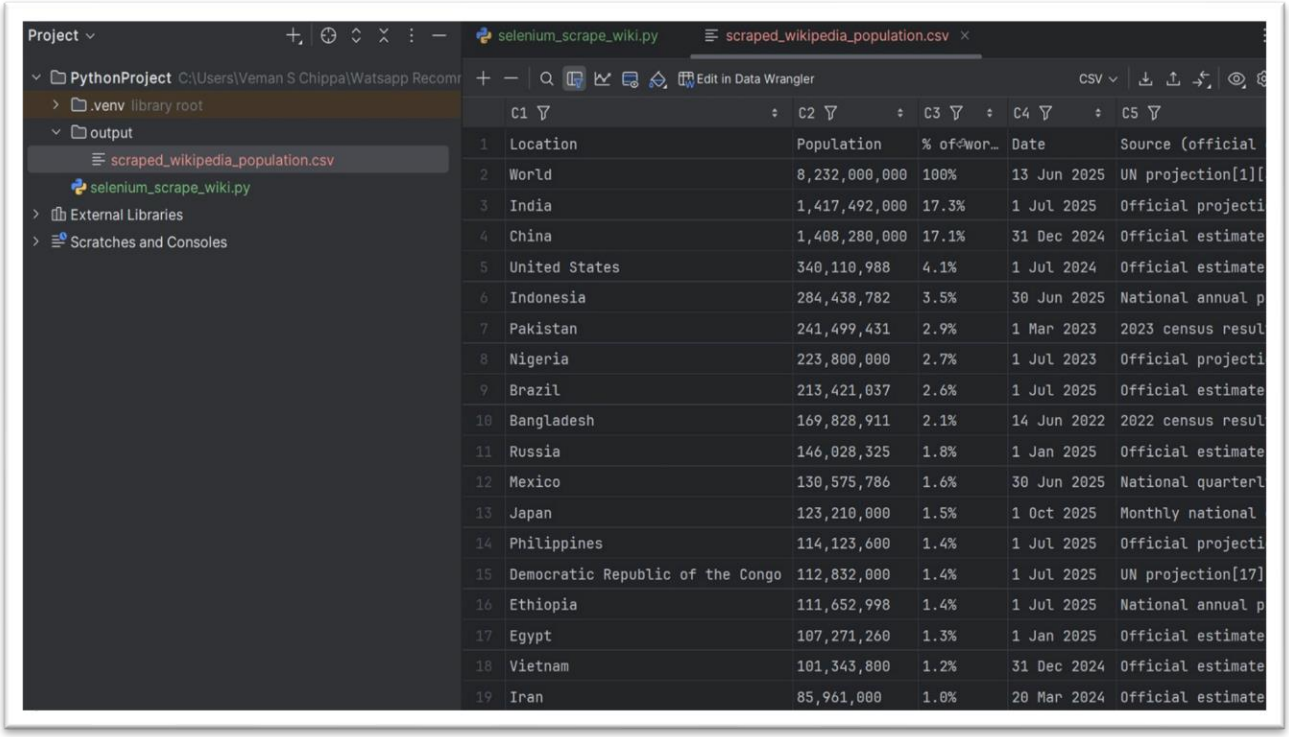


Figure 8.1 — Selenium script in PyCharm

```
indows PowerShell
opyright (C) Microsoft Corporation. All rights reserved.

nstall the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

.venv) PS C:\Users\Veman S Chippa\Watsapp Recommendation System\PythonProject> pip install selenium webdriver-manager pandas openpyxl
ollecting selenium
 Downloading selenium-4.38.0-py3-none-any.whl.metadata (7.5 kB)
ollecting webdriver-manager
 Downloading webdriver_manager-4.0.2-py2.py3-none-any.whl.metadata (12 kB)
ollecting pandas
 Using cached pandas-2.3.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
ollecting openpyxl
 Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
ollecting urllib3<3.0,>=2.5.0 (from urllib3[socks]<3.0,>=2.5.0->selenium)
 Downloading urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
ollecting trio<1.0,>=0.31.0 (from selenium)
 Downloading trio-0.32.0-py3-none-any.whl.metadata (8.5 kB)
ollecting trio-websocket<1.0,>=0.12.2 (from selenium)
 Downloading trio_websocket-0.12.2-py3-none-any.whl.metadata (5.1 kB)
ollecting certifi>=2025.10.5 (from selenium)
 Using cached certifi-2025.11.12-py3-none-any.whl.metadata (2.5 kB)
ollecting typing_extensions<5.0,>=4.15.0 (from selenium)
 Downloading typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
ollecting websocket-client<2.0,>=1.8.0 (from selenium)
 Downloading websocket_client-1.9.0-py3-none-any.whl.metadata (8.3 kB)
ollecting attrs>=23.2.0 (from trio<1.0,>=0.31.0->selenium)
 Downloading attrs-25.4.0-py3-none-any.whl.metadata (10 kB)
ollecting sortedcontainers (from trio<1.0,>=0.31.0->selenium)
 Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl.metadata (10 kB)
ollecting idna (from trio<1.0,>=0.31.0->selenium)
 Using cached idna-3.11-py3-none-any.whl.metadata (8.4 kB)
ollecting outcome (from trio<1.0,>=0.31.0->selenium)
```

```
Using cached certifi-2025.11.12-py3-none-any.whl (159 kB)
Downloading cffi-2.0.0-cp313-cp313-win_amd64.whl (183 kB)
Using cached numpy-2.3.5-cp313-cp313-win_amd64.whl (12.8 MB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached pytz-2025.2-py2.py3-none-any.whl (509 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Downloading sniffio-1.3.1-py3-none-any.whl (10 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Downloading wsproto-1.3.2-py3-none-any.whl (24 kB)
Downloading h11-0.16.0-py3-none-any.whl (37 kB)
Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Using cached idna-3.11-py3-none-any.whl (71 kB)
Using cached packaging-25.0-py3-none-any.whl (66 kB)
Downloading pycparser-2.23-py3-none-any.whl (118 kB)
Downloading python_dotenv-1.2.1-py3-none-any.whl (21 kB)
Downloading requests-2.32.5-py3-none-any.whl (64 kB)
Downloading charset_normalizer-3.4.4-cp313-cp313-win_amd64.whl (107 kB)
Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Installing collected packages: sortedcontainers, pytz, websocket-client, urllib3, tzdata, typing_extensions, sniffio, six, python-dotenv, pysocks,
pycparser, packaging, numpy, idna, h11, et-xmlfile, charset_normalizer, certifi, attrs, wsproto, requests, python-dateutil, outcome, openpyxl, cff
, webdriver-manager, trio, pandas, trio-websocket, selenium
Successfully installed attrs-25.4.0 certifi-2025.11.12 cffi-2.0.0 charset_normalizer-3.4.4 et-xmlfile-2.0.0 h11-0.16.0 idna-3.11 numpy-2.3.5 openp
xl-3.1.5 outcome-1.3.0.post0 packaging-25.0 pandas-2.3.3 pycparser-2.23 pysocks-1.7.1 python-dateutil-2.9.0.post0 python-dotenv-1.2.1 pytz-2025.2
equests-2.32.5 selenium-4.38.0 six-1.17.0 sniffio-1.3.1 sortedcontainers-2.4.0 trio-0.32.0 trio-websocket-0.12.2 typing_extensions-4.15.0 tzdata-20
25.2 urllib3-2.5.0 webdriver-manager-4.0.2 websocket-client-1.9.0 wsproto-1.3.2

[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\Veman S Chippa\Watsapp Recommendation System\PythonProject>
(.venv) PS C:\Users\Veman S Chippa\Watsapp Recommendation System\PythonProject> python selenium_scrape_wiki.py
Saved scraped table to: output/scraped_wikipedia_population.csv
```

Figure 8.2 — Installing Selenium & libraries



	C1	C2	C3	C4	C5
1	Location	Population	% of wor…	Date	Source (official
2	World	8,232,000,000	100%	13 Jun 2025	UN projection[1][
3	India	1,417,492,000	17.3%	1 Jul 2025	Official projecti
4	China	1,408,280,000	17.1%	31 Dec 2024	Official estimate
5	United States	340,110,988	4.1%	1 Jul 2024	Official estimate
6	Indonesia	284,438,782	3.5%	30 Jun 2025	National annual p
7	Pakistan	241,499,431	2.9%	1 Mar 2023	2023 census resul
8	Nigeria	223,800,000	2.7%	1 Jul 2023	Official projecti
9	Brazil	213,421,037	2.6%	1 Jul 2025	Official estimate
10	Bangladesh	169,828,911	2.1%	14 Jun 2022	2022 census resul
11	Russia	146,028,325	1.8%	1 Jan 2025	Official estimate
12	Mexico	130,575,786	1.6%	30 Jun 2025	National quarterl
13	Japan	123,210,000	1.5%	1 Oct 2025	Monthly national
14	Philippines	114,123,600	1.4%	1 Jul 2025	Official projecti
15	Democratic Republic of the Congo	112,832,000	1.4%	1 Jul 2025	UN projection[17]
16	Ethiopia	111,652,998	1.4%	1 Jul 2025	National annual p
17	Egypt	107,271,260	1.3%	1 Jan 2025	Official estimate
18	Vietnam	101,343,800	1.2%	31 Dec 2024	Official estimate
19	Iran	85,961,000	1.0%	20 Mar 2024	Official estimate

Figure 8.3 — Scraped CSV output

Section 9: Python Data Manipulation

Python-based data cleaning and transformation were performed using **Jupyter Notebook** to prepare the scraped dataset for analysis.

Tasks Performed

- Loaded the Selenium-extracted CSV file
- Cleaned column names and removed unwanted characters
- Converted numeric fields (population, percentages) into proper data types
- Applied sorting, filtering, and transformation operations
- Generated summary statistics for key indicators

jupyter data_cleaning_final.ipynb Last Checkpoint: 53 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykernel)

[18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[4]:
```
df = pd.read_csv(r"C:\Users\Veman S Chippa\OneDrive\Desktop\scraped_wikipedia_population.csv")
df.head()
```

[4]:

	Location	Population	% of\nworld	Date	Source (official or from\nthe United Nations)	Notes
0	World	8,232,000,000	100%	13 Jun 2025	UN projection[1][3]	NaN
1	India	1,417,492,000	17.3%	1 Jul 2025	Official projection[4]	[b]
2	China	1,408,280,000	17.1%	31 Dec 2024	Official estimate[5]	[c]
3	United States	340,110,988	4.1%	1 Jul 2024	Official estimate[6]	[d]
4	Indonesia	284,438,782	3.5%	30 Jun 2025	National annual projection[7]	NaN

[6]:
```
df = df.drop_duplicates()
```

[7]:
```
df.columns = df.columns.str.strip()
```

[8]:
```
df['Population'] = (df['Population']
                    .str.replace(',', '')
                    .astype(int))
```

[9]:
```
df = df.fillna("Not Available")
```

[10]:
```
world_total = df['Population'].sum()

df['World_%'] = (df['Population'] / world_total * 100).round(4)
```

[11]:
```
df_sorted = df.sort_values(by="Population", ascending=False)
```

[16]:
```
from pathlib import Path

# Use raw string for Windows path
output_path = Path(r"C:\Users\Veman S Chippa\OneDrive\Desktop\Output\cleaned_population_data.csv")

# Save DataFrame to CSV
df_sorted.to_csv(output_path, index=False)
```

[17]:
```
output_path
```

Figure 9.1: Python data-cleaning and transformation steps executed in Jupyter Notebook

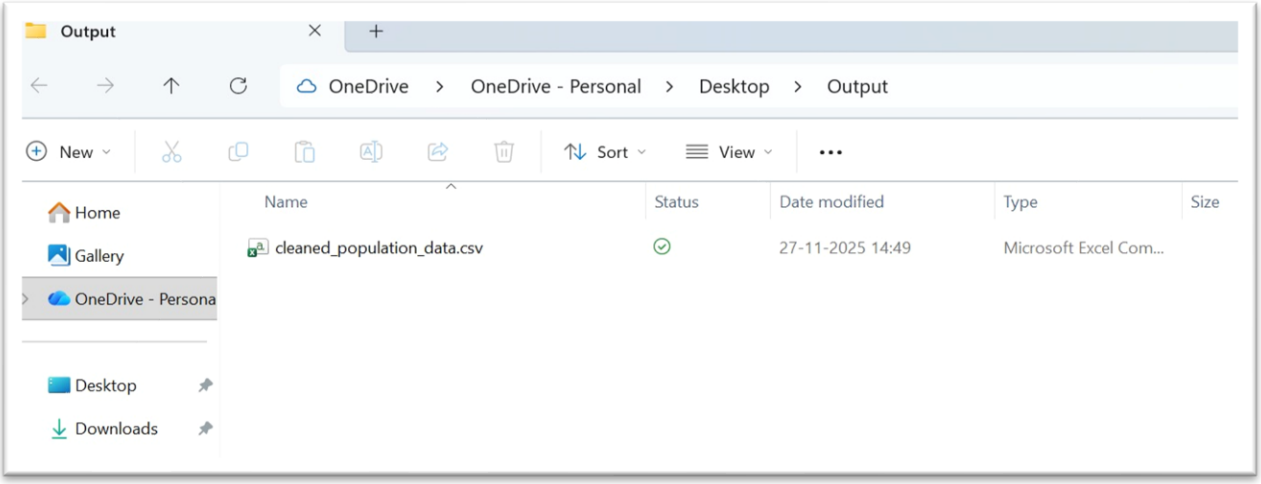


Figure 9.2: Cleaned CSV file generated in the output folder

I4 D

	A	B	C	D	E	F	G	H
1	Location	Population	% of	Date	Source (official or from	Notes	World_%	
2	World	8232000000	100%	13-Jun-25	UN projection[1][3]	Not Available	50.657	
3	India	1417492000	17.30%	01-Jul-25	Official projection[4]	[b]	8.7228	
4	China	1408280000	17.10%	31-Dec-24	Official estimate[5]	[c]	8.6661	
5	United Sta	340110988	4.10%	01-Jul-24	Official estimate[6]	[d]	2.0929	
6	Indonesia	284438782	3.50%	30-Jun-25	National annual projection[7]	Not Available	1.7503	
7	Pakistan	241499431	2.90%	01-Mar-23	2023 census result[8]	[e]	1.4861	
8	Nigeria	223800000	2.70%	01-Jul-23	Official projection[9]	Not Available	1.3772	
9	Brazil	213421037	2.60%	01-Jul-25	Official estimate[10]	Not Available	1.3133	
10	Banglades	169828911	2.10%	14-Jun-22	2022 census result[11]	[f]	1.0451	
11	Russia	146028325	1.80%	01-Jan-25	Official estimate[13]	[g]	0.8986	
12	Mexico	130575786	1.60%	30-Jun-25	National quarterly estimate[14]	Not Available	0.8035	
13	Japan	123210000	1.50%	01-Oct-25	Monthly national estimate[15]	Not Available	0.7582	
14	Philippines	114123600	1.40%	01-Jul-25	Official projection[16]	Not Available	0.7023	
15	Democrati	112832000	1.40%	01-Jul-25	UN projection[17]	Not Available	0.6943	
16	Ethiopia	111652998	1.40%	01-Jul-25	National annual projection[18]	Not Available	0.6871	
17	Egypt	107271260	1.30%	01-Jan-25	Official estimate[19]	Not Available	0.6601	
18	Vietnam	101343800	1.20%	31-Dec-24	Official estimate[20]	Not Available	0.6236	
19	Iran	85961000	1.00%	20-Mar-24	Official estimate[21]	Not Available	0.529	
20	Turkey	85664944	1.00%	31-Dec-24	Official estimate[22]	Not Available	0.5272	
21	Germany	83491249	1.00%	30-Jun-25	National quarterly estimate[23]	Not Available	0.5138	
22	United Kin	69281437	0.80%	30-Jun-24	Official estimate[24]	[h]	0.4263	

cleaned_population_data

Figure 9.3: Cleaned population dataset opened in Excel after processing

Section 10: Visualization

A final visualization was created using the cleaned Selenium-extracted dataset to highlight population patterns across countries.

Examples of Visuals

- **Top 10 most populated countries** (bar chart)
- **Population distribution by major regions or countries** (pie chart)

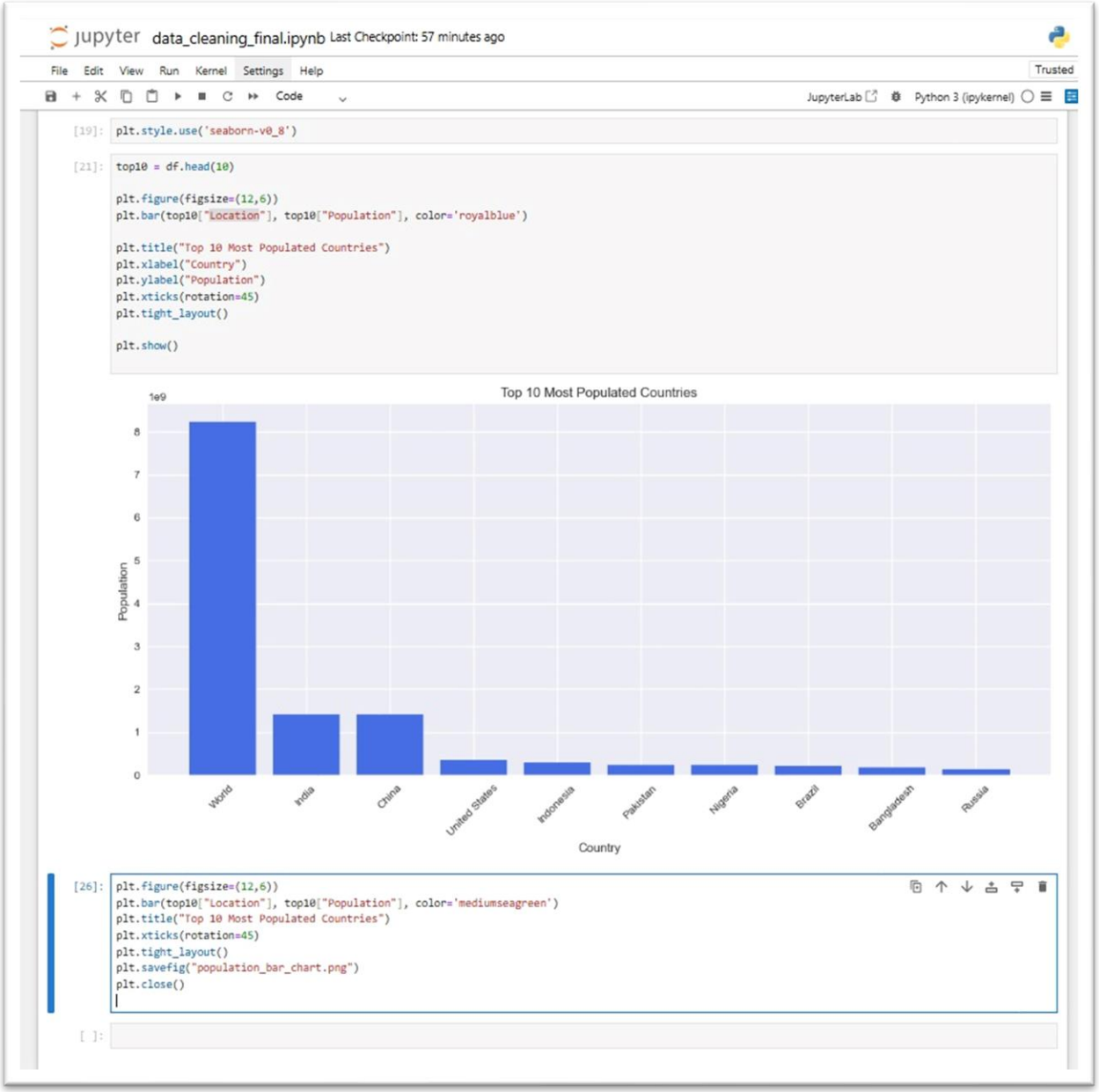


Figure 10.1: Bar chart showing the Top 10 most populated countries

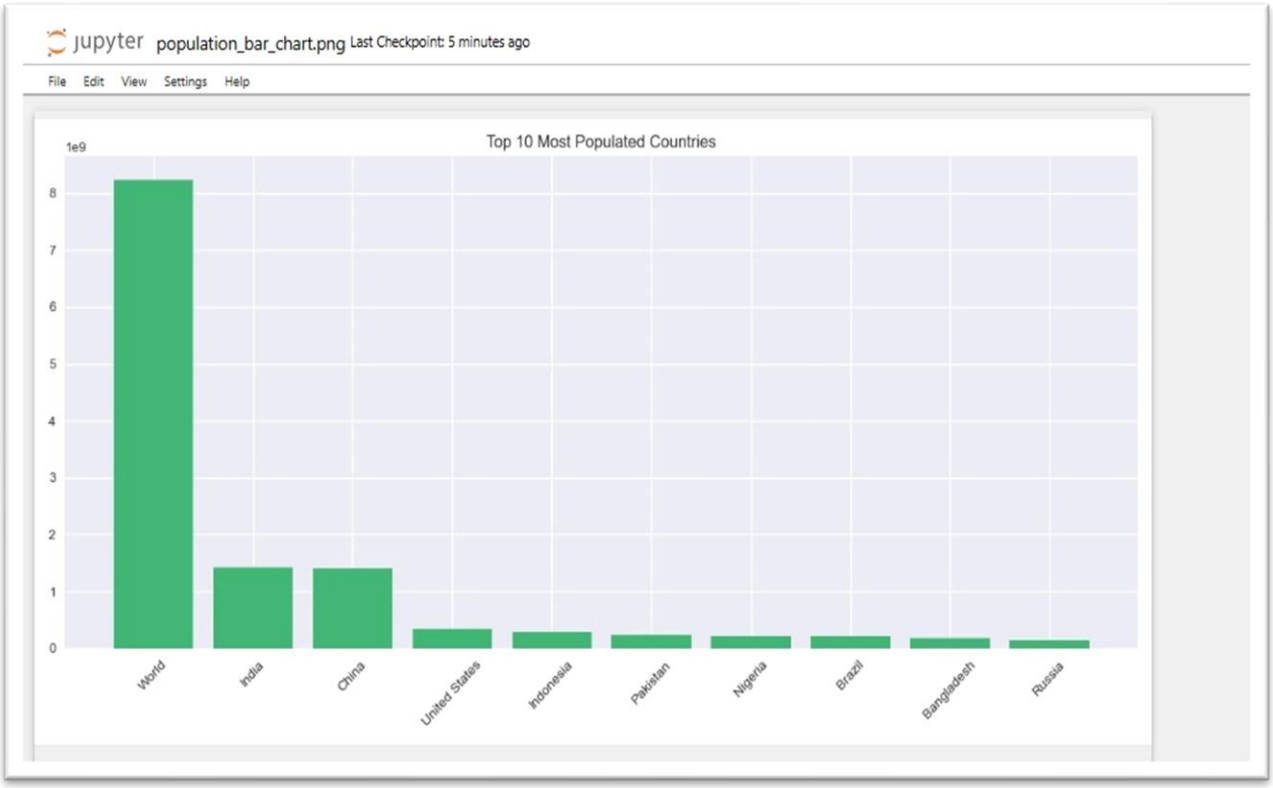


Figure 10.2: Saved bar chart of the Top 10 most populated countries

Section 11: Conclusion

This project successfully delivered a complete end-to-end data engineering workflow, integrating multiple tools to transform raw shipment data into meaningful analytical insights. The work covered every stage of the data lifecycle, including extraction, cleaning, transformation, modeling, automation, and visualization.

The key accomplishments of the project include:

- **Excel preprocessing** to clean and organize the initial dataset
- **Pivot tables** for multi-level summarization and operational insights
- **Computed fields** to standardize inconsistent values
- **Interactive Power BI dashboards** for real-time monitoring
- **MySQL relational database** designed with proper keys and constraints
- **Docker-based automation** for running the ETL pipeline in a portable environment
- **Selenium web scraping** to acquire additional structured datasets
- **Python data manipulation** for sorting, filtering, validation, and summary statistics
- **Final visualizations** to communicate key insights effectively

Overall, this project demonstrates strong proficiency in **ETL development, database management, automation, analytics, and data visualization**. It reflects the ability to build and execute a complete data engineering workflow from raw data ingestion to final reporting.

Thank you for giving me the opportunity to introduce myself.

If I describe myself in one line—**I'm someone who sees data not just as numbers, but as stories waiting to be decoded.**

I'm **Veman Srinivas Chippa**, and my journey into the data domain started much earlier than I realized. During my undergraduate projects and internships, I found myself naturally drawn to solving problems by breaking them down, identifying patterns, and validating every assumption with data.

Whether I was working as a **Database Intern**, optimizing queries, or as a **Cloud Engineering Intern**, monitoring systems and logs, I was always the person asking:

"Why is this happening? What does the data say?"
That curiosity slowly shaped my career path.

After graduation, I worked as a **Technical Support Engineer**, where I dealt with real users, real issues, and real production data. That experience taught me something extremely important:

Data creates value only when it solves real problems for real people.

This mindset encouraged me to pursue my **Master's in Data Science** and focus on building solutions where data drives decisions, efficiency, and innovation.

Today, I'm looking for a role where I can combine:

- My foundation in databases
- My hands-on cloud engineering experience
- My technical troubleshooting background
- My growing expertise in analytics and data science

My goal is simple:
Use data to simplify complexity, uncover insights, and build solutions that create real impact.

Veman Chippa

🔗 PROFESSIONAL LINKS

Recent Project

📘 **BookSense AI** – A smart book recommendation system built using NLP, vector embeddings, and ML inference

🔗 **Live Demo:** https://iveman-booksenseai-berf.onrender.com/

Portfolio & GitHub

🌐 **Portfolio:** https://iveman.vercel.app/

💻 **GitHub:** https://github.com/iveman99