Explanation of Local Sports Analysis Code

Loading Dataset

```
import pandas as pd

# Load dataset
file_path = '/content/Local_sports_dataset.csv
df = pd.read_csv(file_path)
df
```

1. `import pandas as pd`: Imports the pandas library, used for data manipulation and analysis.
2. `file_path`: Specifies the file path to the dataset.
3. `pd.read_csv(file_path)`: Reads the CSV file into a pandas DataFrame named `df`.
4. `df`: Displays the loaded dataset.

Analyze Participants by Age

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

# Load the data
data = pd.read_csv('/content/Local_sports_dataset.csv')

# Participation Analysis: Compare participation across different sports and age groups
participation = data.groupby(['Sport', 'Participant Age Group']).size().unstack()
participation.plot(kind='bar', stacked=True, figsize=(10, 6), cmap='viridis')
plt.title('Participation Across Sports and Age Groups')
plt.xlabel('Sport')
plt.ylabel('Number of Participants')
plt.legend(title='Age Group', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

1. `import matplotlib.pyplot as plt`: Imports Matplotlib for data visualization.
2. `import seaborn as sns`: Imports Seaborn for enhanced data visualization.
3. `data = pd.read_csv(...)`: Loads the dataset into the `data` DataFrame.
4. `data.groupby(['Sport', 'Participant Age Group']).size()`: Groups data by 'Sport' and 'Participant Age Group', counting the occurrences.

5. `.unstack()`: Transforms the grouped data to have 'Age Group' as columns.
6. `participation.plot(...)`: Plots a stacked bar chart.
7. `plt.title`, `plt.xlabel`, `plt.ylabel`: Adds title and axis labels to the chart.
8. `plt.legend`: Adds a legend outside the plot.
9. `plt.show()`: Displays the chart.

Audience Preferences

```
audience_preferences = data['Audience Favourite Sport'].value_counts()
audience_preferences.plot(kind='pie', figsize=(8, 8), autopct='%1.1f%%', colors=['blue',
'cyan', 'red', 'lime', 'orange', 'pink'])
plt.title('Most Popular Sports Among Attendees')
plt.tight_layout()
plt.show()
```

1. `data['Audience Favourite Sport'].value_counts()`: Counts the occurrences of each favorite sport.
2. `.plot(kind='pie', ...)`: Plots a pie chart.
3. `autopct='%1.1f%%'`: Adds percentages to pie slices.
4. `plt.title`: Sets the chart title.
5. `plt.tight_layout()`: Adjusts the layout for better visualization.
6. `plt.show()`: Displays the chart.

Event Popularity Metrics

```
# Convert 'Tickets Sold' to numeric for aggregation
data['Tickets Sold'] = pd.to_numeric(data['Tickets Sold'], errors='coerce')

# Group by 'Sport' and sum 'Tickets Sold'
ticket_sales = data.groupby('Sport')['Tickets Sold'].sum()

plt.figure(figsize=(10, 6))
ticket_sales.plot(kind='line', marker='o', color='red', linestyle='-')
plt.title('Total Tickets Sold by Sport')
plt.xlabel('Sport')
plt.ylabel('Total Tickets Sold')
plt.grid(True)
plt.xticks(ticks=range(len(ticket_sales)), labels=ticket_sales.index, rotation=45)
plt.tight_layout()
plt.show()
```

1. `pd.to_numeric(..., errors='coerce')`: Converts 'Tickets Sold' to numeric values, replacing invalid entries with NaN.
2. `.groupby('Sport')['Tickets Sold'].sum()`: Groups data by 'Sport' and calculates the total tickets sold.
3. `plt.figure(figsize=(10, 6))`: Creates a figure with specified dimensions.
4. `.plot(kind='line', marker='o', ...)`: Plots a line chart with markers.
5. `plt.grid(True)`: Adds a grid to the chart.
6. `plt.xticks(...)`: Rotates x-axis labels for better readability.

Performance Trends

```
top_performers = data['Top Scorer'].value_counts().head(10)
top_performers.plot(kind='barh', figsize=(10, 6), color='lime')
plt.title('Top Performers (Most Scores)')
plt.xlabel('Number of Scores')
plt.ylabel('Top Scorer')
plt.tight_layout()
plt.show()
```

1. `data['Top Scorer'].value_counts().head(10)`: Counts occurrences of top scorers, selecting the top 10.
2. `.plot(kind='barh', ...)`: Plots a horizontal bar chart.
3. `plt.tight_layout()`: Adjusts layout for better visualization.
4. `plt.show()`: Displays the chart.

Winning Streaks

```
winning_streaks = data['Event Winner'].value_counts().head(10)
plt.figure(figsize=(10, 6))
plt.scatter(winning_streaks.index, winning_streaks.values, color='gold', s=100)
plt.title('Winning Streaks (Most Wins by Team)')
plt.xlabel('Event Winner')
plt.ylabel('Number of Wins')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

1. `data['Event Winner'].value_counts().head(10)`: Counts occurrences of winning teams, selecting the top 10.
2. `plt.scatter(..., s=100)`: Creates a scatter plot with specified marker size.

3. `plt.xticks(rotation=45)`: Rotates x-axis labels for better readability.
4. `plt.tight_layout()`: Adjusts layout for better visualization.
5. `plt.show()`: Displays the chart.