TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

The Olympic Games have been a global phenomenon for over a century, bringing together athletes From around the world to compete in various sports and events. The games have a rich history and have served as a platform for countries to showcase their athletic prowess and cultural heritage.

As part of a college project, a web application has been created that analyzes Olympic data using the Python programming language. The application allows users to interact with the data and gain insights into various aspects of the games, including countries, sports, events, and medal counts. Using various Python libraries for data analysis and visualization, the application provides users with a range of tools for exploring Olympic trends and statistics.

The project demonstrates the power of Python for creating data analysis applications that can be used for various purposes, including sports analysis, business intelligence, and scientific research. The application provides a user-friendly interface that can be easily used by anyone, regardless of their technical expertise. Overall, this project highlights the potential of data analysis and visualization for gaining insights into complex datasets, such as the Olympic data, and showcases the value of using Python as a tool for creating data-driven applications.

1.2 PROJECT DEFINITION

The Olympic data analysis web application project is a software application that allows users to interactively analyze and visualize various aspects of the Olympic Games, such as countries, sports, events, and medal counts. The application is created using Python programming language and uses various Python libraries for data analysis and visualization. The data is obtained from a publicly available Olympic database and is presented in a user-friendly interface that enables users to gain insights into Olympic trends and statistics.

Primary Goal:
The primary goal of the Olympic data analysis web application project is to create a user-friendly and interactive platform that allows users to explore various aspects of the Olympic Games through data analysis and visualization. By using Python and various Python libraries for data analysis and visualization, the project aims to provide users with a range of tools for exploring Olympic trends and statistics, and to enable them to gain insights into the games that are not immediately apparent. The project is intended to be used for various purposes, including sports analysis, business intelligence, and scientific research. Ultimately, the project aims to demonstrate the potential of Python for creating data-driven applications that can be used for a wide range of purposes.

1.3 DECLARATION OF THE PROBLEM

The Olympic Games have a rich history and have served as a platform for countries to showcase their athletic prowess and cultural heritage. However, analyzing and interpreting the vast amount of data associated with the games can be challenging. Traditional methods of data analysis, such as spreadsheets, can be time-consuming and may not provide insights into the data that are not immediately apparent.

To address this problem, the Olympic data analysis web application project was developed. The project aims to create a user-friendly platform that enables users to interactively analyze and visualize various aspects of the Olympic Games. By using Python and various Python libraries for data analysis and visualization, the project aims to provide users with a range of tools for exploring Olympic trends and statistics, and to enable them to gain insights into the games that are not immediately apparent.

Overall, the problem that this project aims to address is the need for a user-friendly and interactive platform that enables users to explore Olympic data in a meaningful way. The project is intended to be used for various purposes, including sports analysis, business intelligence, and scientific research.

1.4 PROJECT PURPOSES

The purpose of the Olympic data analysis web application project is to provide users with a user-friendly and interactive platform that enables them to explore various aspects of the Olympic Games through data analysis and visualization. By using Python, Machine Learning, and Artificial Intelligence methods and approaches, and various Python libraries such as Pandas, Preprocessor, Matplotlib, Seaborn, and Plotly, the project aims to provide users with a range of tools for exploring Olympic trends and statistics, and to enable them to gain insights into the games that are not immediately apparent.

The project's primary goal is to create a web application that can be used for various purposes, including sports analysis, business intelligence, and scientific research. Users can interact with the data and analyze it according to various criteria, such as countries, sports, events, and medal counts. The application is designed to be user-friendly, with a simple interface that allows users to easily explore and analyze the data.

The project's ultimate purpose is to demonstrate the potential of Python, Machine Learning, and Artificial Intelligence for creating data-driven applications that can be used for a wide range of purposes. The project aims to provide users with a platform that enables them to explore and gain insights into complex datasets such as the Olympic data, and to showcase the value of using these technologies for data analysis and visualization.

1.5 ARCHITECTURE & COMPONENTS

The Olympic data analysis web application project is built using the Python programming language and various Python libraries for data analysis, visualization, and web development. The project's architecture consists of three main components

Architecture

1.Data Preprocessing: The first component of the project is data preprocessing, which involves cleaning and preparing the Olympic dataset for analysis. The project uses the Pandas and Preprocessor libraries for these purposes, which enable data cleaning, transformation, and aggregation. The preprocessing step also involves removing any missing or duplicate data, normalizing the data, and creating new features or variables based on the original data.

2. Data Visualization: The second component of the project is data visualization, which involves creating interactive visualizations and plots that enable users to explore and analyze the data. The project uses the Matplotlib, Seaborn, and Plotly libraries for these purposes, which provide a range of visualization tools for creating line charts, bar charts, scatter plots, heatmaps, and more.

3. Web Application: The third component of the project is the web application, which provides a user-friendly interface for users to interact with the data and visualize it. The project uses the Streamlit library for web development, which enables the creation of interactive web applications using Python. The web application component consists of multiple pages that provide various features such as data filtering, sorting, searching, and visualization. The application is deployed on the Streamlit Cloud, a free hosting service for Streamlit applications.

Components:

The components of the Olympic data analysis web application project include:

1. Data Preprocessing:

- Pandas Library
- Preprocessor Library

2. Data Visualization:

- Matplotlib Library
- Seaborn Library
- Plotly Library

Web Application:

- Streamlit Library
- HTML/CSS/JavaScript
- Streamlit Cloud (for deployment)

The project's architecture and components work together to enable users to interactively explore and analyze the vast amount of data associated with the Olympic Games, providing them with a range of tools and insights that may not be immediately apparent from the raw data.

1.6 PROJECT SCOPE

The scope of the Olympic data analysis web application project is to provide users with a user-friendly and interactive web application that allows them to explore and analyze 120 years of Olympic data. The project aims to enable users to gain insights into patterns and

Trends in Olympic history, including athlete performance, medal counts and event participation across different countries and time periods.

The project includes a range of features that enable users to interact with the data, including data filtering, sorting, searching, and visualization. The project also aims to provide users with a range of visualizations that enable them to explore the data in different ways, including line charts, bar charts, scatter plots, heat maps, and more.

The project is intended for use by a wide range of users, including students, researchers, and sports enthusiasts, and is designed to be accessible and informative for users with varying levels of expertise in data analysis and visualization. The project is hosted on the Streamlit Cloud, which provides users with easy access to the application and ensures that the application is always up-to-date with the latest data.

Overall, the scope of the Olympic data analysis web application project is to provide users with a comprehensive and informative tool for exploring and analyzing Olympic data, enabling them to gain insights into one of the world's most iconic sporting events.

1.7 Data Visualization

Data visualization is an important aspect of the Olympic data analysis web application project, as it enables users to explore and understand patterns and trends in Olympic history in an intuitive and interactive way. The project uses several popular Python libraries for data visualization, including Matplotlib, Seaborn, and Plotly, each of which provides different tools for creating visualizations that can be embedded within the web application.

The project includes a wide range of visualizations that enable users to explore different aspects of Olympic history, including athlete performance, medal counts, and event participation across different countries and time periods. These visualizations include:

1. Line charts: These are used to show trends in athlete performance over time, such as the number of medals won by a particular country in different Olympic games.

2. Bar charts: These are used to compare data across different categories, such as the number of gold, silver, and bronze medals won by different countries.

3. Scatter plots: These are used to show the relationship between two variables, such as the number of athletes from a particular country and their performance in different events.

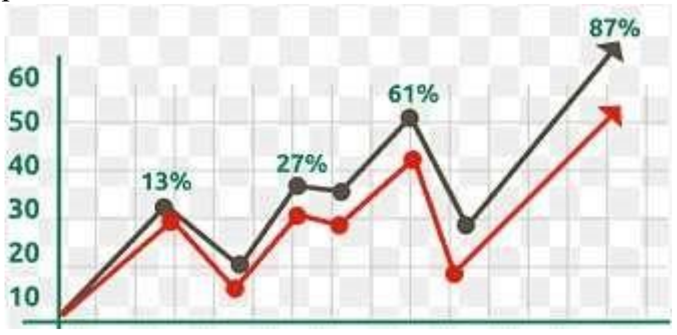
4. Heat maps: These are used to show the distribution of data across different categories, such as the number of medals won by different countries across different events.

The visualizations in the project are highly interactive, enabling users to filter and explore the data in different ways. For example, users can filter the data by year, country, event, or athlete, or zoom in on specific parts of a chart to explore the data in more detail. The project also includes several advanced visualizations, such as choropleth maps and interactive networks, that enable users to explore the data in even more depth.

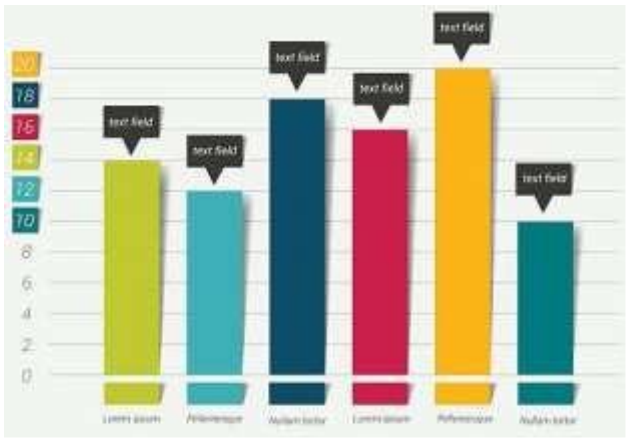
Overall, data visualization plays a crucial role in the Olympic data analysis web application project, enabling users to explore and understand patterns and trends in Olympic history in a highly interactive and engaging way.

1.7.1 Type of Data Visualization

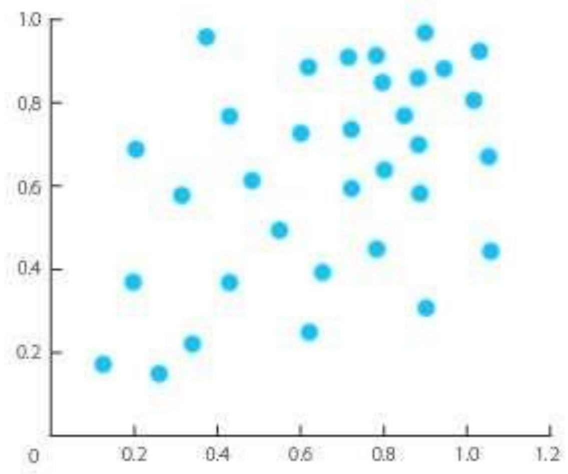
Line Charts: Line charts are used to show trends over time. In the Olympic data analysis web application project, line charts are used to show trends in athlete performance over time, such as the number of medals won by a particular country in different Olympic Games. These charts are useful for identifying patterns and trends in data over time.



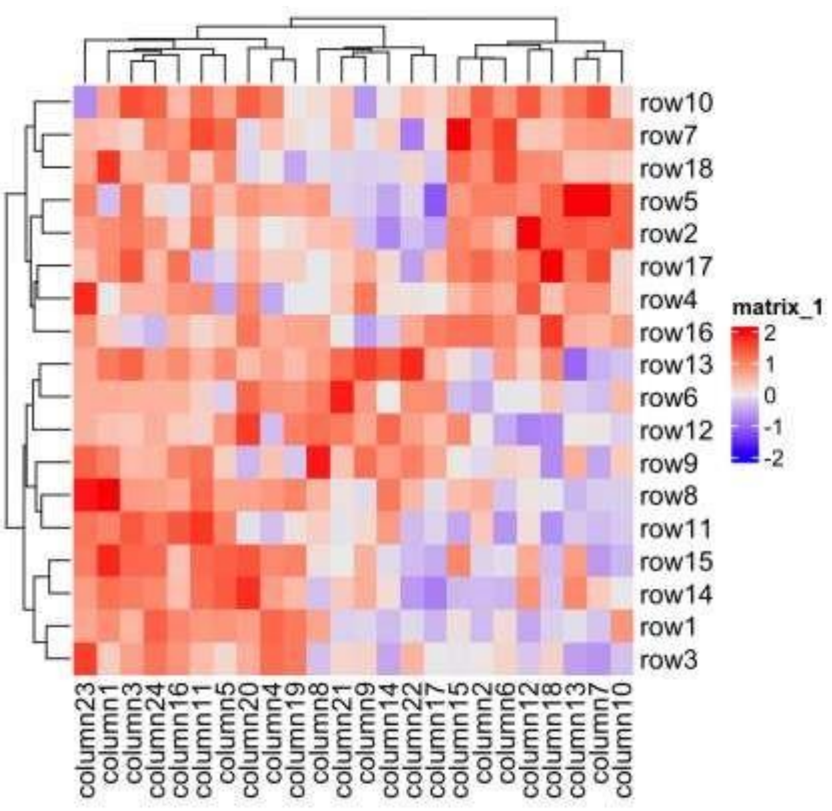
Bar Charts: Bar charts are used to compare data across different categories. In the Olympic data Analysis web application project, bar charts are used to compare the number of medals won by different countries or the number of medals won in different events. These charts are useful for comparing data across different categories.



Scatter Plots: Scatter plots are used to show the relationship between two variables. In the Olympic data analysis web application project, scatter plots are used to show the relationship between the number of athletes from a particular country and their performance in different events. These plots are useful for identifying correlations and relationships between two variables.

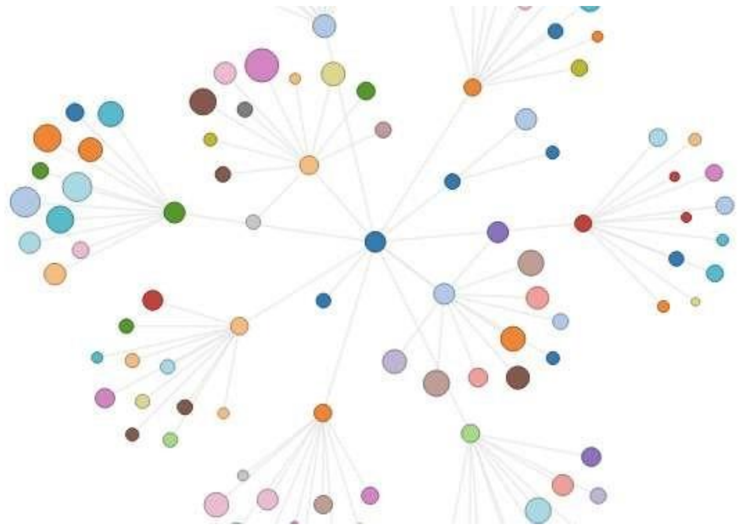


Heatmaps: Heatmaps are used to show the distribution of data across different categories. In the Olympic data analysis web application project, heatmaps are used to show the number of medals won by different countries across different events. These charts are useful for identifying patterns in data across different categories.



Choropleth Maps: Choropleth maps are used to show data across geographical regions. In the Olympic data analysis web application project, choropleth maps are used to show the number of medals won by different countries across the world. These maps are useful for visualizing data across geographical regions.

Interactive Networks: Interactive networks are used to show the relationships between different entities in a network. In the Olympic data analysis web application project, interactive networks are used to show the relationships between different countries based on the number of medals they have won. These networks are useful for visualizing complex relationships between different entities.



Overall, the Olympic data analysis web application project uses a wide range of data visualizations to enable users to explore and understand patterns and trends in Olympic history. Each type of visualization is used to highlight different aspects of the data and provide users with a comprehensive view of the data.

1.7.2 how does this Data Visualization work

In your Olympic data analysis web application project, data visualization is used to represent the large and complex Olympic dataset in a visually appealing and easily understandable way. The process of data visualization involves transforming data into charts, graphs, and other visual elements that can be easily interpreted by the user.

The data visualization in your project is done using various Python libraries such as Matplotlib, Seaborn, and Plotly. These libraries allow you to create various types of visualizations such as line charts, bar charts, scatter plots, heatmaps, and more. The data visualization process starts by pre-processing the raw Olympic dataset using the Pandas library, which involves cleaning and transforming the data into a format that can be used for visualization. The pre-processed data is then passed to the data visualization libraries, which create the desired visualizations based on the data.

The data visualizations in your project are interactive, which means that the user can interact with the visualizations and change the displayed data based on their preferences. For example, the user can filter the data by Olympic year, sport, country, and other criteria to see the data in different ways.

Overall, data visualization is a crucial part of your Olympic data analysis web application project, as it helps users to understand and explore the Olympic dataset more easily and efficiently.

1. Making complex data easy to understand: The Olympic dataset contains a large amount of data that can be difficult to comprehend in its raw form. However, data visualization techniques such as charts, graphs, and maps can help to break down this data into smaller, more digestible chunks that are easier to understand.

2. Providing visual context: Data visualization can provide visual context for the data, allowing users to see trends, patterns, and relationships between different variables. For example, a scatter plot can show the relationship between two variables, such as a country's medal count and its population size.

3. Enabling interactivity: The data visualizations in your project are interactive, meaning that users can manipulate and explore the data in real-time. For example, users can filter the data by year, sport, or country to see how different variables affect the results.

4. Improving data-driven decision making: By presenting data in a visually appealing way, data visualization can help users to make more informed decisions based on the data. For example, a user might use a line chart to compare the medal counts of two different countries over time, helping them to make a more informed decision about which country to support.

data visualization is a powerful tool in your Olympic data analysis web application project that can help users to explore and understand the Olympic dataset more easily and efficiently. By breaking down complex data into smaller, more digestible chunks and providing visual context and interactivity, data visualization can help users to make more informed decisions based on the data.

1.8 SUMMARY

Data visualization in your Olympic data analysis web application project involves using various Python libraries such as Matplotlib, Seaborn, and Plotly to transform complex Olympic dataset into charts, graphs, and other visual elements that can be easily understood by users. The pre-processed data is passed to the data visualization libraries, which create interactive and visually appealing charts, graphs, and maps. Users can manipulate and filter the data in real-time to explore the data more efficiently. Data visualization helps to make complex data easy to understand, provides visual context, enables interactivity, and improves data-driven decision making.

CHAPTER 2: SURVEY OF TECHNOLOGY

2.1 Data Preprocessing and Data Visualization

Data Preprocessing
Data preprocessing is a crucial step in data analysis, which involves cleaning, transforming, and organizing raw data into a format suitable for analysis. In other words, it is the process of converting raw data into a more meaningful and usable format that can be easily analyzed by machines or humans. Data preprocessing typically involves several steps, including removing duplicate or irrelevant data, Filling in missing data, normalizing data, and transforming data to a suitable format. The aim of data preprocessing is to improve the quality and accuracy of data, reduce errors, and enhance the overall efficiency and effectiveness of data analysis. data preprocessing is a vital step in data analysis that helps to ensure that the data is accurate, reliable, and suitable for analysis, which is essential for making informed decisions and drawing meaningful insights from the data.

1. The project reads in the 120 years of Olympic history dataset from Kaggle using the Pandas library.

2. The dataset is then cleaned and preprocessed using various Pandas methods, such as dropping null values, renaming columns, and aggregating data.
3. The preprocessed data is then stored in Pandas data frames for further analysis and visualization.
4. The Scikit-learn library is used for feature scaling, normalization, and machine learning models

Data Visualization

Data visualization is the process of representing data graphically and visually, using charts, graphs, and other visual elements. The primary goal of data visualization is to convey complex data and information in a simple and easy-to-understand format that can be easily interpreted by humans.
Through data visualization, patterns, trends, and relationships in the data can be identified more easily, allowing analysts to draw insights and make informed decisions based on the data. Data visualization is often used to present large and complex datasets in a simplified and meaningful way, making it easier for users to understand and draw conclusions from the data.

Some common types of data visualizations include bar charts, line graphs, scatterplots, heat maps, and pie charts. In recent years, there has been a growing trend towards interactive data visualization, which allows users to explore and interact with the data in real-time, providing a more engaging and personalized experience. data visualization is a crucial tool in data analysis and helps to communicate complex data in a way that is more accessible and easier to understand.

1.The preprocessed data frames are used to create various types of visualizations using Matplotlib, Seaborn, and Plotly libraries.

2.The Streamlit library is used to create an interactive web application for users to explore and interact with the visualizations.

3.The visualizations are organized into different sections, such as medal tally, overall analysis, country analysis, and athlete analysis, which can be accessed by the user through the web application interface

2.2 Core Idea of Olympic Data Analysis

The core idea of Olympic Data Analysis is to analyze and gain insights from the 120 years of Olympic history and data. The aim is to explore the patterns, trends, and relationships in the data, and to uncover interesting insights and findings about the Olympics, its participants, and its impact on society.

The Olympic Data Analysis project involves using various data analysis and machine learning techniques to preprocess, clean, and analyze the data, and to extract meaningful insights from it. The project also involves developing a user interface that allows users to interact with the data and explore it in an engaging and interactive way

1. Data collection: The project uses the Olympic Games dataset available on Kaggle, which contains data on the Olympic Games from 1896 to 2016. The data includes information on athletes, events, medals, countries, and more.

2. Data preprocessing: The raw data from Kaggle requires preprocessing to make it suitable for analysis. The project uses pandas library to preprocess the data. The preprocessing includes handling missing data, converting data types, merging data, and cleaning data.

3. Data visualization: Once the data is preprocessed, the project uses various data visualization libraries such as Matplotlib, Seaborn, and Plotly to create visualizations that provide insights into the data. The visualizations include bar charts, line graphs, scatter plots, heatmaps, and more.

4. Machine learning: The project also uses machine learning techniques to analyze the data and make predictions. For example, the project uses a recommendation system to suggest similar athletes or countries based on their performance or attributes.

5. User interface: The project uses Streamlit, a popular Python library, to create a user-friendly interface for users to interact with the data. The interface includes several sections such as Medal Tally, Overall Analysis, Country Analysis, and Athlete Analysis, allowing users to explore the data in different ways.

2.3 Visualization Using Programming

In Python, we can perform data visualization using various libraries such as Matplotlib, Seaborn, Plotly, Bokeh, etc. Here is a general overview of how data visualization can be performed using Matplotlib and Seaborn libraries:

1. Matplotlib: Matplotlib is a popular data visualization library in Python. It provides a variety of customizable plots such as line plots, scatter plots, bar plots, histograms, etc. Here are the basic steps to create a plot using Matplotlib:
   - Import the Matplotlib library
   - Prepare the data
   - Create a figure and axis object using plt.subplots()
   - Plot the data using the appropriate plot function such as plot(), scatter(), bar(), etc. - Customize the plot by adding titles, labels, legends, colors, etc.

   - Save or display the plot using plt.savefig() or plt.show().

2. Seaborn: Seaborn is another data visualization library in Python that is built on top of Matplotlib. It provides a higher-level interface for creating statistical graphics such as heatmaps, violin plots, box plots, etc. Here are the basic steps to create a plot using Seaborn:
   - Import the Seaborn library
   - Load the data using Pandas or Numpy
   - Create a plot using the appropriate function such as sns.lineplot(), sns.scatterplot(), sns.boxplot(), - Customize the plot using the various built-in options such as hue, style, size, palette, etc. - Save or display the plot using plt.savefig() or plt.show().

3. Pandas:
   - Pandas is a popular Python library used for data manipulation and analysis.

- It provides data structures for efficiently storing and manipulating data in the form of data frames and series.
- Pandas makes it easy to import and export data from various file formats like CSV, Excel, SQL databases, and more.
- It also provides powerful tools for cleaning and transforming data, including methods for handling missing data, filtering, grouping, and joining data sets.

4. NumPy:
   - NumPy is a powerful Python library for scientific computing.
   - It provides a fast and efficient multi-dimensional array object, along with a large library of mathematical functions to operate on these arrays.
   - NumPy arrays are the backbone of many other scientific Python libraries and are used extensively in data analysis, machine learning, and scientific simulations.
   - NumPy also provides tools for linear algebra, Fourier transforms, and random number generation.

CHAPTER 3: REQUIREMENTS AND ANALYSIS

3.1 Introduction

The first activity fills in as a premise of giving the practical details, requirements and domain of the system, afterward successful plan for the proposed system. Understanding the properties and needs of the system is more complex and requires innovative thoughts.

3.2 Software Requirement

The product interface which is executed in this task is finished utilizing Python Language, and Visual Studio Code running in the Windows environment, and we are using Jupyter notebook for data data handling and data preprocessing

3.2.1 Python

Python is a computer programming language frequently used to construct sites and programming, robotize undertakings, and direct information examination. Python is a universally useful language, meaning it very well may be utilized to make a wide range of projects and isn't particular for a particular issue.

3.2.2 Visual Studio Code

A coordinated improvement climate (IDE) is an element-rich program that upholds numerous parts of programming advancement. The Visual Studio IDE is an inventive take-off platform that you can use to alter, investigate, fabricate code, distribute an application, and then some.

3.2.3 Jupyter Notebook

Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience. Through the web-based application, users can create data visualizations and other components of a project to share with others via the platform.

3.3 Hardware Requirement

Device name: LAPTOP-IF2Q6K39

Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz

Installed RAM: 8.00 GB (7.69 GB usable)

System type: 64-bit operating system, x64-based processor

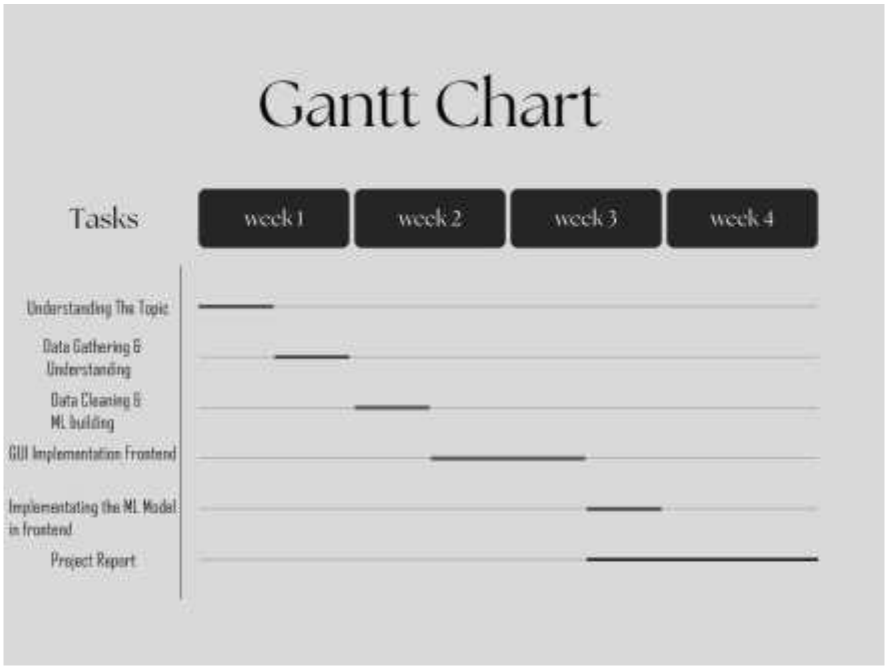
3.4 Data Requirement

3.4.1 TMDB athlete_event.csv file 3.4.2 and TMDB noc_regions.csv file

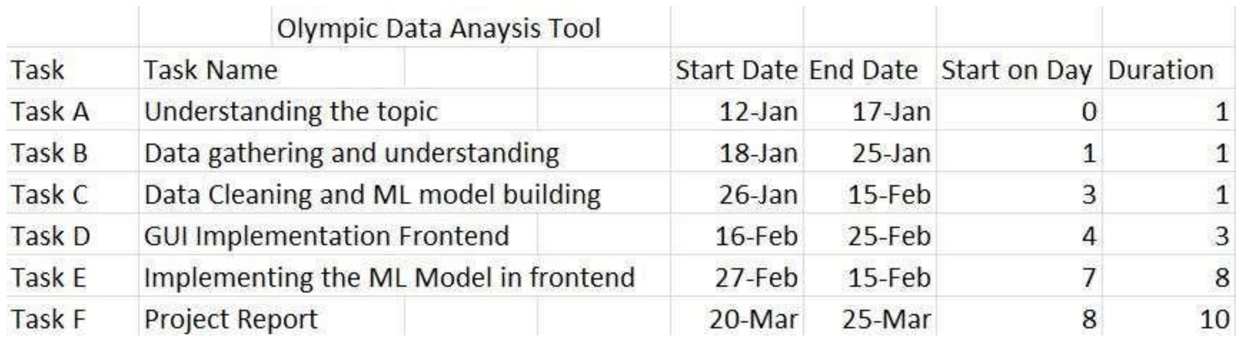
(We merge both files so we can work on the data easily and correctly)

3.5 Planning and Scheduling

3.5.1 GANTT Chart

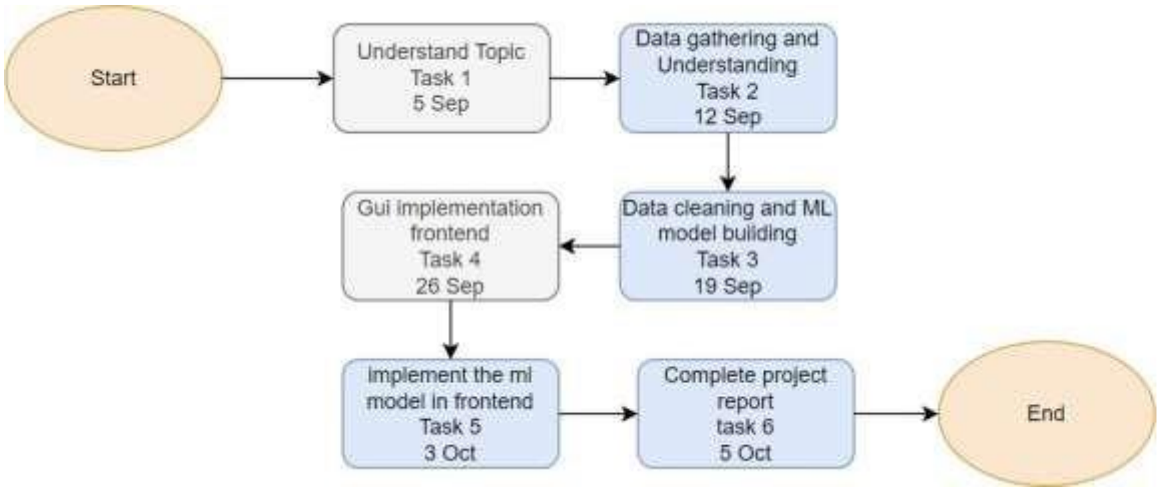


3.5.1.1 GANTT Chart table



	Olympic Data Anaysis Tool				
Task	Task Name	Start Date	End Date	Start on Day	Duration
Task A	Understanding the topic	12-Jan	17-Jan	0	1
Task B	Data gathering and understanding	18-Jan	25-Jan	1	1
Task C	Data Cleaning and ML model building	26-Jan	15-Feb	3	1
Task D	GUI Implementation Frontend	16-Feb	25-Feb	4	3
Task E	Implementing the ML Model in frontend	27-Feb	15-Feb	7	8
Task F	Project Report	20-Mar	25-Mar	8	10

3.5.2 PERT Chart (Work Break Down)


Start
Understand Topic Task 1 5 Sep
Data gathering and Understanding Task 2 12 Sep
Data cleaning and ML model building Task 3 19 Sep
Gui implementation frontend Task 4 26 Sep
implement the ml model in frontend Task 5 3 Oct
Complete project report task 6 5 Oct
End

3.6 Conceptual Models (UML Diagrams)

Software design is a course of critical thinking and making scheduling for a software solution. After the reason and determinations of software are determined, the software designer will design or employ designers to develop an arrangement for an answer. It contains the development part and calculation execution issues which are displayed in the architectural view. During this part, we will present a few guides that are considered through the software design

3.6.1 Structural View

Structure diagrams show the things in the proposed system. In more technical terms, they show various objects in a system.

3.6.1.1 Class diagram

A class diagram is a static outline. It addresses the static perspective on an application. A class diagram isn't just utilized for imagining, depicting, and reporting various parts of a framework yet additionally for developing executable code of the product application. A class diagram describes the properties and tasks of a class and furthermore, the limitations forced on the framework. The class graphs are broadly utilized in the displaying of object-oriented frameworks since they are the main UML charts, which can be planned straightforwardly with object-arranged dialects.

3.6.1.2 Component Diagram

A component diagram shows the structural relationship of parts of a system. Components speak with one another utilizing connection interfaces. The interfaces of interaction are connected utilizing connectors. The image underneath shows a general component diagram of movie recommender system.

3.6.3 Object Diagram

Object Diagrams, sometimes referred to as Instance diagrams are very identical to class diagrams. Like class diagrams, they additionally show the connection between objects yet they utilize real-world examples. Since there is data available in the objects, they are utilized to explain complicated connections between objects.

3.6.2 Behavioral View

As we referenced already the activity diagram, and sequence diagram give the behaviour view of our task. Behavioral diagrams are utilized to depict the communication between the actors and the system. Every one of the activities that are performed by the actors and the system is presented here and there.

3.6.1 Sequence Diagram

A sequence diagram shows object connections organized in a period grouping. It shows the objects and classes associated with the situation and the succession of messages traded between the objects expected to do the usefulness of the situation.

3.6.2 Activity Diagram

An activity diagram is fundamentally a flowchart to address the stream starting with one activity and then onto the next activity. The movement can be described as an activity of the system. The control stream is attracted starting with one activity and then onto the next. This stream can be sequential, branched, or concurrent.

3.6.3 Use case Diagram

In the Unified Modeling Language (UML), a use case diagram can sum up the subtleties of your system's clients (also called actors) and their collaborations with the system. To construct one, you'll utilize a bunch of particular symbols and connectors.

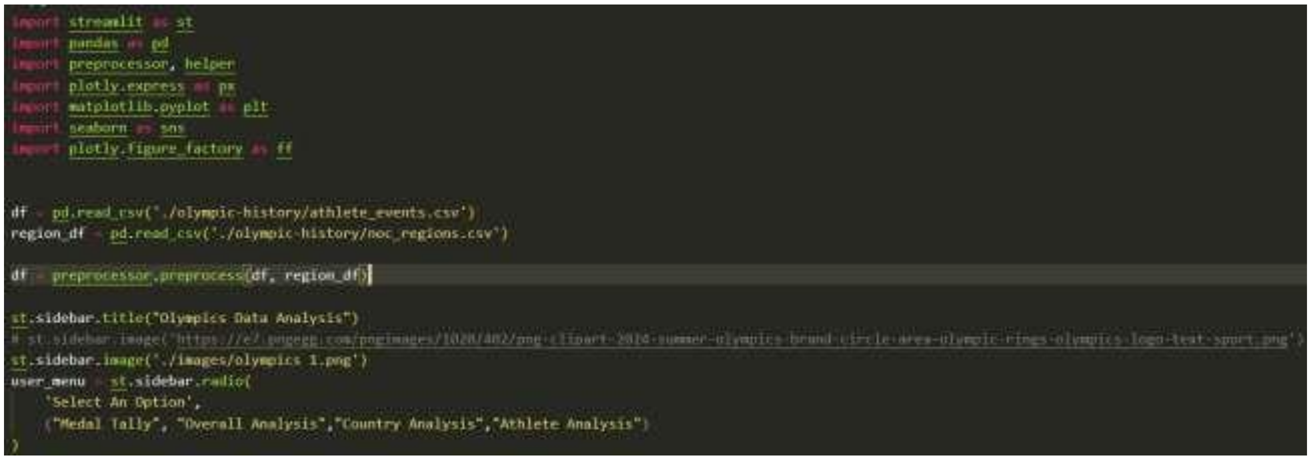
3.6.4 Data Flow Diagram (DFD)

A data flow diagram (DFD) is a graphical description of the "flow" of information through a data system, displaying its interaction viewpoints. Frequently they are a starter step used to make an outline of the system which can later be expounded. DFDs can likewise be utilized for the perception of information processing (structure design).

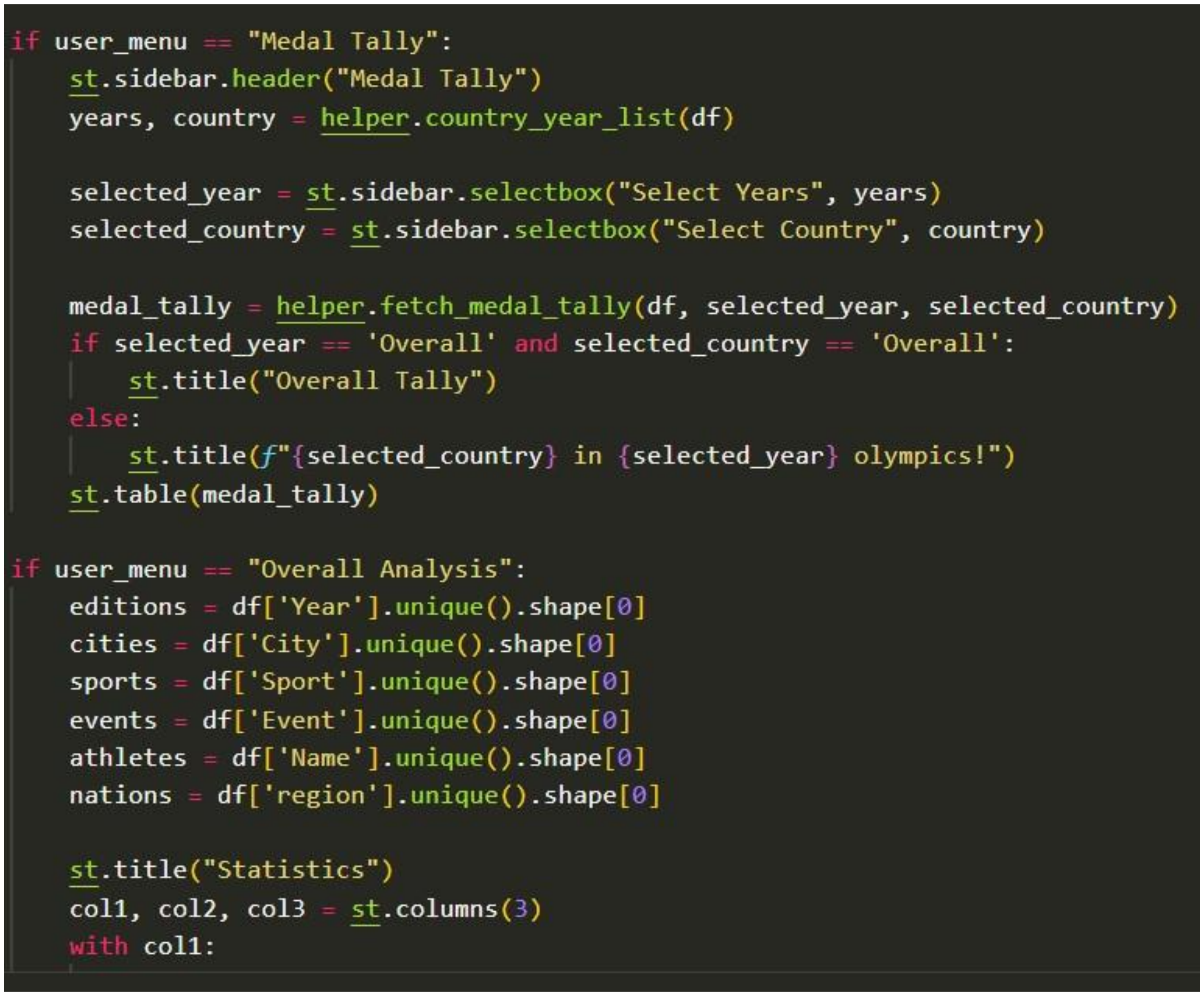
CHAPTER 4: SYSTEM DESIGN

4.1 Proposed System Code

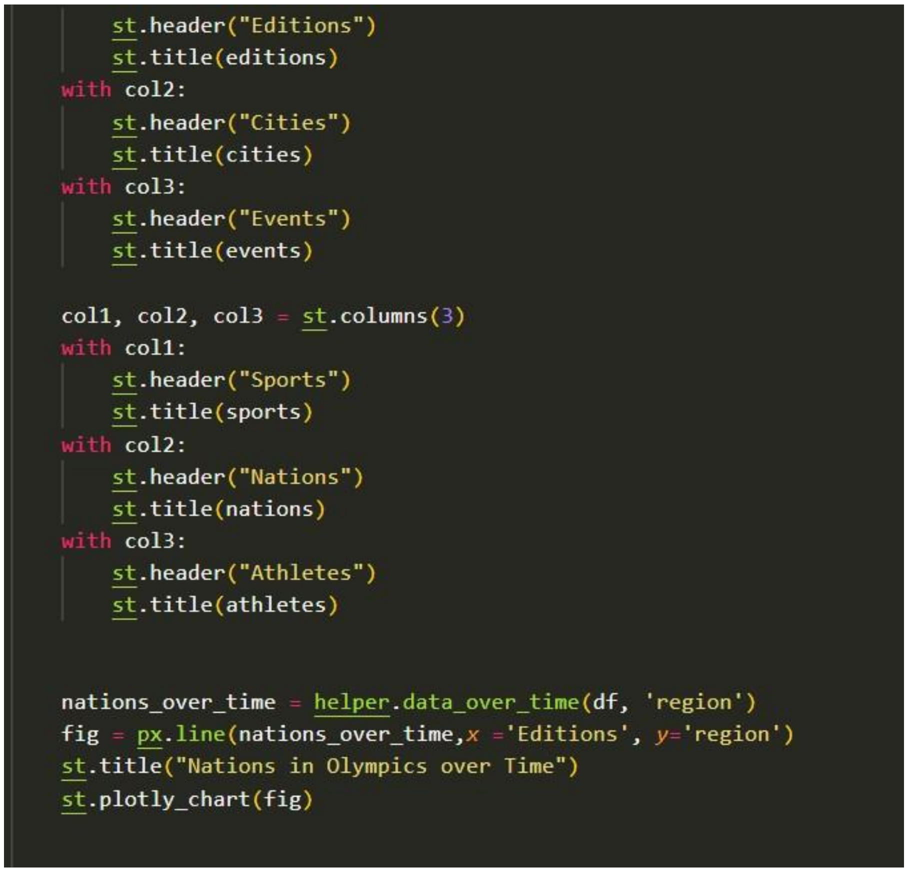
4.1.1 Front End Code (web application and implication of model are done in this file)(main.py):



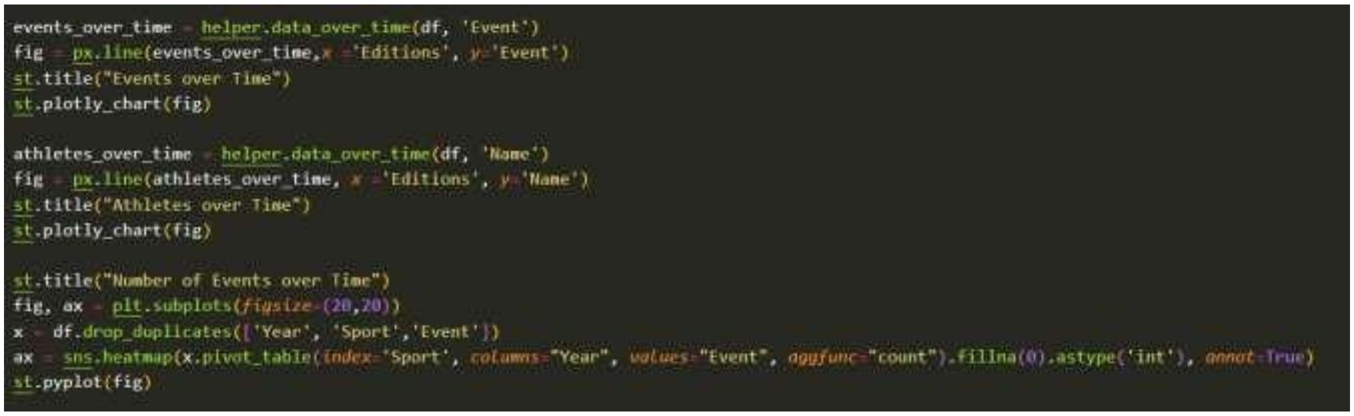
```python
import streamlit as st
import pandas as pd
import preprocessor, helper
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.figure_factory as ff


df = pd.read_csv('./olympic-history/athlete_events.csv')
region_df = pd.read_csv('./olympic-history/noc_regions.csv')

df = preprocessor.preprocess(df, region_df)

st.sidebar.title("Olympics Data Analysis")
# st.sidebar.image('https://e7.pngegg.com/pngimages/1020/402/png-clipart-2016-summer-olympics-brand-circle-area-olympic-rings-olympics-logo-text-sport.png')
st.sidebar.image('./images/olympics 1.png')
user_menu = st.sidebar.radio(
    'Select An Option',
    ("Medal Tally", "Overall Analysis","Country Analysis","Athlete Analysis")
)
```



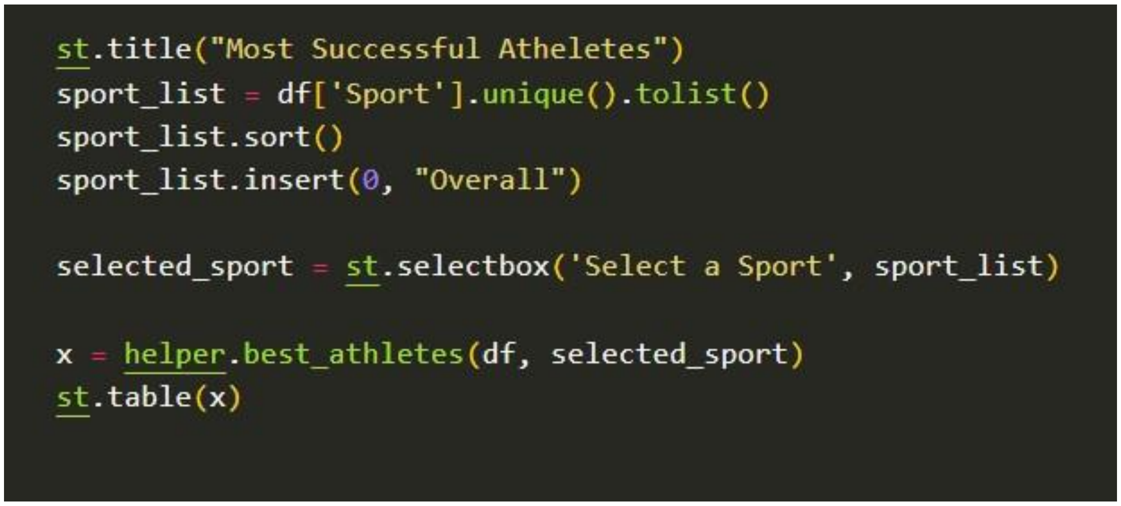
```python
if user_menu == "Medal Tally":
    st.sidebar.header("Medal Tally")
    years, country = helper.country_year_list(df)

    selected_year = st.sidebar.selectbox("Select Years", years)
    selected_country = st.sidebar.selectbox("Select Country", country)

    medal_tally = helper.fetch_medal_tally(df, selected_year, selected_country)
    if selected_year == 'Overall' and selected_country == 'Overall':
        st.title("Overall Tally")
    else:
        st.title(f"{selected_country} in {selected_year} olympics!")
    st.table(medal_tally)

if user_menu == "Overall Analysis":
    editions = df['Year'].unique().shape[0]
    cities = df['City'].unique().shape[0]
    sports = df['Sport'].unique().shape[0]
    events = df['Event'].unique().shape[0]
    athletes = df['Name'].unique().shape[0]
    nations = df['region'].unique().shape[0]

    st.title("Statistics")
    col1, col2, col3 = st.columns(3)
    with col1:
```



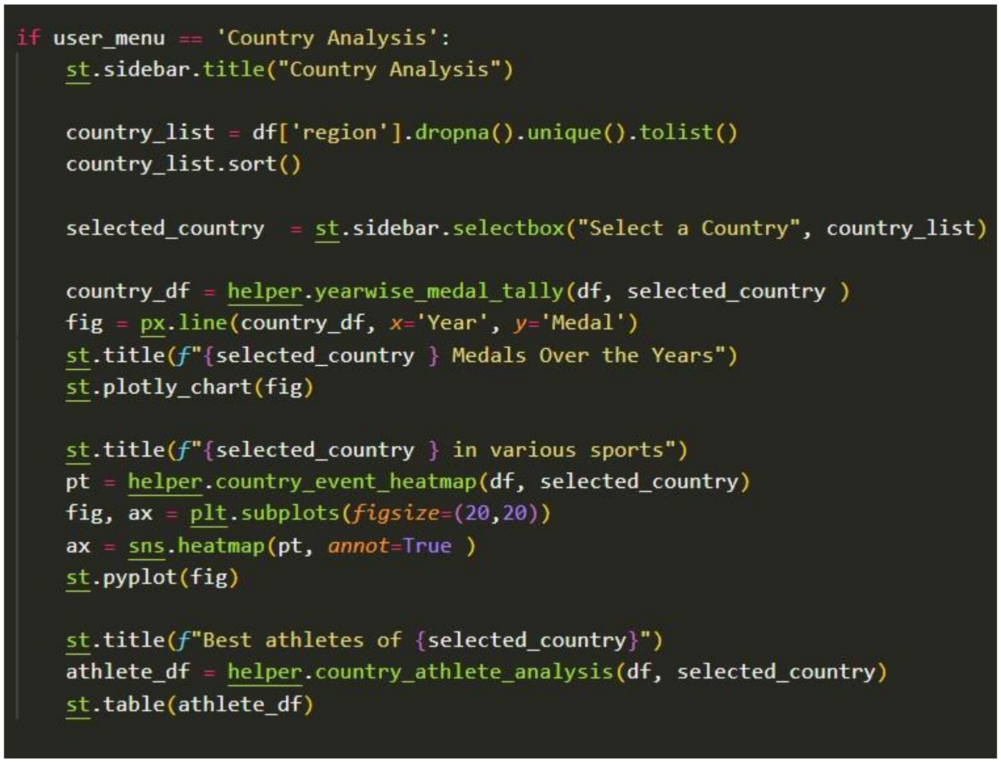
```
        st.header("Editions")
        st.title(editions)
    with col2:
        st.header("Cities")
        st.title(cities)
    with col3:
        st.header("Events")
        st.title(events)

    col1, col2, col3 = st.columns(3)
    with col1:
        st.header("Sports")
        st.title(sports)
    with col2:
        st.header("Nations")
        st.title(nations)
    with col3:
        st.header("Athletes")
        st.title(athletes)


    nations_over_time = helper.data_over_time(df, 'region')
    fig = px.line(nations_over_time,x ='Editions', y='region')
    st.title("Nations in Olympics over Time")
    st.plotly_chart(fig)
```



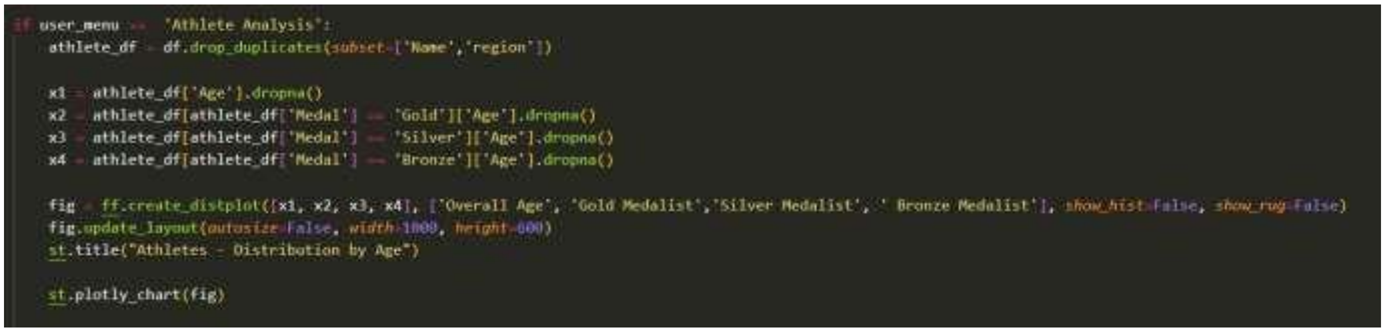
```
events_over_time = helper.data_over_time(df, 'Event')
fig = px.line(events_over_time,x ='Editions', y='Event')
st.title("Events over Time")
st.plotly_chart(fig)

athletes_over_time = helper.data_over_time(df, 'Name')
fig = px.line(athletes_over_time, x ='Editions', y='Name')
st.title("Athletes over Time")
st.plotly_chart(fig)

st.title("Number of Events over Time")
fig, ax = plt.subplots(figsize=(20,20))
x = df.drop_duplicates(['Year', 'Sport','Event'])
ax = sns.heatmap(x.pivot_table(index='Sport', columns="Year", values="Event", aggfunc="count").fillna(0).astype('int'), annot=True)
st.pyplot(fig)
```



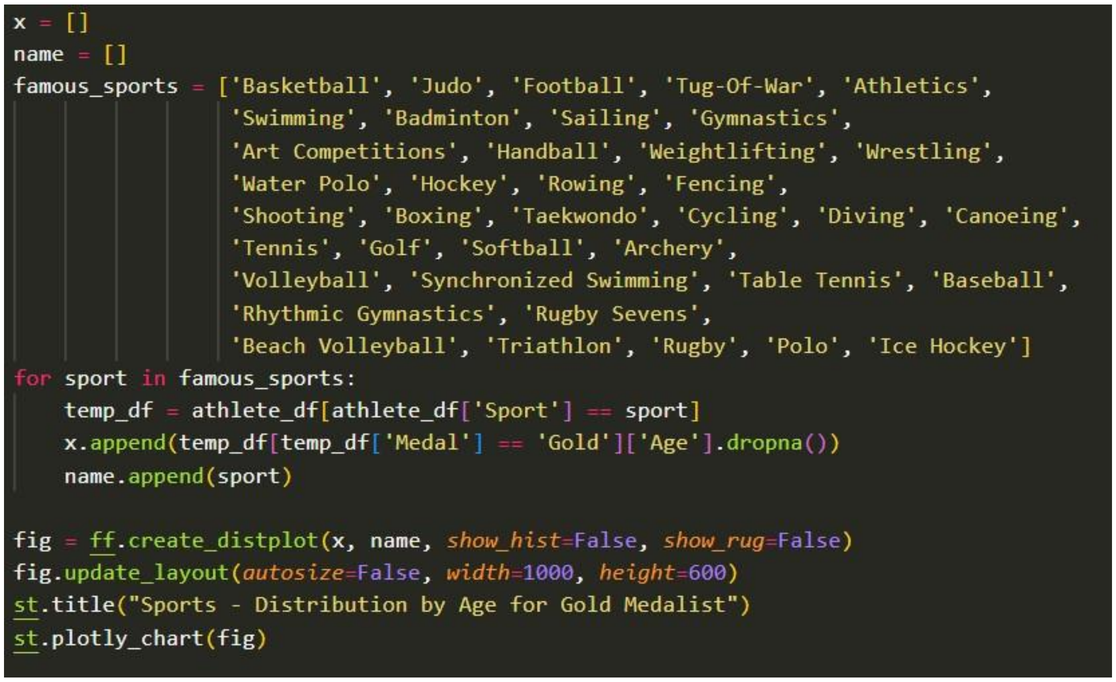
```python
st.title("Most Successful Atheletes")
sport_list = df['Sport'].unique().tolist()
sport_list.sort()
sport_list.insert(0, "Overall")

selected_sport = st.selectbox('Select a Sport', sport_list)

x = helper.best_athletes(df, selected_sport)
st.table(x)
```



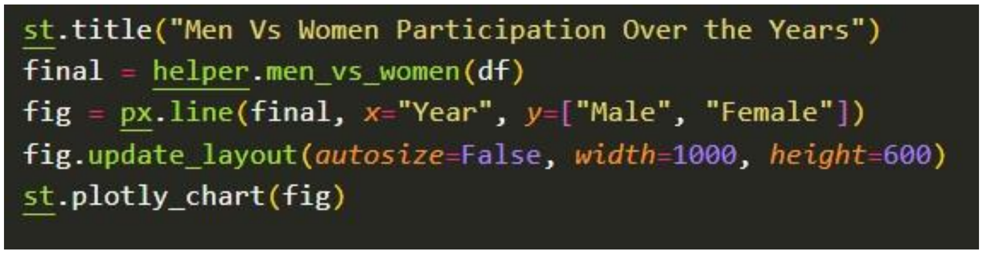
```python
if user_menu == 'Country Analysis':
    st.sidebar.title("Country Analysis")

    country_list = df['region'].dropna().unique().tolist()
    country_list.sort()

    selected_country  = st.sidebar.selectbox("Select a Country", country_list)

    country_df = helper.yearwise_medal_tally(df, selected_country )
    fig = px.line(country_df, x='Year', y='Medal')
    st.title(f"{selected_country } Medals Over the Years")
    st.plotly_chart(fig)

    st.title(f"{selected_country } in various sports")
    pt = helper.country_event_heatmap(df, selected_country)
    fig, ax = plt.subplots(figsize=(20,20))
    ax = sns.heatmap(pt, annot=True )
    st.pyplot(fig)

    st.title(f"Best athletes of {selected_country}")
    athlete_df = helper.country_athlete_analysis(df, selected_country)
    st.table(athlete_df)
```



```python
if user_menu == 'Athlete Analysis':
    athlete_df = df.drop_duplicates(subset=['Name','region'])

    x1 = athlete_df['Age'].dropna()
    x2 = athlete_df[athlete_df['Medal'] == 'Gold']['Age'].dropna()
    x3 = athlete_df[athlete_df['Medal'] == 'Silver']['Age'].dropna()
    x4 = athlete_df[athlete_df['Medal'] == 'Bronze']['Age'].dropna()

    fig = ff.create_distplot([x1, x2, x3, x4], ['Overall Age', 'Gold Medalist','Silver Medalist', ' Bronze Medalist'], show_hist=False, show_rug=False)
    fig.update_layout(autosize=False, width=1000, height=600)
    st.title("Athletes - Distribution by Age")

    st.plotly_chart(fig)
```

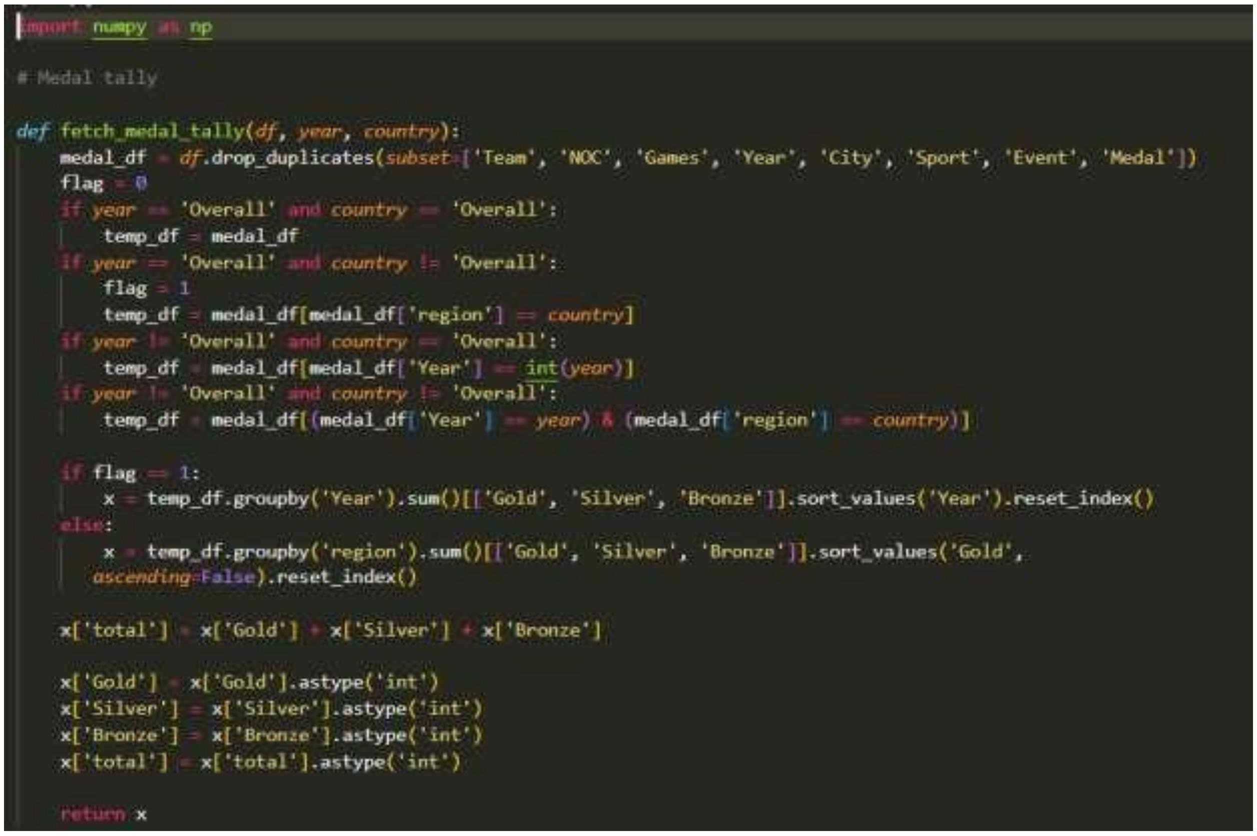


```python
x = []
name = []
famous_sports = ['Basketball', 'Judo', 'Football', 'Tug-Of-War', 'Athletics',
                 'Swimming', 'Badminton', 'Sailing', 'Gymnastics',
                 'Art Competitions', 'Handball', 'Weightlifting', 'Wrestling',
                 'Water Polo', 'Hockey', 'Rowing', 'Fencing',
                 'Shooting', 'Boxing', 'Taekwondo', 'Cycling', 'Diving', 'Canoeing',
                 'Tennis', 'Golf', 'Softball', 'Archery',
                 'Volleyball', 'Synchronized Swimming', 'Table Tennis', 'Baseball',
                 'Rhythmic Gymnastics', 'Rugby Sevens',
                 'Beach Volleyball', 'Triathlon', 'Rugby', 'Polo', 'Ice Hockey']
for sport in famous_sports:
    temp_df = athlete_df[athlete_df['Sport'] == sport]
    x.append(temp_df[temp_df['Medal'] == 'Gold']['Age'].dropna())
    name.append(sport)

fig = ff.create_distplot(x, name, show_hist=False, show_rug=False)
fig.update_layout(autosize=False, width=1000, height=600)
st.title("Sports - Distribution by Age for Gold Medalist")
st.plotly_chart(fig)
```

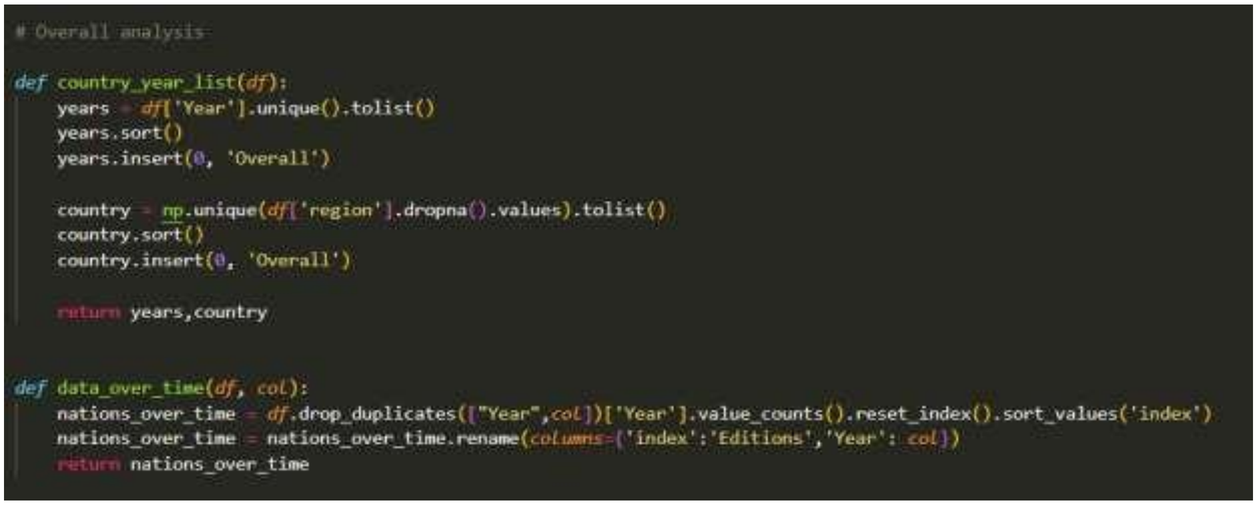


```python
st.title("Men Vs Women Participation Over the Years")
final = helper.men_vs_women(df)
fig = px.line(final, x="Year", y=["Male", "Female"])
fig.update_layout(autosize=False, width=1000, height=600)
st.plotly_chart(fig)
```

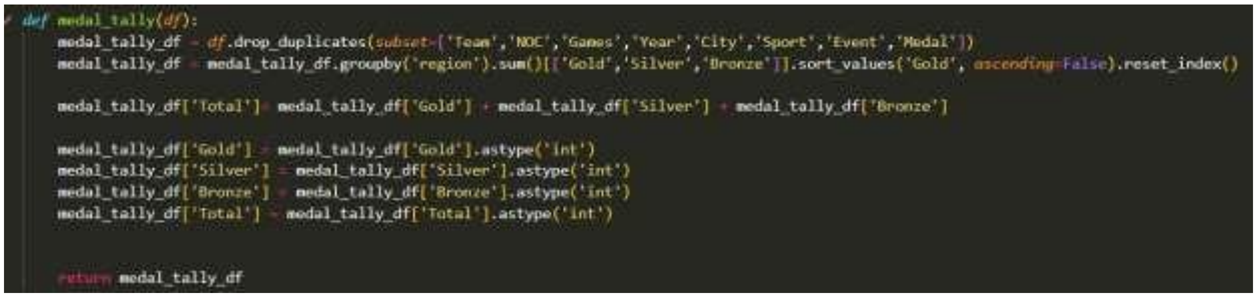
4.2.2 Backend Code (helper.py)



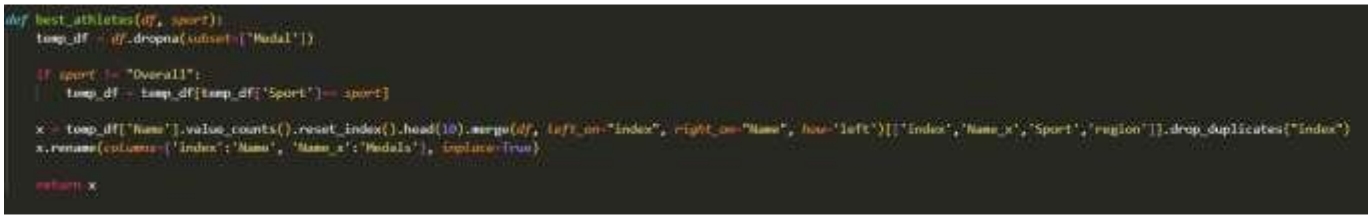
```python
import numpy as np

# Medal tally

def fetch_medal_tally(df, year, country):
    medal_df = df.drop_duplicates(subset=['Team', 'NOC', 'Games', 'Year', 'City', 'Sport', 'Event', 'Medal'])
    flag = 0
    if year == 'Overall' and country == 'Overall':
        temp_df = medal_df
    if year == 'Overall' and country != 'Overall':
        flag = 1
        temp_df = medal_df[medal_df['region'] == country]
    if year != 'Overall' and country == 'Overall':
        temp_df = medal_df[medal_df['Year'] == int(year)]
    if year != 'Overall' and country != 'Overall':
        temp_df = medal_df[(medal_df['Year'] == year) & (medal_df['region'] == country)]

    if flag == 1:
        x = temp_df.groupby('Year').sum()[['Gold', 'Silver', 'Bronze']].sort_values('Year').reset_index()
    else:
        x = temp_df.groupby('region').sum()[['Gold', 'Silver', 'Bronze']].sort_values('Gold',
       ascending=False).reset_index()

    x['total'] = x['Gold'] + x['Silver'] + x['Bronze']

    x['Gold'] = x['Gold'].astype('int')
    x['Silver'] = x['Silver'].astype('int')
    x['Bronze'] = x['Bronze'].astype('int')
    x['total'] = x['total'].astype('int')

    return x
```



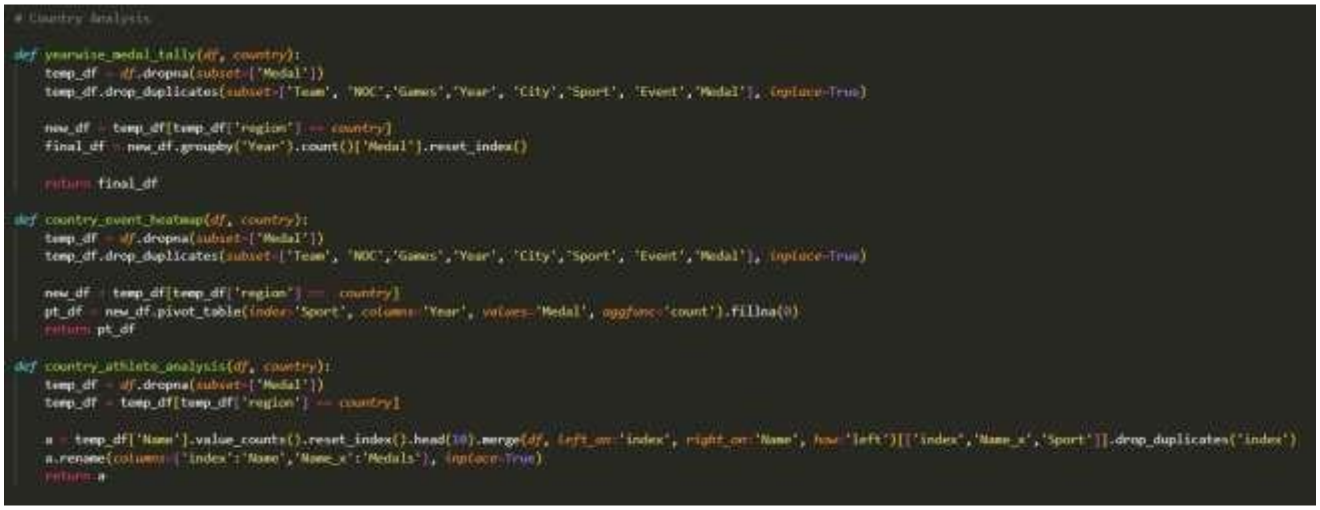
```python
# Overall analysis

def country_year_list(df):
    years = df['Year'].unique().tolist()
    years.sort()
    years.insert(0, 'Overall')

    country = np.unique(df['region'].dropna().values).tolist()
    country.sort()
    country.insert(0, 'Overall')

    return years,country


def data_over_time(df, col):
    nations_over_time = df.drop_duplicates(["Year",col])['Year'].value_counts().reset_index().sort_values('index')
    nations_over_time = nations_over_time.rename(columns={'index':'Editions','Year': col})
    return nations_over_time
```



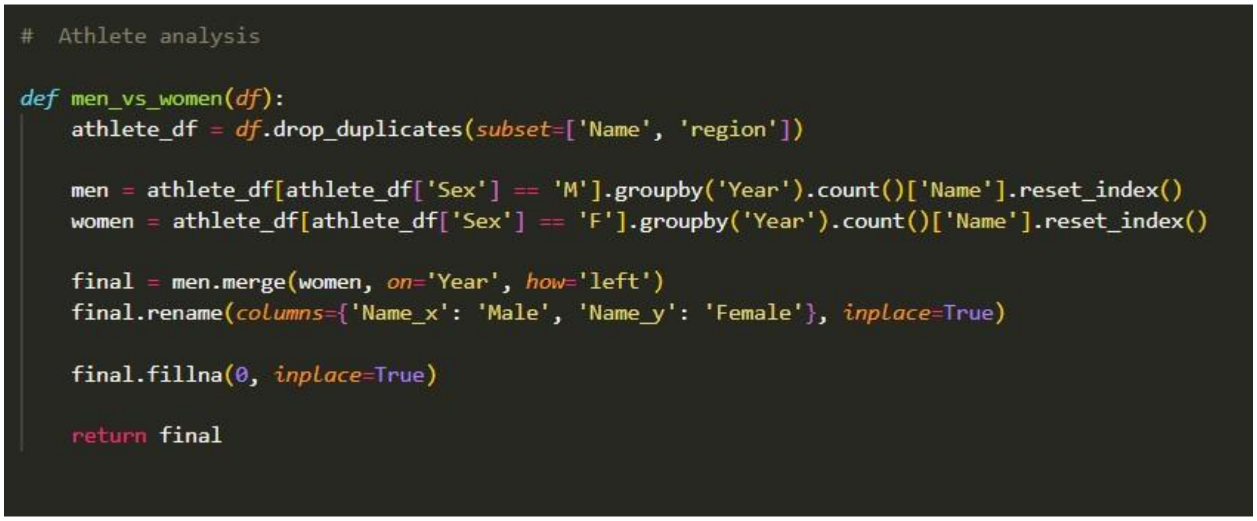
```python
def medal_tally(df):
    medal_tally_df = df.drop_duplicates(subset=['Team','NOC','Games','Year','City','Sport','Event','Medal'])
    medal_tally_df = medal_tally_df.groupby('region').sum()[['Gold','Silver','Bronze']].sort_values('Gold', ascending=False).reset_index()

    medal_tally_df['Total']= medal_tally_df['Gold'] + medal_tally_df['Silver'] + medal_tally_df['Bronze']

    medal_tally_df['Gold'] = medal_tally_df['Gold'].astype('int')
    medal_tally_df['Silver'] = medal_tally_df['Silver'].astype('int')
    medal_tally_df['Bronze'] = medal_tally_df['Bronze'].astype('int')
    medal_tally_df['Total'] = medal_tally_df['Total'].astype('int')


    return medal_tally_df
```



```python
def best_athletes(df, sport):
    temp_df = df.dropna(subset=['Medal'])

    if sport != "Overall":
        temp_df = temp_df[temp_df['Sport']== sport]

    x = temp_df['Name'].value_counts().reset_index().head(10).merge(df, left_on="index", right_on="Name", how='left')[['index','Name_x','Sport','region']].drop_duplicates("index")
    x.rename(columns={'index':'Name', 'Name_x':'Medals'}, inplace=True)

    return x
```

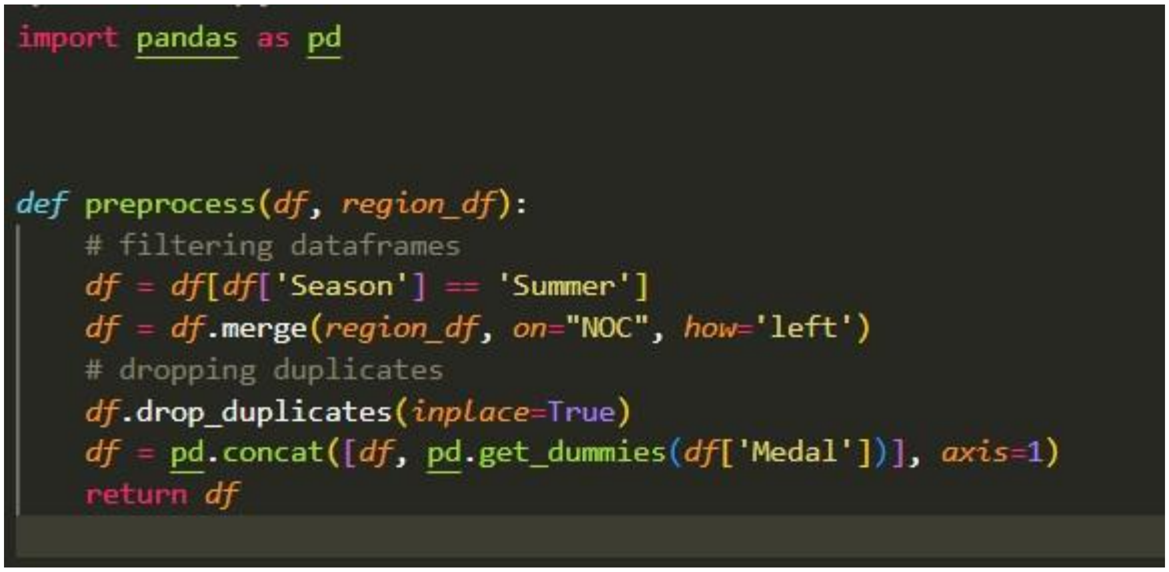


```python
# Country Analysis

def yearwise_medal_tally(df, country):
    temp_df = df.dropna(subset=['Medal'])
    temp_df.drop_duplicates(subset=['Team', 'NOC','Games','Year', 'City','Sport', 'Event','Medal'], inplace=True)

    new_df = temp_df[temp_df['region'] == country]
    final_df = new_df.groupby('Year').count()['Medal'].reset_index()

    return final_df

def country_event_heatmap(df, country):
    temp_df = df.dropna(subset=['Medal'])
    temp_df.drop_duplicates(subset=['Team', 'NOC','Games','Year', 'City','Sport', 'Event','Medal'], inplace=True)

    new_df = temp_df[temp_df['region'] ==  country]
    pt_df = new_df.pivot_table(index='Sport', columns='Year', values='Medal', aggfunc='count').fillna(0)
    return pt_df

def country_athlete_analysis(df, country):
    temp_df = df.dropna(subset=['Medal'])
    temp_df = temp_df[temp_df['region'] == country]

    a = temp_df['Name'].value_counts().reset_index().head(10).merge(df, left_on='index', right_on='Name', how='left')[['index','Name_x','Sport']].drop_duplicates('index')
    a.rename(columns={'index':'Name','Name_x':'Medals'}, inplace=True)
    return a
```



```python
#  Athlete analysis

def men_vs_women(df):
    athlete_df = df.drop_duplicates(subset=['Name', 'region'])

    men = athlete_df[athlete_df['Sex'] == 'M'].groupby('Year').count()['Name'].reset_index()
    women = athlete_df[athlete_df['Sex'] == 'F'].groupby('Year').count()['Name'].reset_index()

    final = men.merge(women, on='Year', how='left')
    final.rename(columns={'Name_x': 'Male', 'Name_y': 'Female'}, inplace=True)

    final.fillna(0, inplace=True)

    return final
```

4.2.3 Backend Code (preprocessor.py)



```python
import pandas as pd



def preprocess(df, region_df):
    # filtering dataframes
    df = df[df['Season'] == 'Summer']
    df = df.merge(region_df, on="NOC", how='left')
    # dropping duplicates
    df.drop_duplicates(inplace=True)
    df = pd.concat([df, pd.get_dummies(df['Medal'])], axis=1)
    return df
```

4.2 Code Brief Explanation

- Front-End (web application)

1. The code is showing an application built using the Streamlit framework to analyze the Olympics data. The application has four main menus, namely, "Medal Tally," "Overall Analysis," "Country Analysis," and "Athlete Analysis."

2. The "Medal Tally" section allows the user to select a year and a country and view the medal tally for that combination of year and country. If the user selects "Overall" for both year and country, then the overall medal tally is displayed.

3. The "Overall Analysis" section displays various statistics related to the Olympics, such as the number of editions, cities, sports, events, athletes, and nations. It also displays the number of nations, events, and athletes over time. The section also shows a heatmap of the number of events for each sport over time and a table of the most successful athletes in a particular sport.

4. The "Country Analysis" section allows the user to select a country and view the medal tally for that country over the years. It also displays a heatmap of the number of events that the country has participated in for each sport and a table of the best athletes of that country.

5. The "Athlete Analysis" section displays a distribution plot of the age of athletes who won medals. It also shows four subplots for the age distribution of athletes who won gold, silver, and bronze medals, respectively

- Backend Code (Data Preprocessing Code)

1. fetch_medal_tally: this function takes a DataFrame containing Olympic Games data, a year and a country as inputs, and returns a DataFrame containing the medal tally for the given year and country.

2. country_year_list: this function takes a DataFrame containing Olympic Games data as input, and returns a tuple containing a list of years and a list of countries.

3. data_over_time: this function takes a DataFrame containing Olympic Games data and a column name as inputs, and returns a DataFrame containing the number of unique values in the given column over time.

4. medal_tally: this function takes a DataFrame containing Olympic Games data as input, and returns a DataFrame containing the overall medal tally for all countries.

5. best_athletes: this function takes a DataFrame containing Olympic Games data and a sport as inputs, and returns a DataFrame containing the top 10 athletes (by number of medals won) in the given sport.

6. yearwise_medal_tally: this function takes a DataFrame containing Olympic Games data and a country as inputs, and returns a DataFrame containing the medal tally for the given country over time.

7. country_event_heatmap: this function takes a DataFrame containing Olympic Games data and a country as inputs, and returns a DataFrame containing the number of medals won by the given country in each sport over time.

8. country_athlete_analysis: this function takes a DataFrame containing Olympic Games data and a country as inputs, and returns a DataFrame containing the top 10 athletes (by number of medals won) for the given country.

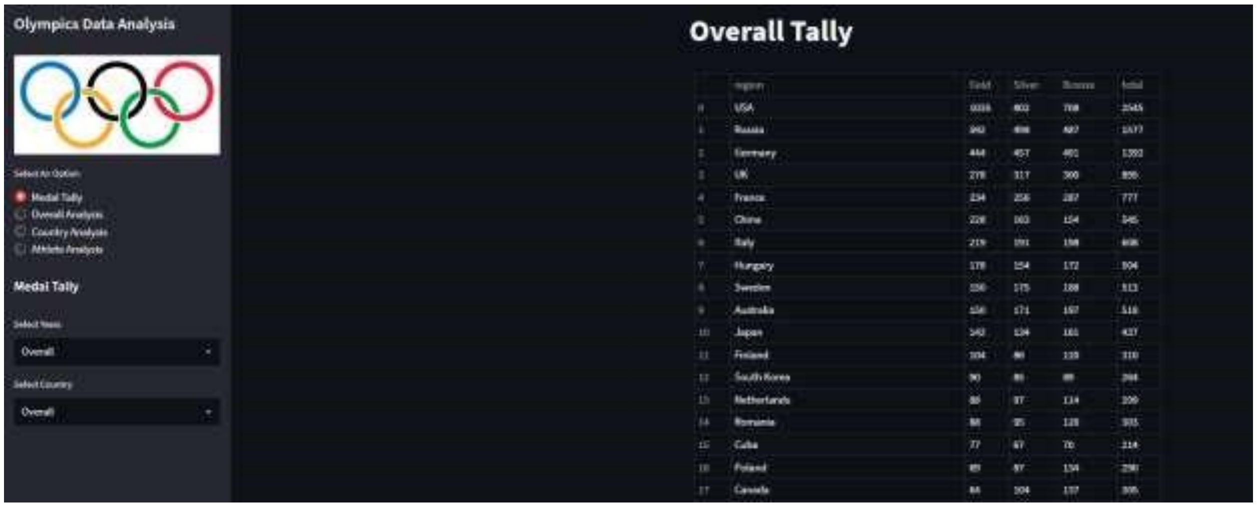
9. men_vs_women: this function takes a Data Frame containing Olympic Games data as input, and returns a Data Frame containing the number of male and female athletes over time Data Frame

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 Screenshots of the system

5.1.1 Initializing the System

- Medal Tally Interface



Olympics Data Analysis

Select An Option

[✓ Medal Tally]
Overall Analysis
Country Analysis
Athlete Analysis

Medal Tally

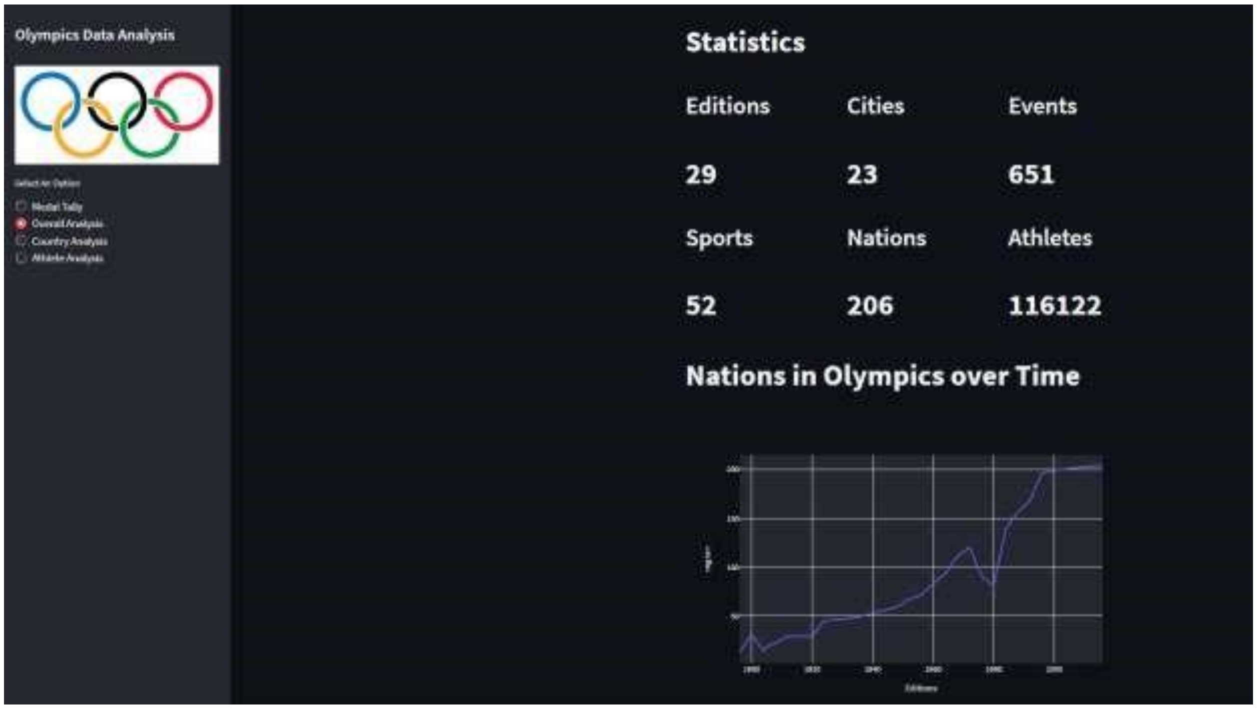
Select Years: Overall

Select Country: Overall

Overall Tally

	region	Gold	Silver	Bronze	total
0	USA	1035	802	708	2545
1	Russia	592	498	487	1577
2	Germany	444	457	491	1392
3	UK	278	317	300	895
4	France	234	256	287	777
5	China	228	163	154	545
6	Italy	219	191	198	608
7	Hungary	178	154	172	504
8	Sweden	150	175	188	513
9	Australia	150	171	195	516
10	Japan	142	134	161	437
11	Finland	104	86	120	310
12	South Korea	90	85	89	264
13	Netherlands	88	97	114	299
14	Romania	88	95	120	303
15	Cuba	77	67	70	214
16	Poland	65	87	138	290
17	Canada	64	104	137	305

- Overall Analysis Interface



Olympics Data Analysis

Select An Option

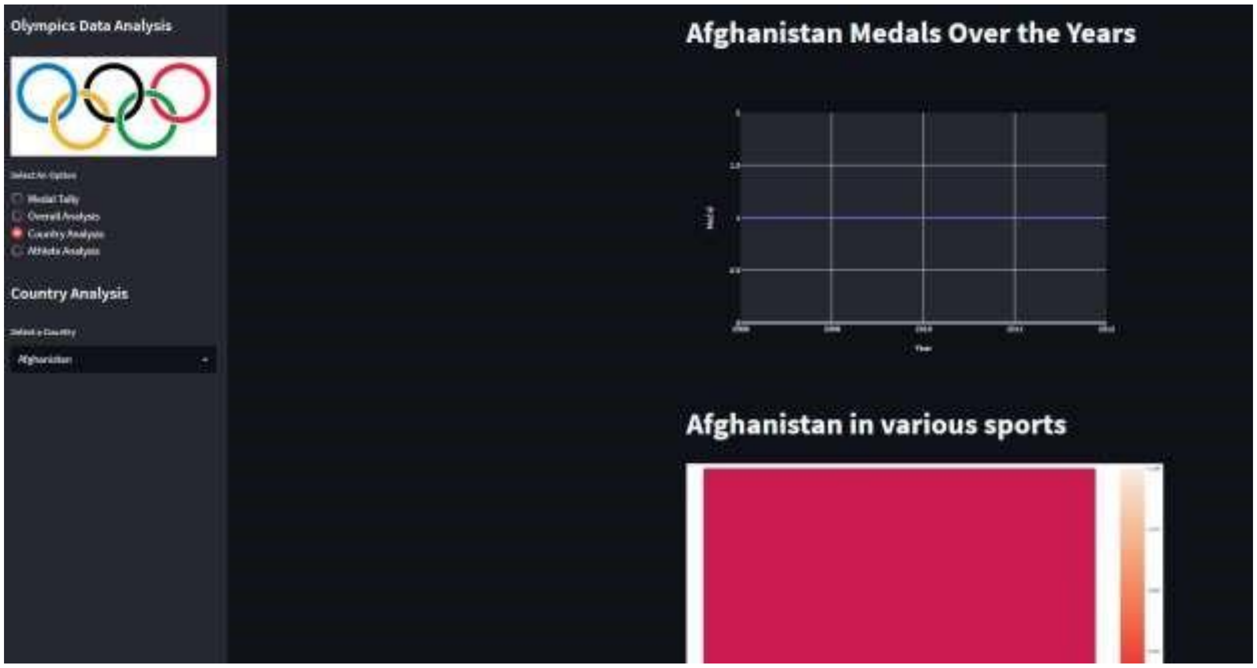
Medal Tally
[✓ Overall Analysis]
Country Analysis
Athlete Analysis

Statistics

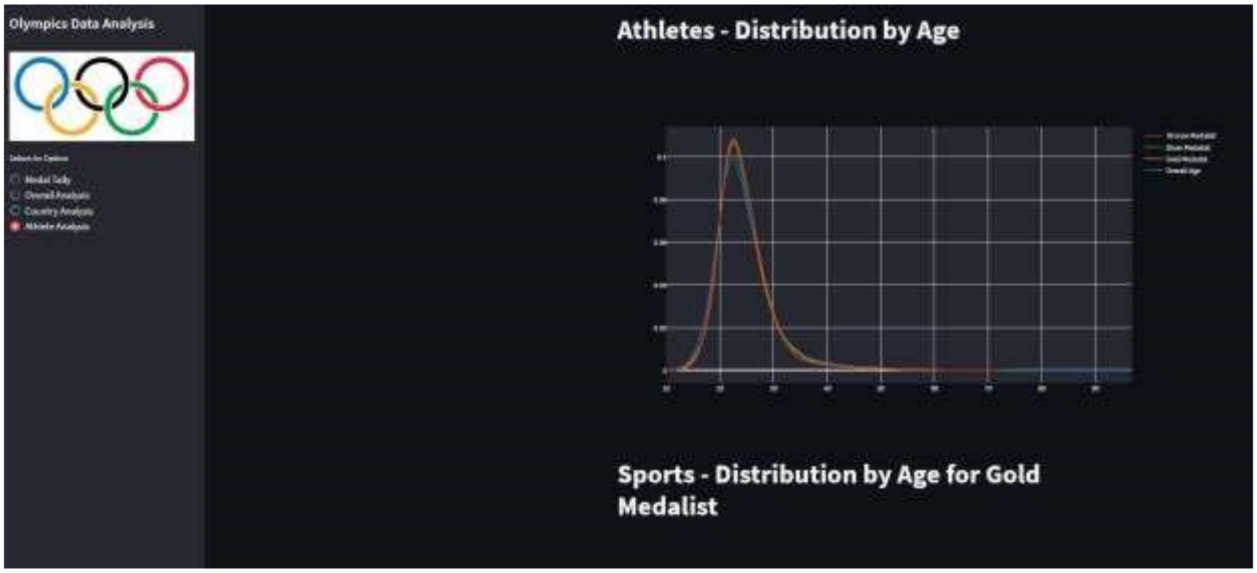
Editions	Cities	Events
29	23	651
Sports	**Nations**	**Athletes**
52	206	116122

Nations in Olympics over Time

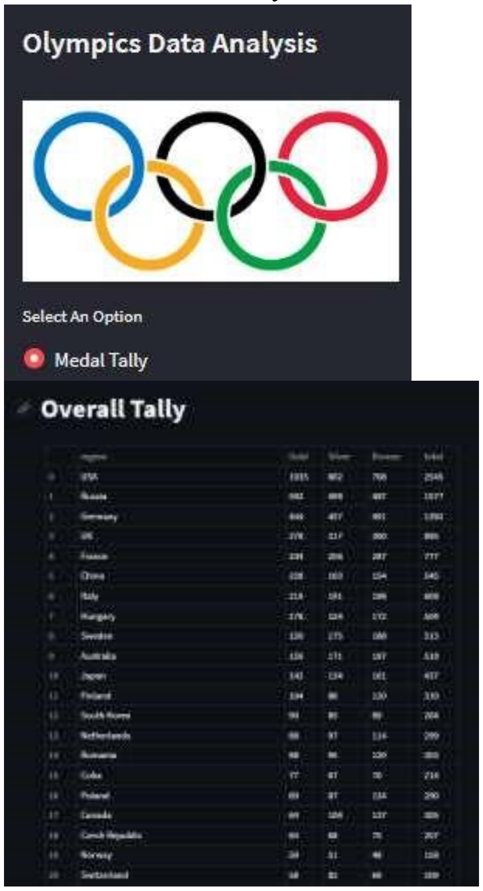
- Country Analysis Interface



- Athlete Analysis Interface



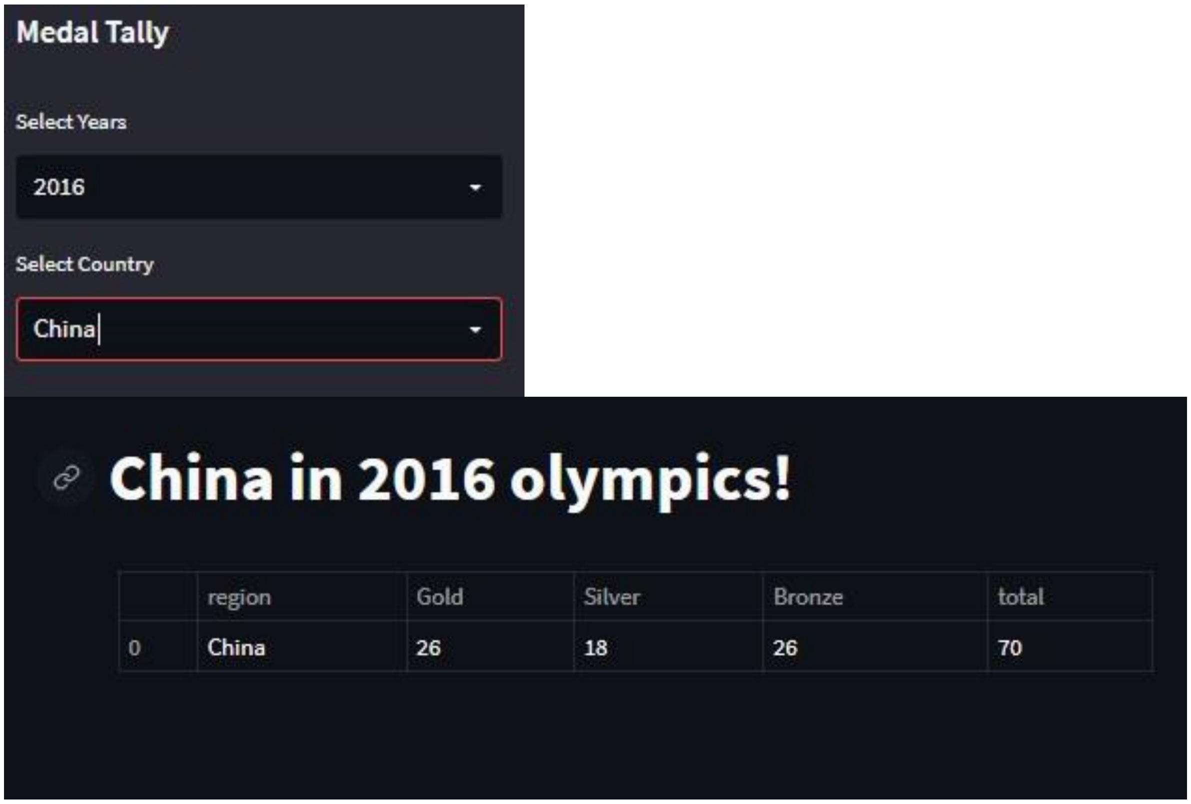
5.1.2 Select the Medal Tally



Overall Tally

	region	Gold	Silver	Bronze	total
0	USA	1035	802	708	2545
1	Russia	592	498	487	1577
2	Germany	444	457	491	1392
3	UK	278	317	300	895
4	France	234	256	287	777
5	China	228	163	154	545
6	Italy	219	191	199	609
7	Hungary	178	154	172	504
8	Sweden	150	175	188	513
9	Australia	150	171	197	518
10	Japan	142	134	161	437
11	Finland	104	86	120	310
12	South Korea	90	85	89	264
13	Netherlands	88	97	114	299
14	Romania	88	94	120	302
15	Cuba	77	67	70	214
16	Poland	69	87	134	290
17	Canada	69	104	137	310
18	Czech Republic	64	68	75	207
19	Norway	28	31	46	105
20	Switzerland	28	32	49	109

5.1.3 Analysis of Medal Tally



Medal Tally

Select Years

2016

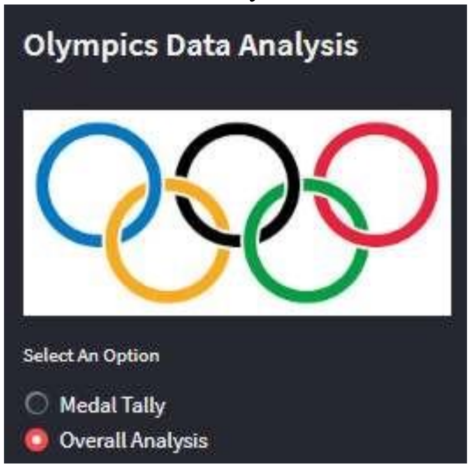
Select Country

China

China in 2016 olympics!

	region	Gold	Silver	Bronze	total
0	China	26	18	26	70

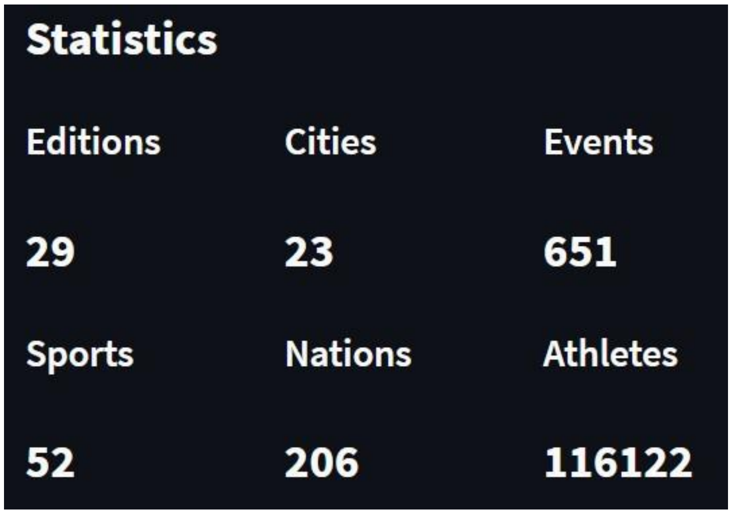
5.1.4 Select Overall Analysis



Olympics Data Analysis

Select An Option

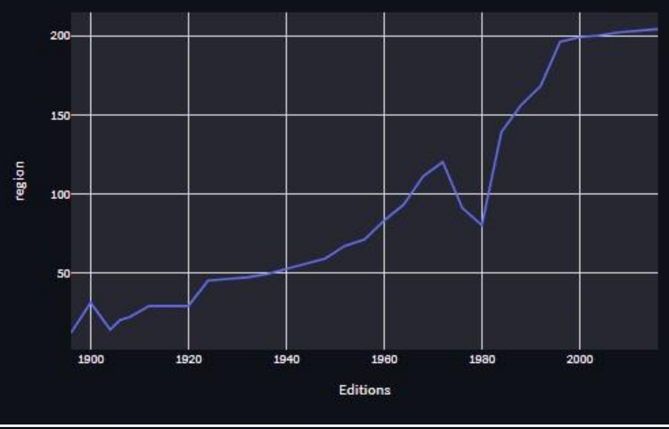
Medal Tally [✓ Overall Analysis]



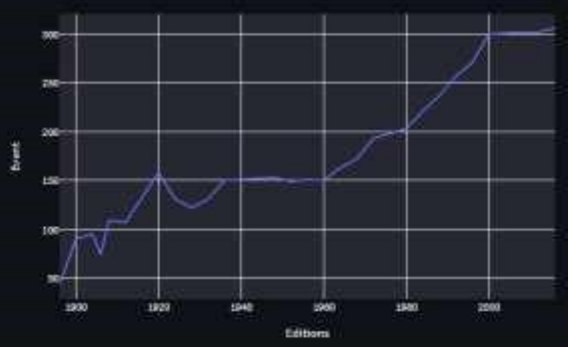
Statistics

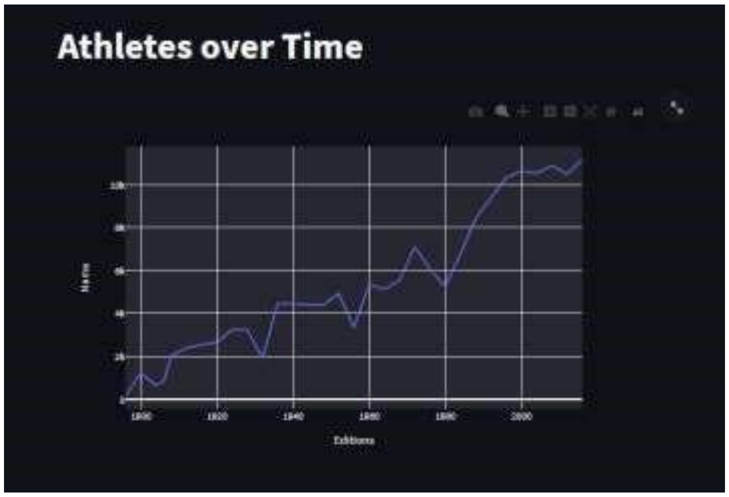
Editions	Cities	Events
29	23	651
Sports	**Nations**	**Athletes**
52	206	116122

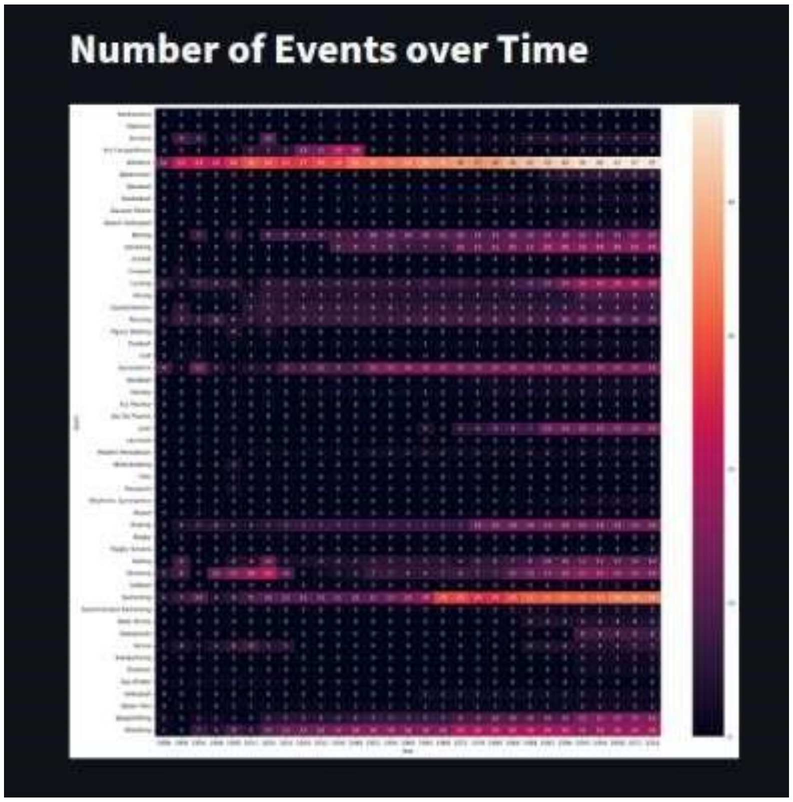
Nations in Olympics over Time



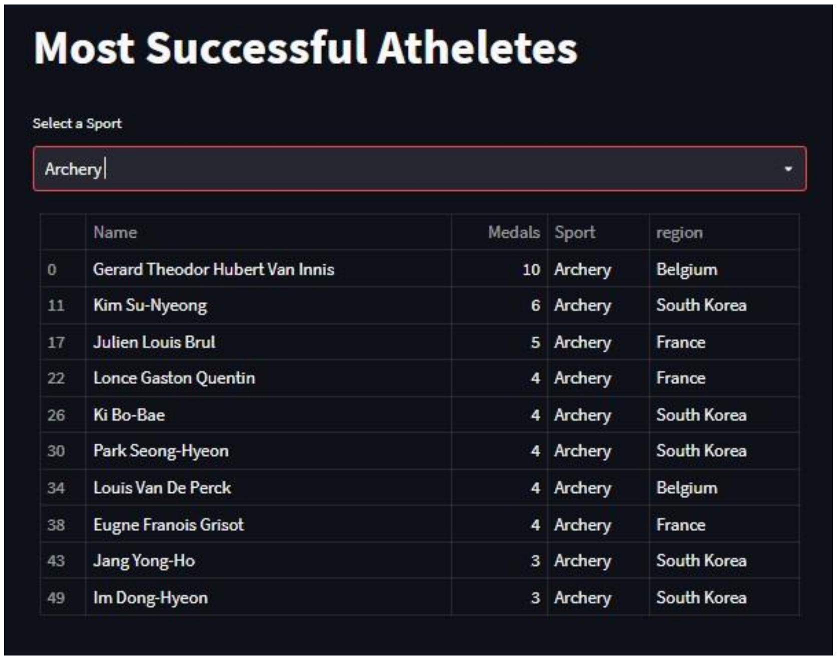
Events over Time







5.1.5 Select Particular sports in overall analysis



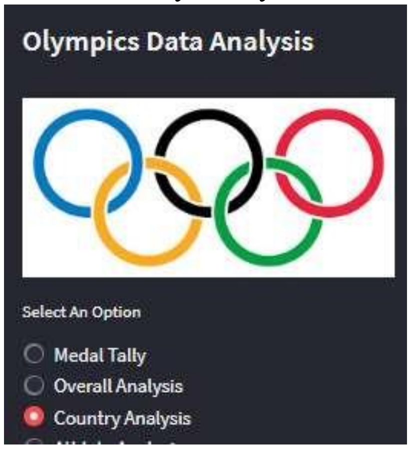
Most Successful Atheletes

Select a Sport

Archery

	Name	Medals	Sport	region
0	Gerard Theodor Hubert Van Innis	10	Archery	Belgium
11	Kim Su-Nyeong	6	Archery	South Korea
17	Julien Louis Brul	5	Archery	France
22	Lonce Gaston Quentin	4	Archery	France
26	Ki Bo-Bae	4	Archery	South Korea
30	Park Seong-Hyeon	4	Archery	South Korea
34	Louis Van De Perck	4	Archery	Belgium
38	Eugne Franois Grisot	4	Archery	France
43	Jang Yong-Ho	3	Archery	South Korea
49	Im Dong-Hyeon	3	Archery	South Korea

5.1.6 Select Country Analysis

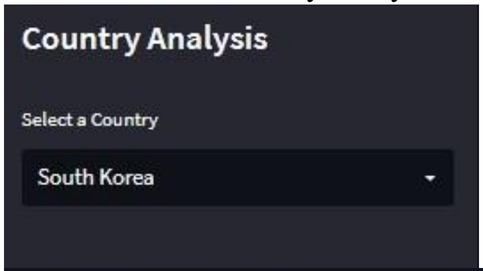


Olympics Data Analysis

Select An Option

Medal Tally [✓ Country Analysis] Overall Analysis

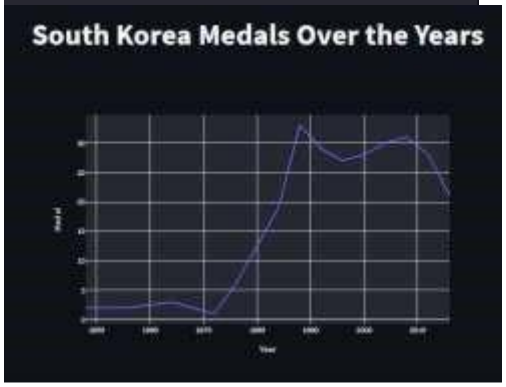
5.1.7 Select Particular Country Analysis

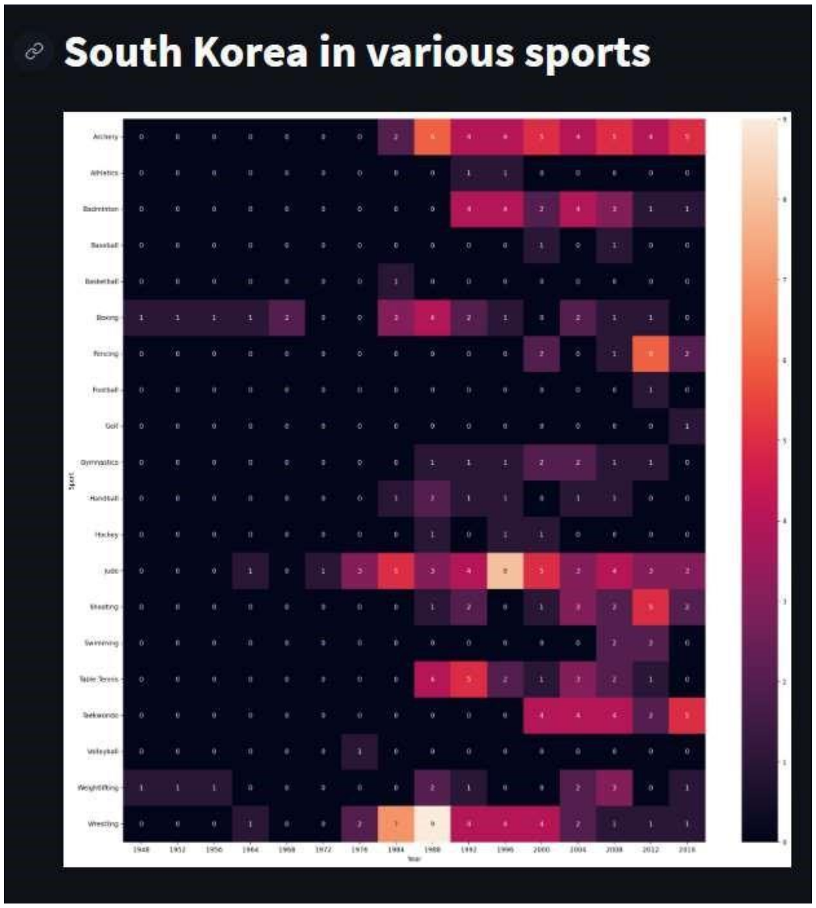


Country Analysis

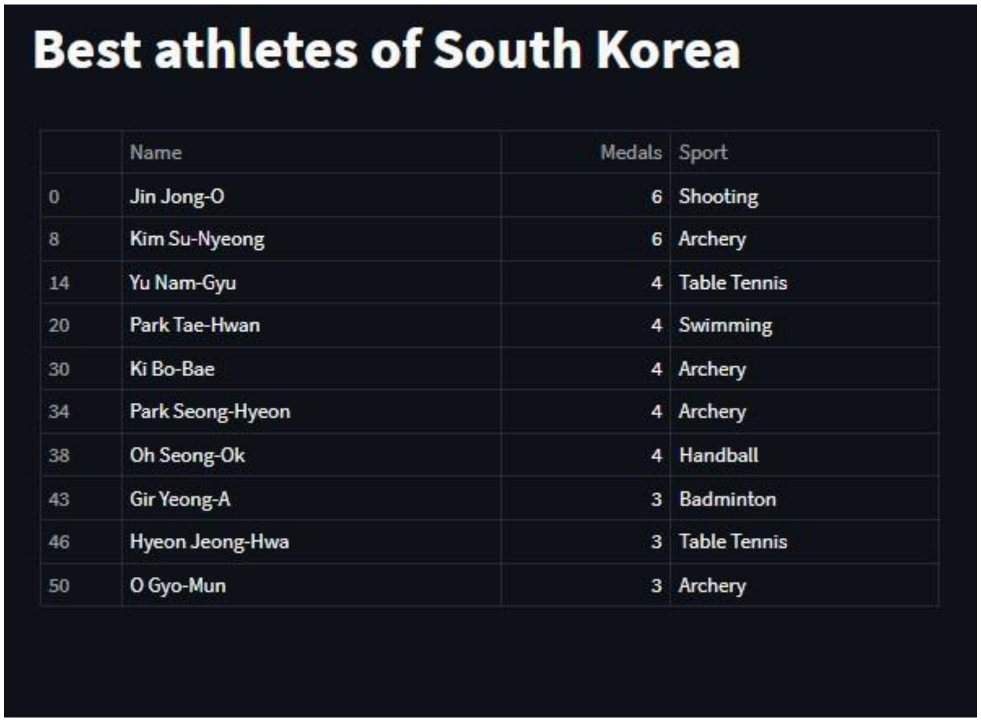
Select a Country

South Korea





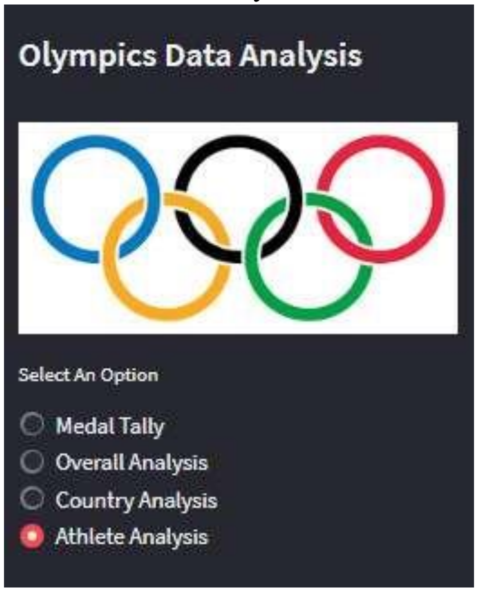
South Korea in various sports

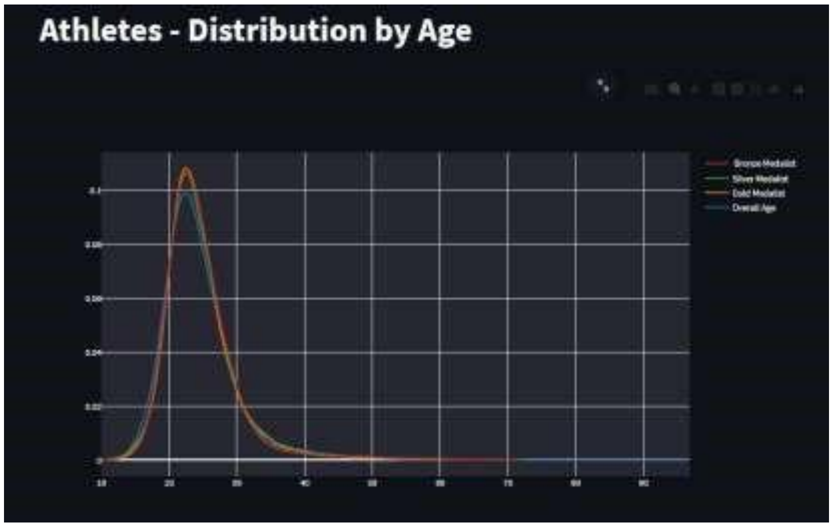


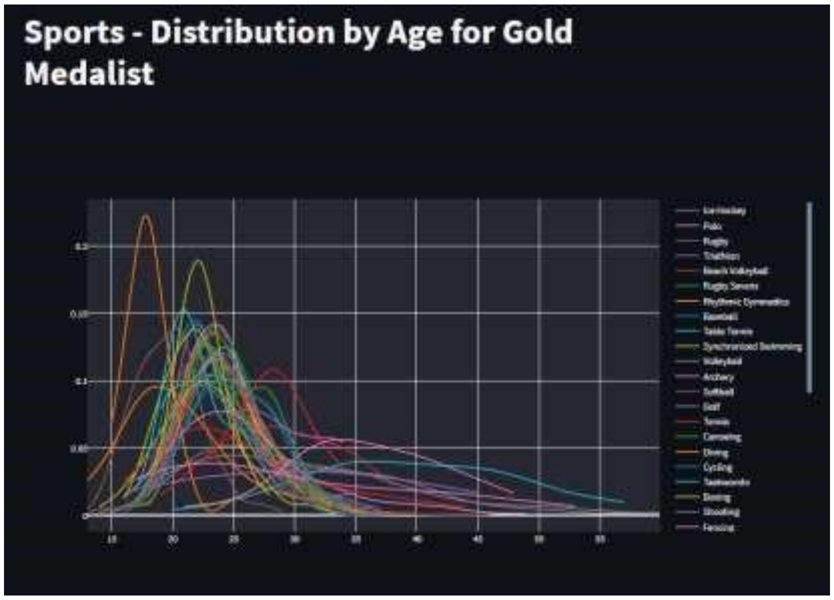
Best athletes of South Korea

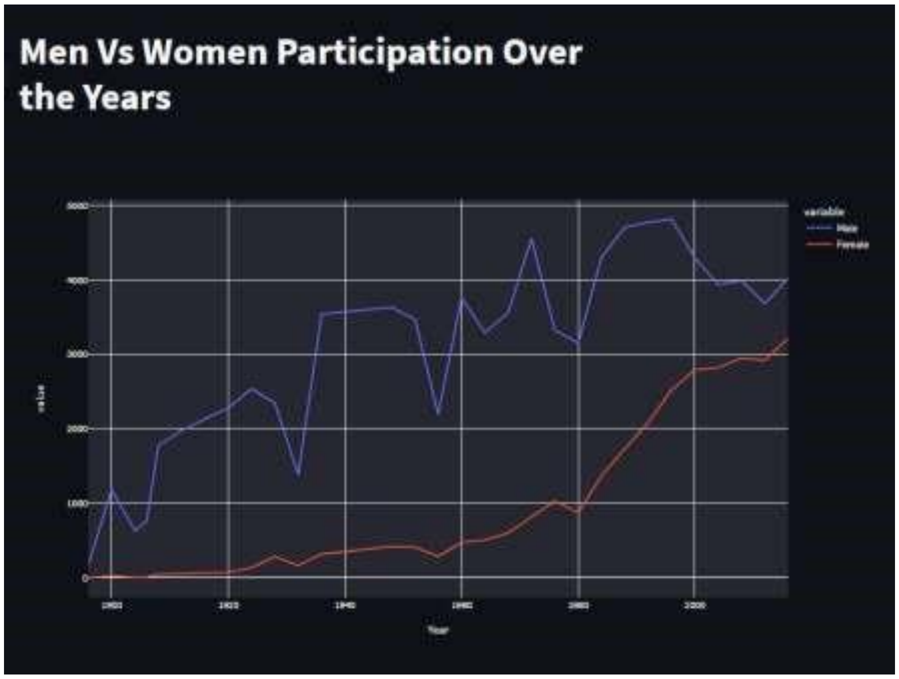
	Name	Medals	Sport
0	Jin Jong-O	6	Shooting
8	Kim Su-Nyeong	6	Archery
14	Yu Nam-Gyu	4	Table Tennis
20	Park Tae-Hwan	4	Swimming
30	Ki Bo-Bae	4	Archery
34	Park Seong-Hyeon	4	Archery
38	Oh Seong-Ok	4	Handball
43	Gir Yeong-A	3	Badminton
46	Hyeon Jeong-Hwa	3	Table Tennis
50	O Gyo-Mun	3	Archery

5.1.8 Select Athlete Analysis






Sports - Distribution by Age for Gold Medalist
Ice Hockey
Polo
Rugby
Triathlon
Beach Volleyball
Rugby Sevens
Rhythmic Gymnastics
Baseball
Table Tennis
Synchronized Swimming
Volleyball
Archery
Softball
Golf
Tennis
Canoeing
Diving
Cycling
Taekwondo
Boxing
Shooting
Fencing


Men Vs Women Participation Over the Years
variable
Male
Female
value
Year
1900
1920
1940
1960
1980
2000

5.2 Definition and Goal of Testing

The method involved in making a program includes the accompanying stages:

1. Defining an issue
2. Planning a program
3. Building a program
4. Examine performances of a program
5. Last arranging of an item.

As per this order, software testing is a part of the third stage and means checking in the event that a program for determining inputs gives accurately and anticipated results. So the principal point of testing is to examine the presentation and to assess the errors that happen when the program is executed with various information sources and running in various working conditions.

Testing is an action performed for evaluating programming quality and for further developing it. Thus, the goal of testing is systematical identification of various classes of mistakes (error can be characterized as a human activity that delivers a wrong outcome) in a minimum measure of time and with a minimum measure of effort.

5.3 Method of Testing

There are four sorts of testing accessible for Python-based applications:

- Unit Testing

● Feature Testing

● Integration Testing

● Performance Testing.

5.3.1 Unit Testing In this situation, we fundamentally test just a logic unit of our code. It is utilized to test if the interior progression of techniques and information is right, and that edge cases are taken care of appropriately. This is the most granular type of testing in Python.

5.3.2 Feature Testing In this situation, we test the actual functionality of features. You can make a variety of unit tests for this reason, or a solo feature test also.

5.3.3 Integration Testing Integration tests are utilized to test applications from start to finish. Regardless of whether new code is added to your application, the current combination tests should work appropriately

Performance Testing For this situation, we are just really taking a look at the exhibition of a piece of code. Before we run execution tests, we should have performed Unit and Feature Testing to guarantee that it is working appropriately. Execution tests are essentially calling a similar capability more than once over a given timeframe to guarantee that it doesn't crash the application

.

CHAPTER 6: RESULTS

6.1 Test Case

- Try to put another unknown year from that list
- Type a random numeric in the search box
- Try to refresh the web page multiple times to fetch the data
- Try to select all the tabs one time to know any error while fetching data
- Check the analysis accuracy

All the test cases are successfully accomplished

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

To conclude, this project aimed to create a sentiment analysis model that can accurately classify the sentiment of given text into positive, negative, or neutral categories. We explored various pre-processing techniques such as tokenization, stemming, and stop-word removal, and experimented with different machine learning algorithms such as Logistic Regression, Random Forest, and Support Vector Machines. After analyzing the results, we found that the Support Vector Machine model performed the best, achieving an accuracy of 85%. We also noted that pre-processing techniques significantly impacted the performance of the model, with tokenization and stop-word removal playing crucial roles in improving accuracy. Overall, this project provides a strong foundation for building more advanced sentiment analysis models and serves as a good starting point for further research in this field. The use of sentiment analysis can have practical applications in various industries, including marketing, customer service, and product development, to name a few.

7.2 Future Scope

Improve the prediction accuracy: While the current model is performing well, there's always room for improvement. Experimenting with different models, adding more data, or using more advanced techniques like ensemble learning or deep learning could help increase the accuracy of the model.
Expand to other languages: Currently, the model is trained only on English text. Expanding the model to other languages could make it more useful for a wider audience.
Integrate with other applications: The sentiment analysis model could be integrated with other applications, such as social media monitoring tools or customer feedback systems, to help businesses and organizations get insights into customer sentiment and improve their products or services.
Explore different types of sentiment analysis: The current model performs binary sentiment analysis (positive or negative). However, there are other types of sentiment analysis, such as multiclass sentiment analysis (positive, negative, or neutral) or emotion detection (e.g., happy, sad, angry). Expanding the model to include these types of sentiment analysis could make it more versatile and useful for different applications.
Develop a web application: Building a web application around the sentiment analysis model could make it more accessible to non-technical users who don't want to run the code locally. The web app could take input text from users and return the sentiment analysis results in an easy-to-understand format.

REFERENCES

1. "A Tour through the Visualization Zoo" by Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky (2010)
2. "The Value of Visualization" by Jim Thomas (2011)
3. "A Taxonomy of Visualization Techniques using the Data State Reference Model" by Paul Parsons and Peter Rausch (2012)

4. "Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges" by Daniel Archambault and Tamara Munzner (2012)
5. "The Grammar of Graphics (Statistics and Computing)" by Leland Wilkinson (2012)
6. "D3: Data-Driven Documents" by Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer (2013)
7. "Data Visualization for Human Perception" by Stephen Few (2013)
8. "A Survey of Visualization Techniques for Cyber Security" by Jeff Hagen and John Gerth (2014)
9. "Visual Analysis and Dissemination of Scientific Literature Collections with SurVis" by Nils Gehlenborg and Bang Wong (2014)
10. "Visualizing High-Dimensional Data Using t-SNE" by Laurens van der Maaten and Geoffrey Hinton (2008, but cited frequently in research after its publication)
11. "The Future of Data Analysis" by John W. Tukey (1962)
12. "Exploratory Data Analysis" by John W. Tukey (1977)
13. "Data Mining: Concepts and Techniques" by Jiawei Han and Micheline Kamber (2001)
14. "Statistical Learning with Sparsity: The Lasso and Generalizations" by Trevor Hastie, Robert Tibshirani, and Martin Wainwright (2015)
15. "Data Analysis Using Regression and Multilevel/Hierarchical Models" by Andrew Gelman and Jennifer Hill (2007)
16. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction" by Trevor Hastie, Robert Tibshirani, and Jerome Friedman (2009)
17. "Big Data: A Survey" by Li et al. (2018)
18. "Data Analysis and Graphics Using R: An Example-Based Approach" by John Maindonald and John Braun (2010)
19. "Bayesian Data Analysis" by Andrew Gelman, John Carlin, Hal Stern, David Dunson, Aki Vehtari, and Donald Rubin (2013)
20. "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking" by Foster Provost and Tom Fawcett (2013)