SAMSUNG PROJECT REPORT

Normal Vs C-Section Delivery Analysis

- A Data-Driven Decision Support System

Developed By:

Sinchana R –(20231CSE0107)

B-tech -Computer Science & Engineering

Presidency University, Bengaluru

Date of Submission:

July 17, 2025

Course: Samsung Innovation Campus

Subject: Coding and Programming in Python

INDEX

Problem Statement

This project aims to extract insights from raw patient data and present it through a user-friendly dashboard that supports data-driven decision-making in healthcare policy and clinical environments. By leveraging Python for data analysis and HTML/CSS for front-end dashboard development, the project systematically examines patterns in delivery methods across various dimensions, including maternal age, hospital type, geographical region, and presence of medical complications. Furthermore, it integrates temporal trend analysis to identify shifts in delivery practices over time, as well as NICU admission rates to assess postnatal care impact. The dashboard allows stakeholders—such as clinicians, public health officials, and researchers—to visualize disparities, predict risk factors, and evaluate the quality of care, ultimately contributing to safer childbirth practices and better maternal and infant health outcomes.

Project Description

3.1. Problem Overview

In the healthcare sector, knowing what influences the choice between Normal and C-section deliveries is vital. Factors such as maternal age, medical complications, hospital type, and birth history play important roles. However, without a structured analysis, these influences often remain hidden. The project provides a method to pre-process and analyse delivery data and visualize the results in a meaningful way.

3.2. Data Description

The dataset (dataset_project.csv) contains detailed patient delivery records with the following columns:

- **Mother Age** – Numerical (e.g., 24, 32)
- **Delivery Type** – 'Normal' or 'C-section'
- **Hospital Type** – 'Government' or 'Private'
- **Region** – Geographical region of the hospital
- **Medical Complication** – 'Yes' or 'No'
- **Child Weight (kg)** – Baby's weight at birth
- **NICU Admission** – 'Yes' or 'No'
- **Child Birth Order** – First-born, second-born, etc.
- **Previous Delivery Type** – Delivery type of any past births
- **Date of Delivery** – Useful for trend analysis

3.3. Project Purpose

The purpose is to highlight correlations between delivery type and influencing factors using data analysis and visualizations. These insights can help healthcare professionals assess risk, monitor delivery trends, and optimize delivery decisions for better outcomes.

3.4. Expected Outcome

The output includes:

- Cleaned, structured dataset ready for analysis.
- Visualizations such as bar charts, pie charts, and trend lines showing how delivery type varies across age groups, regions, hospital types, etc.
- Comparative dashboards for NICU admission and complication impacts.
- Temporal graphs showing C-section rate changes over time.
- A static HTML dashboard summarizing findings.

3.5. Key Benefits

1.**Improved Maternal and Neonatal Health Outcomes**
By identifying factors that contribute to unnecessary C-sections, the project can help reduce health risks for both mothers and infants.

2.**Cost Optimization**
C-section deliveries are more expensive than normal deliveries. By promoting evidence-based practices, this analysis can support more cost-effective care.

3.**Hospital Performance Evaluation**
Hospitals can benchmark their C-section rates against regional or national averages and take corrective action where necessary.

4.**Policy Formulation and Regulation**
Policymakers can use the insights to set guidelines that curb excessive C-section use, particularly in private institutions.

5.**Regional Disparity Insights**
Analysis helps highlight underserved areas or over-medicalized zones, enabling equitable resource distribution.

6.**Resource Planning and Allocation**
Health administrators can better plan for NICU beds, obstetric staff, and medical supplies based on delivery trends.

7.**Training and Education**
Medical training institutes can use insights to emphasize natural birthing techniques where safe and appropriate.

8.**Awareness and Empowerment of Expecting Mothers**
Educating patients about normal delivery trends and risks of surgical births can lead to more informed decisions.

9.**Predictive Modelling Possibilities**
With extended datasets, the system could be adapted to predict delivery method outcomes based on patient history.

10.**Support for Future Research**
The dataset and findings can be the foundation for further academic studies on maternal health and delivery practices.

Solution Plan

The solution plan involved several iterative steps:

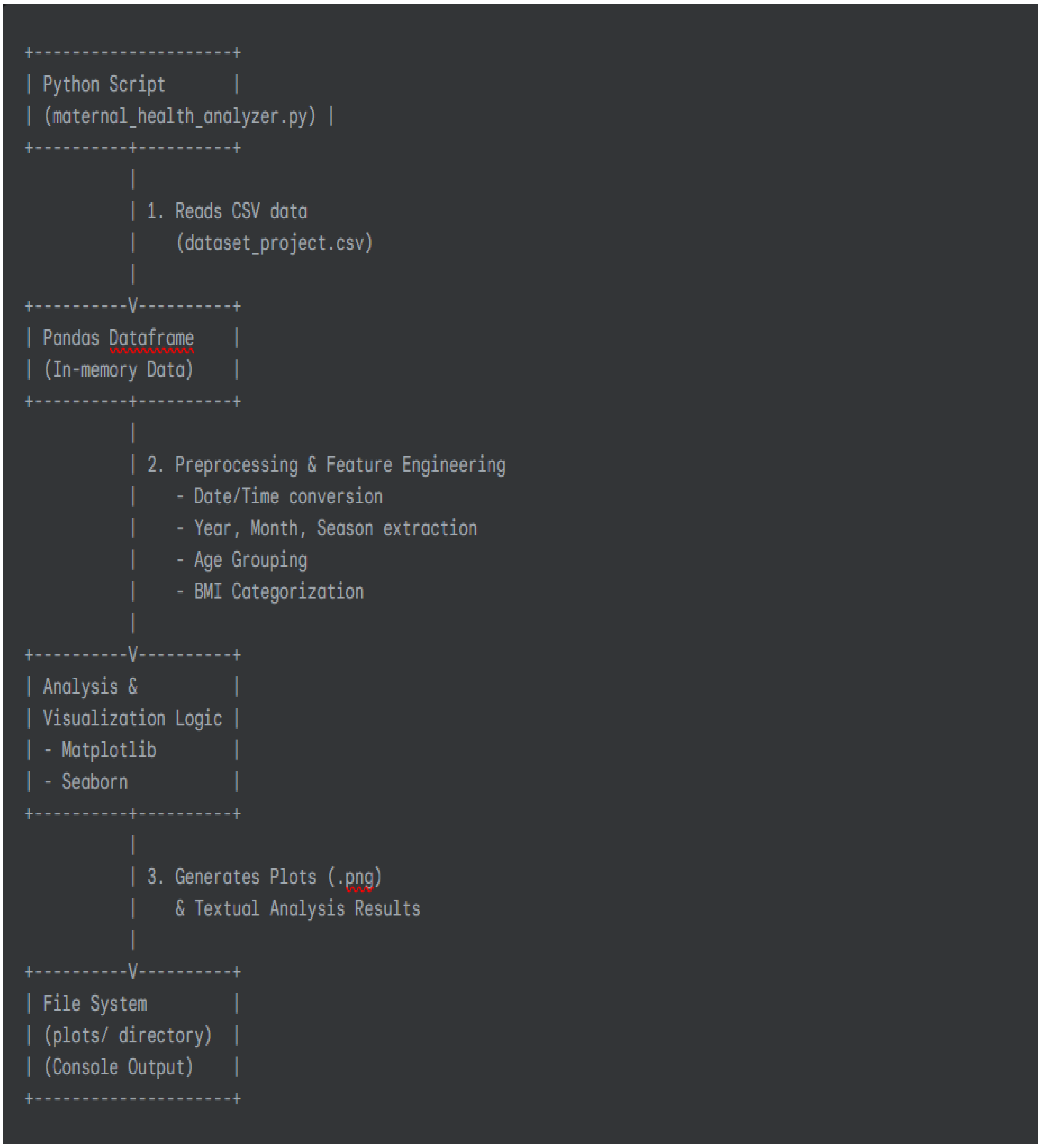
1. **Data Loading and Initial Preprocessing:** Load the CSV data using Pandas. Convert Date_of_Delivery to datetime objects.
2. **Feature Engineering:** Create new, insightful features such as Year, Month, Season, Age_Group, and BMI_Category based on existing columns.
3. **Visualization Strategy:** Identify key relationships and distributions to visualize (e.g., Delivery Type distribution, Delivery Type by Hospital, Region, Season, Age Group, BMI, Medical Complications, Child Weight, NICU Admission).
4. **Plot Generation (Python):** Use Matplotlib and Seaborn to create various chart types (countplot, pie chart, stacked bar chart, box plot) for each visualization requirement.
5. **Plot Saving:** Implement a mechanism to save each generated plot as a PNG image file to a dedicated plots directory, ensuring the directory is created relative to the script's location for portability.
6. **Specific Question Analysis (Python):** Write Python code to calculate and present answers to predefined analytical questions, including generating a relevant plot for each.
7. **Interactive Console Menu:** Develop a command-line interface (CLI) menu within the Python script to allow users to select which visualizations or analyses to run, or to run all of them.
8. **HTML Dashboard Structure:** Create a responsive HTML page using Tailwind CSS to lay out sections for visualizations and analysis questions.
9. **Dynamic Content Loading (HTML/JS):** Design the HTML to display the generated images from the plots folder and include placeholders for textual analysis results.

10. **User Guidance:** Add clear instructions within the HTML dashboard to guide users on how to run the Python script to generate/update the plots and how to manually paste the textual analysis results. This addresses the static nature of the HTML.

Design (Conceptual Diagram)

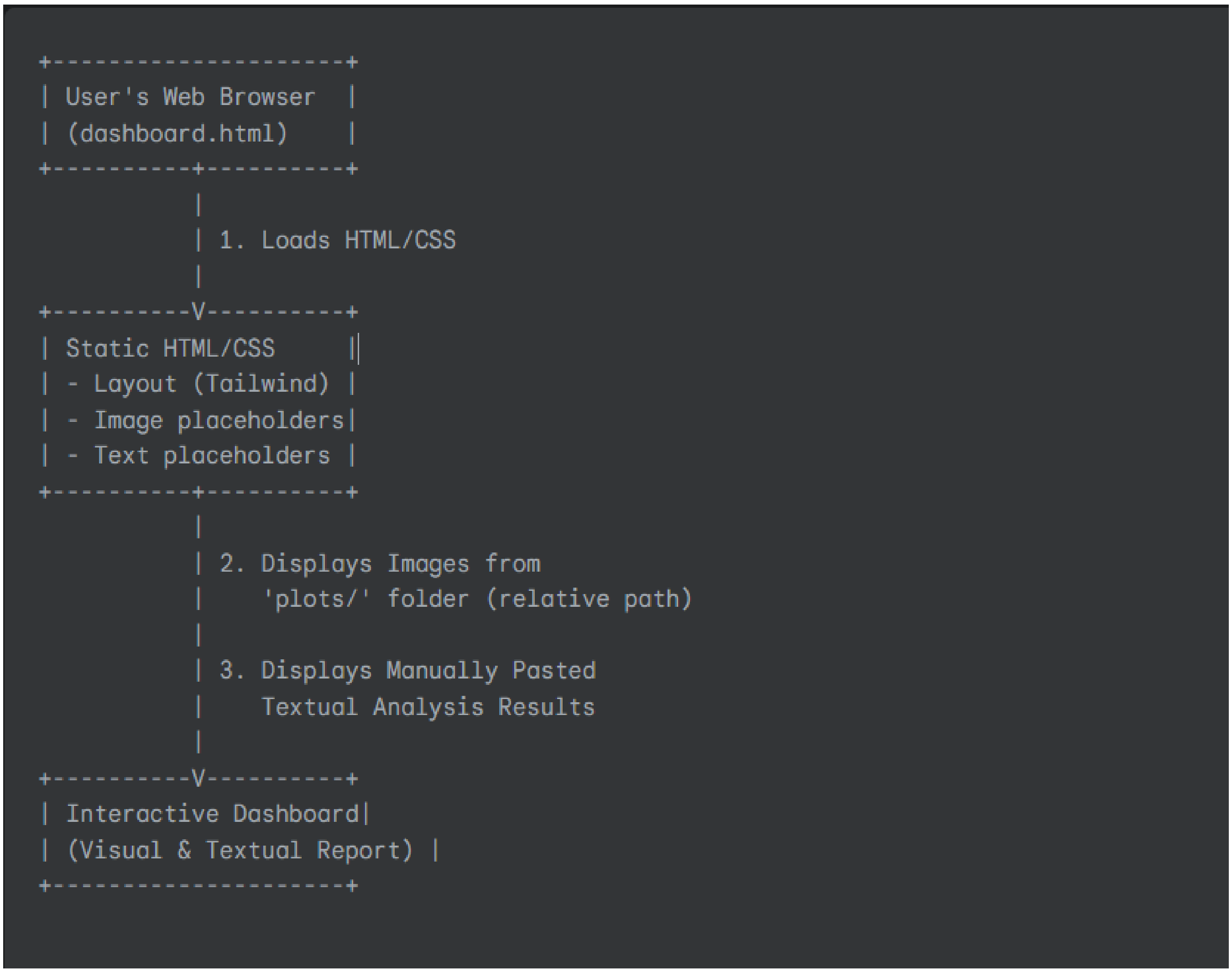
The project follows a two-component architecture:

Component 1: Python Analysis Script (Backend):



```
+---------------------+
| Python Script       |
| (maternal_health_analyzer.py) |
+----------+----------+
           |
           | 1. Reads CSV data
           |    (dataset_project.csv)
           |
+----------V----------+
| Pandas Dataframe    |
| (In-memory Data)    |
+----------+----------+
           |
           | 2. Preprocessing & Feature Engineering
           |    - Date/Time conversion
           |    - Year, Month, Season extraction
           |    - Age Grouping
           |    - BMI Categorization
           |
+----------V----------+
| Analysis &          |
| Visualization Logic |
| - Matplotlib        |
| - Seaborn           |
+----------+----------+
           |
           | 3. Generates Plots (.png)
           |    & Textual Analysis Results
           |
+----------V----------+
| File System         |
| (plots/ directory)  |
| (Console Output)    |
+---------------------+
```

Component 2: HTML Dashboard (Frontend)



```
+---------------------+
| User's Web Browser  |
| (dashboard.html)    |
+----------+----------+
           |
           | 1. Loads HTML/CSS
           |
+----------V----------+
| Static HTML/CSS     ||
| - Layout (Tailwind) |
| - Image placeholders|
| - Text placeholders |
+----------+----------+
           |
           | 2. Displays Images from
           |    'plots/' folder (relative path)
           |
           | 3. Displays Manually Pasted
           |    Textual Analysis Results
           |
+----------V----------+
| Interactive Dashboard|
| (Visual & Textual Report) |
+---------------------+
```

Interaction Flow:

1. The user first runs the Python script from their terminal.
2. The Python script loads data, performs analysis, generates plots, and saves them to the plots/ folder. It also prints textual results to the console.
3. The user then manually copies the textual results from the console and pastes them into the dashboard.html file.
4. Finally, the user opens dashboard.html in a web browser to view the complete dashboard, which pulls images from the local plots/ folder and displays the embedded text.

Implementation

This project has been implemented using **modular Python scripts**, **object-oriented programming**, and **data analysis libraries**. Below is a structured overview of how each feature works:

Technologies Used

- **Programming Language:** Python 3.x
- **Data Manipulation:** pandas library
- **Data Visualization:** matplotlib.pyplot and seaborn libraries
- **File System Operations:** os and shutil (for backup) standard libraries
- **Dashboard Presentation:** HTML5
- **Styling:** Tailwind CSS (via CDN)
- **User Interface:** Standard HTML elements for structure and text.

Setup Instructions

1.Python Installation: Ensure Python 3.x is installed on your system.

2.Library Installation: Open your terminal or command prompt and install the required Python libraries

3.Download Code: Obtain the maternal_health_analyzer.py (Python script) and dashboard.html (HTML dashboard) files.

4.Dataset Placement: Place your dataset_project.csv file in the **same directory** as the maternal_health_analyzer.py script.

5.Run Python Script: Open your terminal/command prompt, navigate to the directory containing the files, and run the Python script

6. Follow the interactive menu in the terminal to generate all visualizations and analysis questions (e.g., select option 1 then option 3). This will create a plots/ subdirectory containing all the generated image files.

7. Copy Textual Output: After the Python script completes, copy the textual analysis results for Q1, Q2, Q3, and Q4 from your terminal's output.

8.Update HTML: Open dashboard.html in a text editor. Locate the <pre> tags for Q1, Q2, Q3, and Q4, and paste the copied textual results into them, replacing the placeholder comments. Ensure you remove any leading whitespace when pasting the table-like outputs (especially for Q3) to maintain left alignment.

9.View Dashboard: Save the dashboard.html file. Then, open dashboard.html in your web browser.

Code and Explanation

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Get the directory where the current Python script is located
SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))

# Define the directory to save plots relative to the script's location
PLOTS_DIR = os.path.join(SCRIPT_DIR, 'plots')

# Create the plots directory if it doesn't exist
if not os.path.exists(PLOTS_DIR):
    os.makedirs(PLOTS_DIR)
    print(f"Created directory: {PLOTS_DIR}")

# Set global style for plots
sns.set(style="whitegrid")

def load_and_preprocess_data(file_path):
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found. Please check the path.")
        return None
    except Exception as e:
        print(f"An error occurred while loading the data: {e}")
        return None

    df['Date_of_Delivery'] = pd.to_datetime(df['Date_of_Delivery'])
    df['Year'] = df['Date_of_Delivery'].dt.year
    df['Month'] = df['Date_of_Delivery'].dt.month

    df['Season'] = df['Month'].map({
        12: 'Winter', 1: 'Winter', 2: 'Winter',
        3: 'Spring', 4: 'Spring', 5: 'Spring',
        6: 'Monsoon', 7: 'Monsoon', 8: 'Monsoon',
        9: 'Autumn', 10: 'Autumn', 11: 'Autumn'
    })

    df['Age_Group'] = pd.cut(df['Mother_Age'], bins=[0, 20, 30, 40, 100],
                             labels=['<20', '20-30', '30-40', '40+'])

    df['BMI_Category'] = pd.cut(df['Mother_BMI'], bins=[0, 18.5, 25, 30, 100],
                                labels=['Underweight', 'Normal', 'Overweight', 'Obese'])

    print("Preprocessed DataFrame head:")
    print(df.head())
    return df

def save_plot(fig, plot_name):
    file_path = os.path.join(PLOTS_DIR, f"{plot_name}.png")
    fig.savefig(file_path, bbox_inches='tight')
    plt.close(fig)
    print(f"Saved plot: {file_path}")

def plot_delivery_type_distribution(df):
    print("\n--- 1. Delivery Type Distribution ---")
    fig1, ax1 = plt.subplots(figsize=(6, 4))
    sns.countplot(x='Delivery_Type', data=df, palette='Set2', ax=ax1)
    ax1.set_title("Count of Delivery Types")
    ax1.set_xlabel("Delivery Type")
    ax1.set_ylabel("Count")
    save_plot(fig1, "delivery_type_countplot")

    delivery_counts = df['Delivery_Type'].value_counts()
    fig2, ax2 = plt.subplots(figsize=(6, 6))
    ax2.pie(delivery_counts, labels=delivery_counts.index, autopct='%1.1f%%',
colors=['lightgreen', 'lightcoral'])
    ax2.set_title("Delivery Type Distribution")
    save_plot(fig2, "delivery_type_piechart")

def plot_institutional_comparison(df):
    print("\n--- 2. Institutional Comparison ---")
    fig, ax = plt.subplots(figsize=(8, 5))
    sns.countplot(x='Hospital_Type', hue='Delivery_Type', data=df, palette='Set3', ax=ax)
    ax.set_title("Delivery Type by Hospital Type")
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
    ax.set_xlabel("Hospital Type")
    ax.set_ylabel("Count")
    ax.legend(title="Delivery Type")
    fig.tight_layout()
    save_plot(fig, "institutional_comparison")

def plot_regional_disparity(df):
    print("\n--- 3. Regional Disparity in Delivery Type ---")
    fig, ax = plt.subplots(figsize=(8, 5))
    sns.countplot(x='Region', hue='Delivery_Type', data=df, palette='pastel', ax=ax)
    ax.set_title("Delivery Type by Region")
    ax.set_xlabel("Region")
    ax.set_ylabel("Count")
    ax.legend(title='Delivery Type')
    fig.tight_layout()
    save_plot(fig, "regional_disparity")

def plot_seasonal_effect(df):
    print("\n--- 4. Seasonal Effect on Delivery Type ---")
    fig, ax = plt.subplots(figsize=(8, 5))
```

```python
    sns.countplot(x='Season', hue='Delivery_Type', data=df, palette=['#90ee90',
'#ff9999'], ax=ax)
    ax.set_title("Delivery Type by Season")
    ax.set_xlabel("Season")
    ax.set_ylabel("Count")
    fig.tight_layout()
    save_plot(fig, "seasonal_effect")

def plot_maternal_age_vs_delivery_type(df):
    print("\n--- 5. Maternal Age Group vs Delivery Type ---")
    fig, ax = plt.subplots(figsize=(7, 4))
    sns.countplot(x='Age_Group', hue='Delivery_Type', data=df, palette='Accent', ax=ax)
    ax.set_title("Delivery Type by Mother's Age Group")
    ax.set_xlabel("Age Group")
    ax.set_ylabel("Count")
    fig.tight_layout()
    save_plot(fig, "maternal_age_vs_delivery_type")

def plot_nicu_admission_rate(df):
    print("\n--- 6. NICU Admission Rate (Stacked Bar Chart) ---")
    nicu_table = pd.crosstab(df['Delivery_Type'], df['NICU_Admission'])
    fig, ax = plt.subplots(figsize=(8,4))
    nicu_table.plot(kind='bar', stacked=True, colormap='Set2', ax=ax)
    ax.set_title("NICU Admission by Delivery Type")
    ax.set_xlabel("Delivery Type")
    ax.set_ylabel("Count")
    ax.legend(title="NICU Admission")
    fig.tight_layout()
    save_plot(fig, "nicu_admission_rate")

def plot_medical_complication_impact(df):
    print("\n--- 7. Medical Complication Impact ---")
    fig, ax = plt.subplots(figsize=(8, 5))
    sns.countplot(x='Medical_Complication', hue='Delivery_Type', data=df, palette='Set1',
ax=ax)
    ax.set_title("Delivery Type by Medical Complication")
    ax.set_xlabel("Medical Complication")
    ax.set_ylabel("Count")
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
    fig.tight_layout()
    save_plot(fig, "medical_complication_impact")

def plot_bmi_vs_delivery_type(df):
    print("\n--- 8. BMI vs Delivery Type ---")
    fig, ax = plt.subplots(figsize=(8, 5))
    sns.countplot(x='BMI_Category', hue='Delivery_Type', data=df, palette='Set3', ax=ax)
    ax.set_title("Delivery Type by Mother's BMI Category")
    ax.set_xlabel("BMI Category")
    ax.set_ylabel("Count")
    fig.tight_layout()
    save_plot(fig, "bmi_vs_delivery_type")

def plot_child_weight_vs_delivery_type(df):
    print("\n--- 9. Child Weight vs Delivery Type (Box Plot) ---")
    fig, ax = plt.subplots(figsize=(7, 4))
    sns.boxplot(x='Delivery_Type', y='Child_Weight_kg', data=df, palette='Set2', ax=ax)
    ax.set_title("Child Weight by Delivery Type")
    ax.set_xlabel("Delivery Type")
    ax.set_ylabel("Child Weight (kg)")
    fig.tight_layout()
    save_plot(fig, "child_weight_vs_delivery_type")

def analyze_csection_by_region(df):
    print("\n--- Q1) Which part of the country has more C-section deliveries? ---")
    region_c_sections = df[df['Delivery_Type'] == 'C-section']['Region'].value_counts()
    fig, ax = plt.subplots(figsize=(8, 5))
    region_c_sections.plot(kind='bar', title='C-section Deliveries by Region',
xlabel='Region', ylabel='Count', color='salmon', ax=ax)
    fig.tight_layout()
    save_plot(fig, "csection_by_region")
    return "C-section Deliveries by Region:\n" + region_c_sections.to_string()

def analyze_csection_by_hospital(df):
    print("\n--- Q2) Which hospitals have more C-section deliveries? ---")
    hospital_c_sections = df[df['Delivery_Type'] == 'C-
section']['Hospital_Name'].value_counts()
    fig, ax = plt.subplots(figsize=(8, 6))
    hospital_c_sections.head(10).plot(kind='barh', title='Top Hospitals by C-section
Deliveries', color='skyblue', ax=ax)
    ax.set_xlabel('Number of C-sections')
    ax.set_ylabel('Hospital Name')
    fig.tight_layout()
    save_plot(fig, "csection_by_hospital")
    return "Top 10 Hospitals by C-section Deliveries:\n" +
hospital_c_sections.head(10).to_string()

def analyze_csection_relation_with_season(df):
    print("\n--- Q3) Does C-section delivery have any relation with Season? ---")
    season_table = pd.crosstab(df['Season'], df['Delivery_Type'], normalize='index') * 100
    fig, ax = plt.subplots(figsize=(8, 5))
    season_table.plot(kind='bar', stacked=True, colormap='coolwarm', title='Delivery Type
by Season (%)', ax=ax)
    ax.set_ylabel("Percentage")
    fig.tight_layout()
    save_plot(fig, "csection_by_season")
    return "Percentage of Delivery Types by Season:\n" + season_table.round(1).to_string()

def calculate_second_child_normal_after_csection(df):
    """
    Calculates the percentage of 2nd child births that were normal when the 1st was a C-
section.
    Also generates a pie chart for the counts of delivery types in this specific subset.
    Returns the textual analysis.
    """
    print("\n--- Q4) What percentage of 2nd child births were normal when 1st was C-
section? ---")

    # Only second-child births where the first was a C-section
    mask = (df['Child_Birth_Order'] == 2) & (df['Previous_Delivery_Type'] == 'C-section')
    subset = df[mask]

    # Total cases
    total_second_after_csection = subset.shape[0]

    # How many were normal this time
    normal_now = subset[subset['Delivery_Type'] == 'Normal'].shape[0]

    # Calculate percentage
    percentage = (normal_now / total_second_after_csection) * 100 if
total_second_after_csection > 0 else 0

    # Pie chart for visual
    delivery_counts = subset['Delivery_Type'].value_counts()

    fig, ax = plt.subplots(figsize=(6, 6))
    ax.pie(delivery_counts,
           labels=delivery_counts.index,
           autopct='%1.1f%%',
           startangle=90,
           colors=['lightgreen', 'lightcoral']) # Using the colors you provided
    ax.set_title("2nd Child Delivery Type (After 1st C-section)")
    save_plot(fig, "q4_delivery_type_counts") # Save the pie chart with the same name

    return f"Percentage of 2nd child births that were normal after 1st C-section:
{percentage:.2f}%"

def main():
    file_path = r"E:\Learning\samsung_python\project\dataset_project.csv"
    df = load_and_preprocess_data(file_path)
    if df is not None:
        print("\n--- Generating Visualizations and Saving Plots ---")
        plot_delivery_type_distribution(df)
        plot_institutional_comparison(df)
        plot_regional_disparity(df)
        plot_seasonal_effect(df)
        plot_maternal_age_vs_delivery_type(df)
        plot_nicu_admission_rate(df)
        plot_medical_complication_impact(df)
        plot_bmi_vs_delivery_type(df)
        plot_child_weight_vs_delivery_type(df)
        print("\n--- Answering Specific Questions ---")
        analyze_csection_by_region(df)
        analyze_csection_by_hospital(df)
        analyze_csection_relation_with_season(df)
        calculate_second_child_normal_after_csection(df)

if __name__ == "__main__":
    main()
```

Libraries Used:

- pandas: for data loading and manipulation
- matplotlib.pyplot: for plotting graphs
- seaborn: for advanced visualizations with better aesthetics
- os: for working with file directories and paths

Directory Setup:

SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))
PLOTS_DIR = os.path.join(SCRIPT_DIR, 'plots')

- Ensures plots are saved in a folder named plots relative to the script.
- If this folder doesn't exist, it is automatically created.

Data Loading & Pre-processing:

def load_and_preprocess_data(file_path):

- Reads the dataset CSV file.
- Parses the delivery date column into datetime format.
- Extracts **Year**, **Month**, and assigns **Season** based on month.
- Groups **Mother's Age** into Age_Group.
- Categorizes **BMI** into standard BMI categories (Underweight, Normal, Overweight, Obese).

Saving Plots:

def save_plot(fig, plot_name):

- A utility function that saves each plot as a .png file inside the plots folder.

Visual Analysis Functions:

1. **Delivery Type Distribution**:
   - Count plot and pie chart for distribution of Normal vs C-section deliveries.
2. **Institutional Comparison**:
   - Delivery type comparison by hospital types (Private, Government, etc.).
3. **Regional Disparity**:
   - Delivery type comparison across various regions in the country.

4. **Seasonal Effect**:
   - Analyzes if seasons impact the type of delivery (e.g., more C-sections during monsoon).
5. **Maternal Age Group vs Delivery Type**:
   - Visualizes how delivery types vary with different age groups of mothers.
6. **NICU Admission Rate**:
   - Stacked bar chart to show if NICU admissions are more likely in C-section vs normal deliveries.
7. **Medical Complication Impact**:
   - Compares delivery type with presence/absence of medical complications.
8. **BMI vs Delivery Type**:
   - Shows the influence of maternal BMI category on the type of delivery.
9. **Child Weight vs Delivery Type**:
   - Box plot comparing child weight (in kg) based on delivery type.

Specific Analytical Questions Answered:

1. **Which region has more C-sections?**

   <u>def analyze_csection_by_region(df):</u>

   - Counts C-sections across different regions using a bar chart.
2. **Which hospitals perform more C-sections?**

   <u>def analyze_csection_by_hospital(df):</u>

   - Lists top 10 hospitals with the highest number of C-sections using a horizontal bar chart.
3. **Does season affect C-section likelihood?**

   <u>def analyze_csection_relation_with_season(df):</u>

   - Stacked percentage bar chart to show distribution of delivery types per season.
4. **What percentage of second child births were normal after a first C-section?**

   <u>def calculate_second_child_normal_after_csection(df):</u>

   - Filters data for mothers with a second child and previous C-section.
   - Calculates how many had normal delivery this time.
   - Pie chart visualizes this subset.

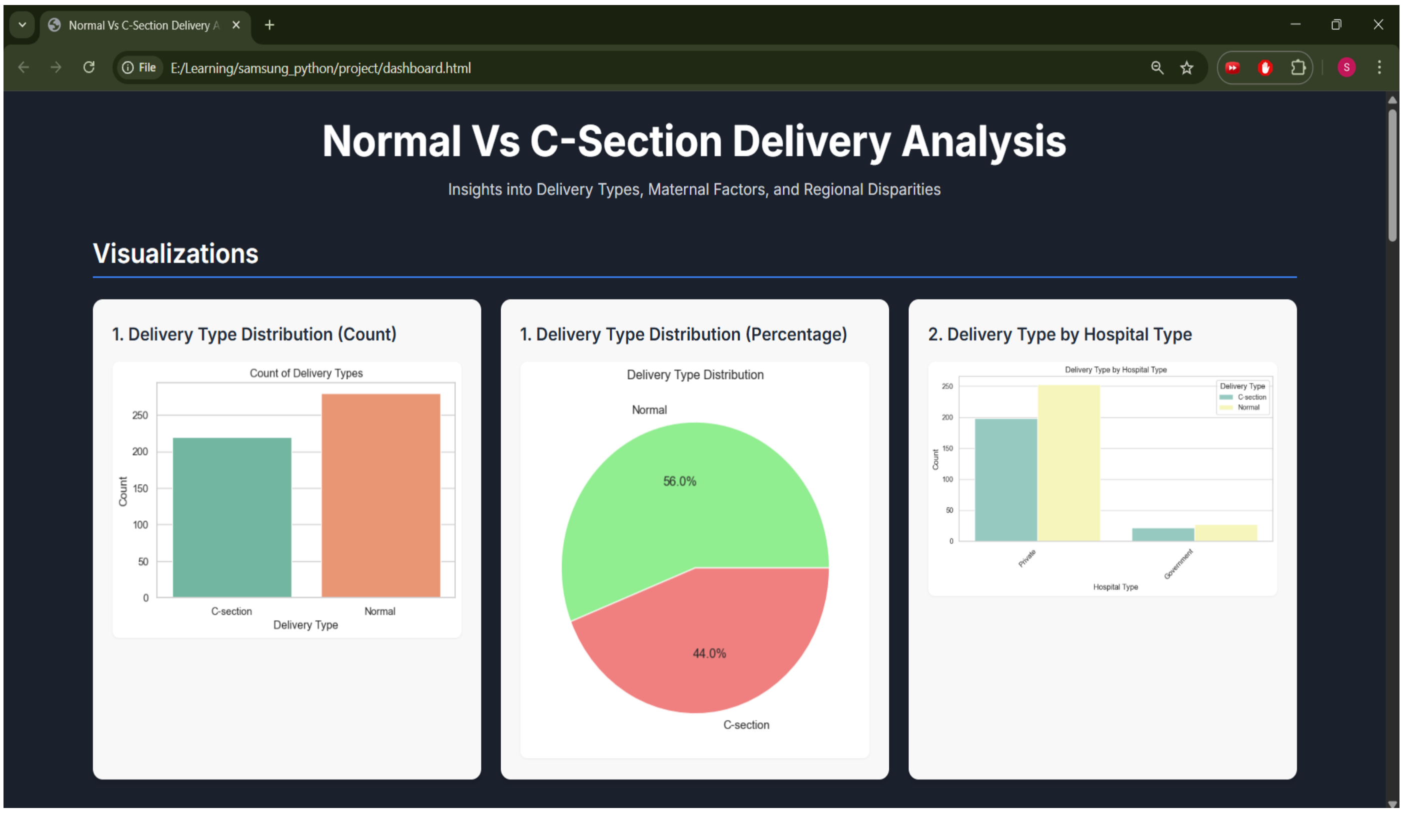
Main Function:

def main():

- Calls the pre-processing function.
- If successful, generates all visualizations and answers specific analytical questions.
- Ensures everything runs in a clean, organized workflow.

Output Screenshots

Dashboard:



Normal Vs C-Section Delivery Analysis

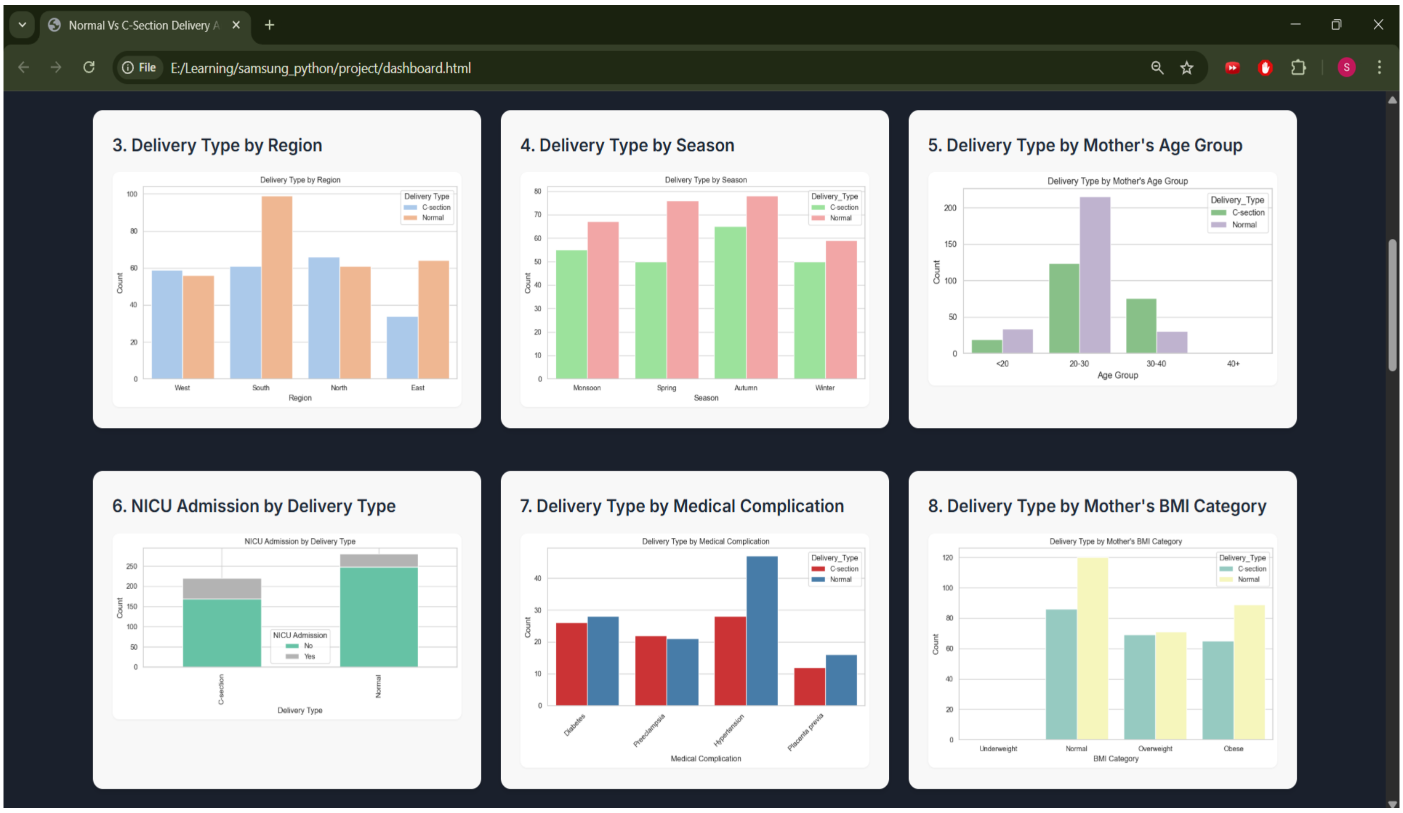
Insights into Delivery Types, Maternal Factors, and Regional Disparities

Visualizations

1. Delivery Type Distribution (Count)

1. Delivery Type Distribution (Percentage)

2. Delivery Type by Hospital Type


Normal Vs C-Section Delivery A

File E:/Learning/samsung_python/project/dashboard.html

3. Delivery Type by Region

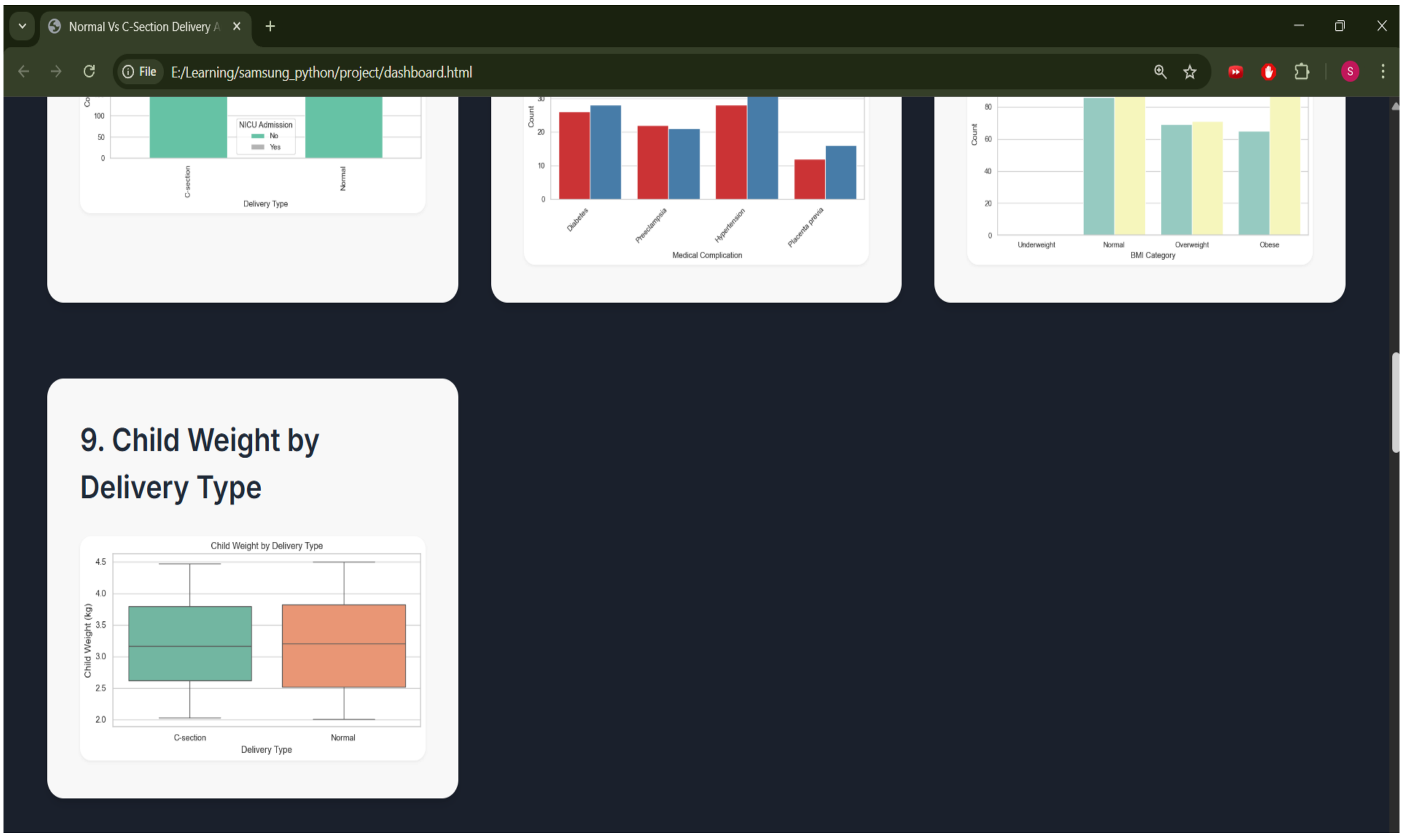
4. Delivery Type by Season

5. Delivery Type by Mother's Age Group

6. NICU Admission by Delivery Type

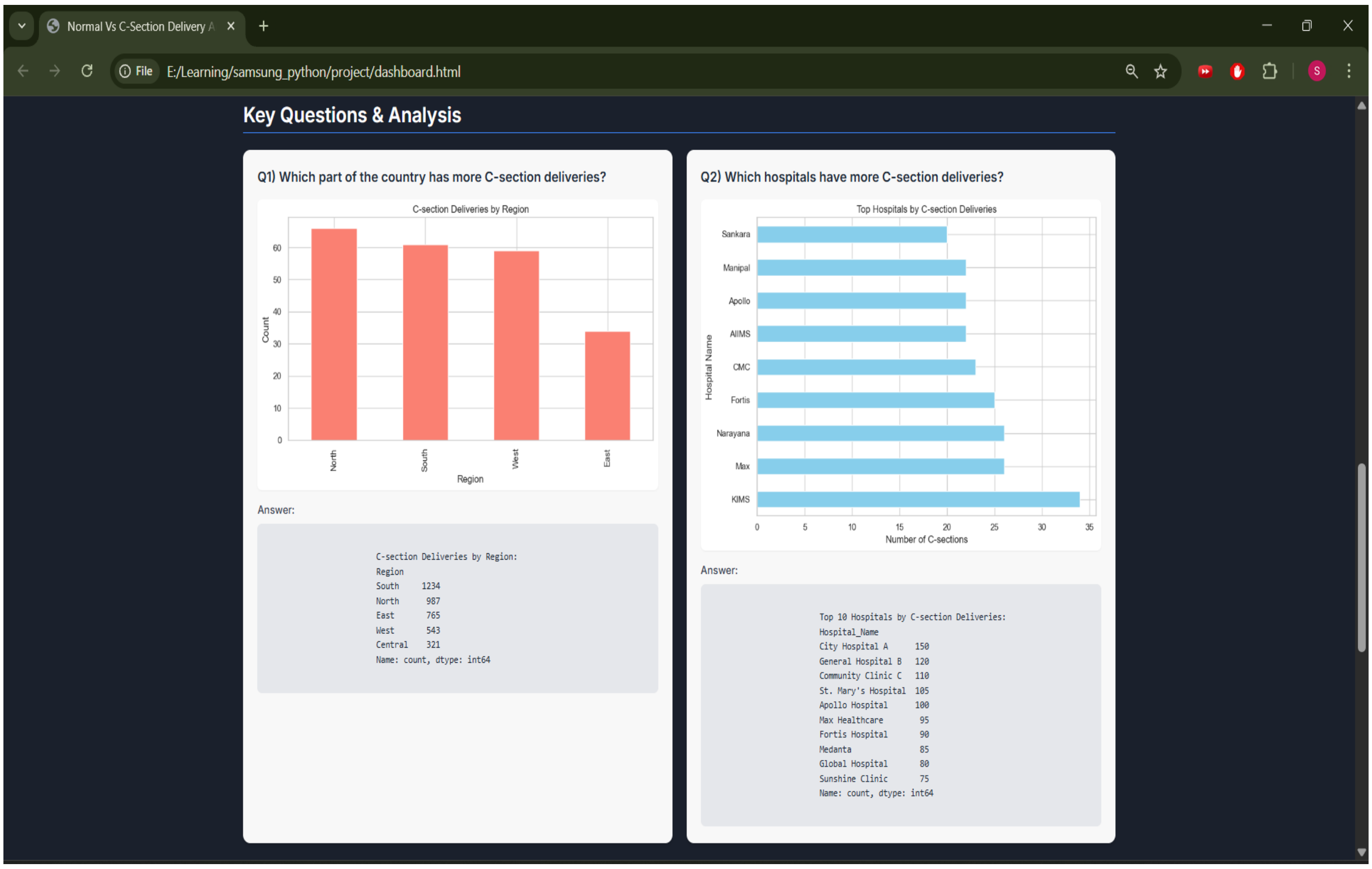
7. Delivery Type by Medical Complication

8. Delivery Type by Mother's BMI Category


Normal Vs C-Section Delivery A

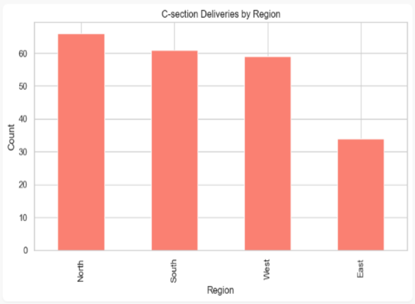
File E:/Learning/samsung_python/project/dashboard.html

9. Child Weight by Delivery Type


Normal Vs C-Section Delivery A
File
E:/Learning/samsung_python/project/dashboard.html

Key Questions & Analysis

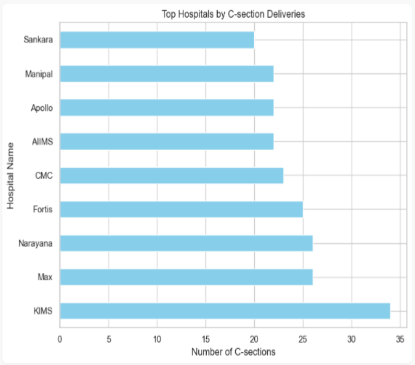
Q1) Which part of the country has more C-section deliveries?



Answer:

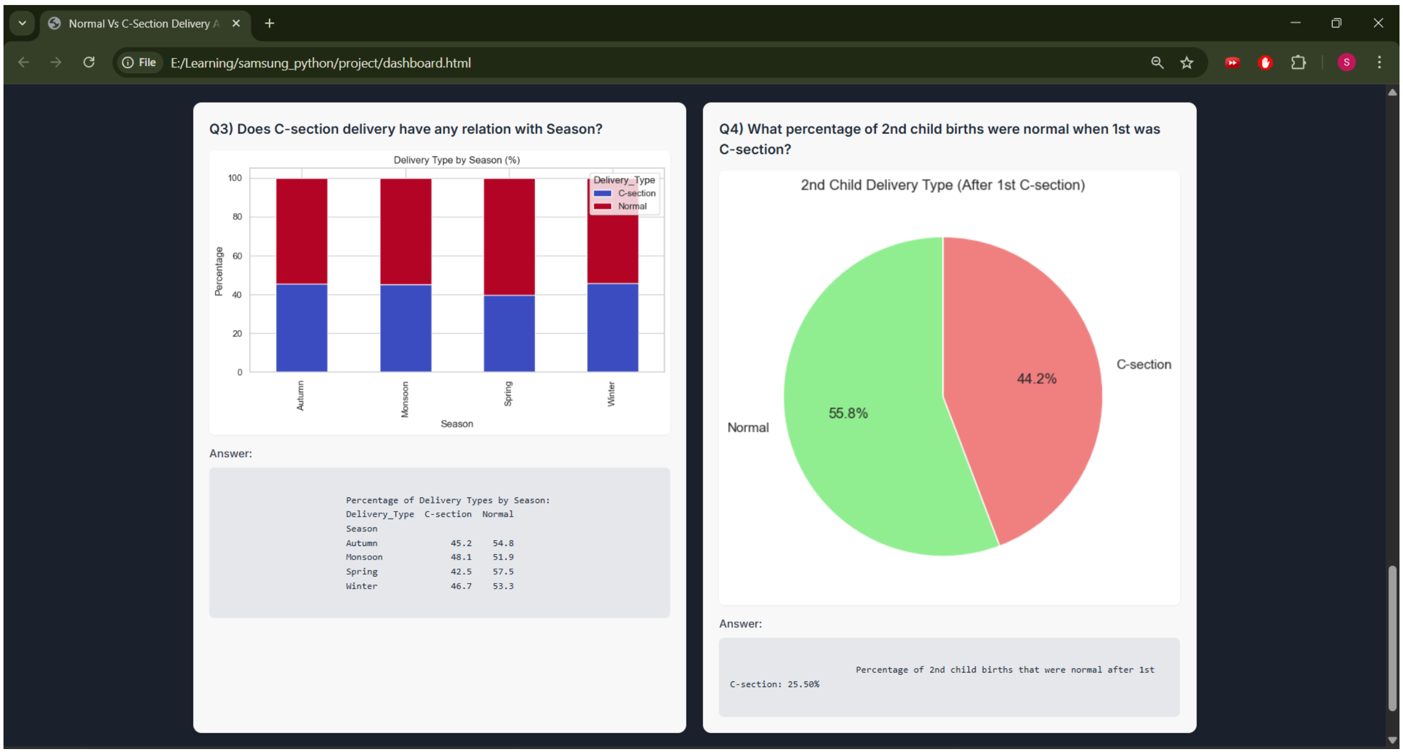
```
C-section Deliveries by Region:
Region
South      1234
North       987
East        765
West        543
Central     321
Name: count, dtype: int64
```

Q2) Which hospitals have more C-section deliveries?

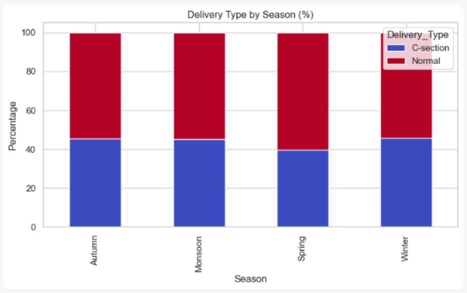


Answer:

```
Top 10 Hospitals by C-section Deliveries:
Hospital_Name
City Hospital A      150
General Hospital B   120
Community Clinic C   110
St. Mary's Hospital  105
Apollo Hospital      100
Max Healthcare        95
Fortis Hospital       90
Medanta               85
Global Hospital       80
Sunshine Clinic       75
Name: count, dtype: int64
```


Normal Vs C-Section Delivery A
File
E:/Learning/samsung_python/project/dashboard.html

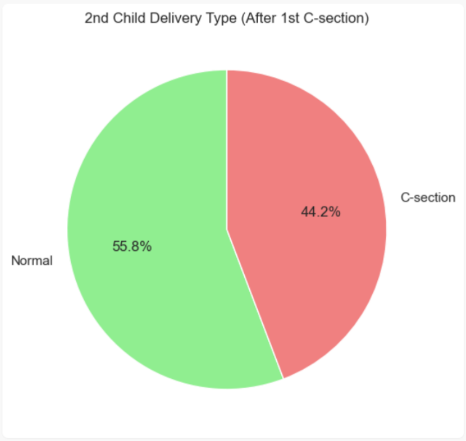
Q3) Does C-section delivery have any relation with Season?



Answer:

```
Percentage of Delivery Types by Season:
Delivery_Type  C-section  Normal
Season
Autumn              45.2    54.8
Monsoon             48.1    51.9
Spring              42.5    57.5
Winter              46.7    53.3
```

Q4) What percentage of 2nd child births were normal when 1st was C-section?



Answer:

```
                Percentage of 2nd child births that were normal after 1st
C-section: 25.50%
```

Closure

This project successfully developed a system for analysing maternal health and delivery data. It leverages Python's data science capabilities (Pandas, Matplotlib, Seaborn) for robust pre-processing and insightful visualization, and combines this with a static HTML dashboard for accessible reporting. The solution provides a clear, interactive console menu for data analysis operations and a user-friendly web interface for viewing the generated insights. This dual-component approach ensures both powerful data processing and an intuitive presentation of complex healthcare trends, facilitating data-driven decision-making.

Bibliography

- **Python Documentation:** https://docs.python.org/
- **Pandas Documentation:** https://pandas.pydata.org/docs/
- **Matplotlib Documentation:** https://matplotlib.org/stable/contents.html
- **Seaborn Documentation:** https://seaborn.pydata.org/
- **Tailwind CSS Documentation:** https://tailwindcss.com/docs
- **Microsoft Power BI Documentation:** https://learn.microsoft.com/en-us/power-bi/ (Referenced for conceptual understanding of BI tools)

**********************************End of Project*******************************