CW2-PORTFOLIO
PRINCIPLES OF DATA SCIENCE

MSc in Data Science, Coventry University, UK

B M J N Balasuriya
Index No: COMScDS242P-009

Link to GitHub Public Repository:

https://github.com/j2damax/principles-of-ds-cw02-portfolio.git

Question 1:

Tuk Tuk Tournament - Adventure Across Sri Lanka

The Tuk Tuk Tournament is a fun and exciting two-week event in Sri Lanka, hosted by Tuk Tuk Rental. Teams of 2 or 3 travelers sign up each year to explore the country in a three-wheeled Tuk Tuk, completing challenges along the way. The goal is to experience Sri Lanka's culture like a local while promoting sustainable tourism. Each team earns points by completing challenges, which involve posting photos or videos as proof. The team with the highest points wins! This unique adventure blends independent travel, teamwork, and competition, making it a one-of-a-kind way to see Sri Lanka!

How the Tuk Tuk Tournament Works:

1. Sign Up & Form a Team
   - Travelers sign up for the event in teams of 2 or 3 members.
2. Get Your Tuk Tuk & Tools
   - Each team gets a self-driven Tuk Tuk and access to maps, rules, and a mobile app.
3. Complete Challenges & Earn Points
   - Teams explore Sri Lanka while completing challenges in two ways:
     - Location-based challenges (specific places to visit).
     - Questbook challenges (tasks that involve cultural experiences, activities, or adventures).
   - Teams must upload a photo or video as proof of completing each challenge.
4. Compete for the Highest Score
   - Each challenge has a pre-defined point value.
   - The team with the most points at the end of the event wins the competition.
5. Experience the Culture & Adventure
   - The event encourages sustainable tourism and helps travelers experience Sri Lankan life, food, nature, and traditions in a unique way.
6. Celebrate & Share Stories
   - There are opening, mid-event, and final parties where teams share experiences, celebrate, and enjoy the journey together!

1. Data Collection:

Original data set is hosted in a Airtable base downloaded via a Airtable Rest API using a python code.

Airtable Rest API Request:

```
curl "https://api.airtable.com/v0/apprqf6ijdtKiSNF3/Score?
maxRecords=3&view=Photo%20of%20the%20Day" \
  -H "Authorization: Bearer YOUR_SECRET_API_TOKEN"
```

Airtable Rest API JSON Response:


```
{
    "id": "reckSQkp7lTsX5vBE",
    "createdTime": "2024-11-01T10:21:14.000Z",
    "fields": {
        "challenge_type": "questbook",
        "points": 3,
        "media_type": "photo",
        "media_url": "https://ik.imagekit.io/kkflp95sa/2024/TukNRoll/F5C71361-FA02-452B-B411-
47E1A726F96C_wyufusIcf.jpg",
        "media_latitude": 6.010301719577316,
        "media_longitude": 80.24328140297976,
        "media_timestamp": "2024-11-01T10:20:39.627Z",
        "questbook": [
            "rec66VSyR48ioGLzT"
        ],
        "team": [
            "rec1472pzRMvQzkcg"
        ],
        "caption": "Highlight of Cynthia's trip",
        "DeviceType": "iOS",
        "AppVersion": "2.9.0",
        "initiated": "2024-11-01T10:21:09.313Z",
        "_id_": "reckSQkp7lTsX5vBE",
        "questbook_name": [
            "EPIC PHOTO"
        ],
        "questbook_category": [
            "reczlU3vLr8Yt1tN8"
        ],
        "questbook_type": [
            "Artifact"
        ],
        "team_profile_image": [
            "https://ik.imagekit.io/kkflp95sa/2024/B8A8D0D5-7DCD-46D4-A379-46C5D09B382F_9RTffsgnE.jpeg"
        ],
        "updatedAt": "2024-11-01T10:21:14.000Z",
        "team_name_clean_automation": "TukNRoll",
        "day": "2024-11-01",
        "media_long_lat_formular": "6.010301719577316,80.24328140297976",
        "comments_count": 0,
        "created": "2024-11-01T10:21:14.000Z",
```

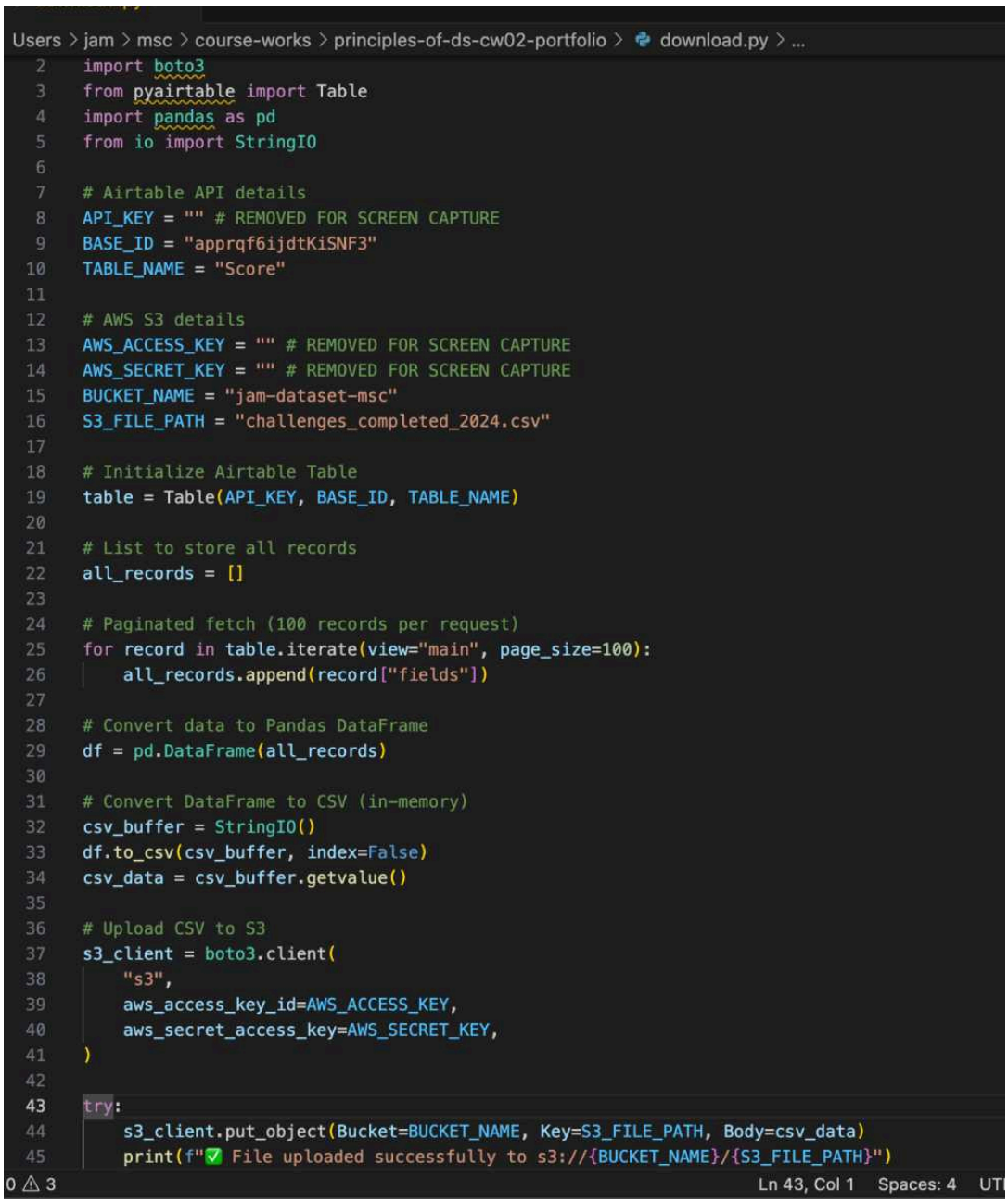
Data Retrieval:

Download data using Airtable's recommended python library pyairtable (https://github.com/gtalarico/pyairtable) using a python code and create a csv file.

Airtable REST API returns maximum of 100 records per response and paginated requests used to fetch all the records.

Once all records fetched from Airtable API, this create and upload a CSV file to AWS S3.

Filename: download.py



Connect Data:

Using Google Colab and Bot3, AWS S3 bucket can be access and access stored csv file for EDA.

Filename: tuk_tuk_tournament_eda.ipynb

```
#define s3 bucket name and file name
bucket_name = "jam-dataset-msc"
file_key = "challenges_completed_2024.csv"

# Access them in boto3
s3 = boto3.client(
    "s3",
    aws_access_key_id=os.getenv("AWS_ACCESS_KEY_ID"),
    aws_secret_access_key=os.getenv("AWS_SECRET_ACCESS_KEY")
)
```

```
# Read the CSV file from S3
obj = s3.get_object(Bucket=bucket_name, Key=file_key)
df = pd.read_csv(obj["Body"])
df.head()
```

```
df.columns
```

```
Index(['_id_', 'team', 'team_name', 'team_name_clean_automation',
       'team_profile_image', 'challenge_type', 'location', 'location_name',
       'location_zone', 'location_area', 'questbook', 'MaxPerDay', 'MaxPerTTT',
       'questbook_name', 'questbook_category', 'questbook_type', 'points',
       'seekers_portion', 'partner', 'partner_team_name', 'media_type',
       'media_name', 'media_url', 'caption', 'media_timestamp',
       'media_latitude', 'media_longitude', 'deleted', 'created', 'updatedAt',
       'comments', 'comment_status', 'report_comment', 'comment_team_name',
       'admin_comment', 'day', 'media_long_lat_formular', 'GeoLocation',
       'location_id', 'DeviceType', 'AppVersion', 'Voting of the Day',
       'URL_IS_VALID', 'IS_DUPLICATE', 'comments_count', 'initiated',
       'team_tokens', 'latitude', 'longitude', 'coordinates', 'AFTER_DEADLINE',
       'media_duplicated_automation_completed'],
      dtype='object')
```

```
df.shape
```

```
(17907, 52)
```

The dataset initially has 17,907 rows and 52 columns. When saving the csv file, ignored the unwanted columns and created a data frame with 17907 rows and 28 columns.

2. Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA): Understand patterns, trends, and distributions in the dataset.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17907 entries, 0 to 17906
Data columns (total 28 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   team_name           17907 non-null  object
 1   challenge_type      17907 non-null  object
 2   location_name       5405 non-null   object
 3   location_zone       5405 non-null   float64
 4   location_area       5405 non-null   object
 5   questbook           12502 non-null  object
 6   MaxPerDay           12502 non-null  float64
 7   MaxPerTTT           12502 non-null  float64
 8   questbook_name      12502 non-null  object
 9   questbook_category  12502 non-null  object
 10  questbook_type      12502 non-null  object
 11  points              17907 non-null  int64
 12  seekers_portion     996 non-null    object
 13  partner_team_name   769 non-null    object
 14  media_type          17907 non-null  object
 15  caption             9915 non-null   object
 16  deleted             248 non-null    object
 17  created             17907 non-null  object
 18  updatedAt           17907 non-null  object
 19  comment_status      523 non-null    object
 20  report_comment      523 non-null    object
 21  comment_team_name   523 non-null    object
 22  admin_comment       516 non-null    object
 23  day                 17907 non-null  object
 24  comments_count      17907 non-null  int64
 25  initiated           17907 non-null  object
 26  latitude            5405 non-null   float64
 27  longitude           5405 non-null   float64
dtypes: float64(5), int64(2), object(21)
memory usage: 3.8+ MB
```

- **Unwanted Columns**: Identified and removed irrelevant columns before EDA.
- **Missing Values**: Detected missing values; imputed them based on data type (mean/median for numerical, mode for categorical).
- **Data Type Corrections**: Converted data types as required:
  - Categorical columns → Converted to category.
  - Date columns → Converted to datetime.
  - Numerical columns stored as strings → Converted to appropriate numeric format.
- **Duplicates**: Checked for and removed duplicate rows.
- **Outliers**: Identified using boxplots and handled appropriately.

```
df.isnull().sum()
```

```
team_name                0
challenge_type           0
location_name        12502
location_zone        12502
location_area        12502
questbook             5405
MaxPerDay             5405
MaxPerTTT             5405
questbook_name        5405
questbook_category    5405
questbook_type        5405
points                   0
seekers_portion      16911
partner_team_name    17138
media_type               0
caption               7992
deleted              17659
created                  0
updatedAt                0
comment_status       17384
report_comment       17384
comment_team_name    17384
admin_comment        17391
day                      0
comments_count           0
initiated                0
latitude             12502
longitude            12502
dtype: int64
```

Based on the dataset, each data row does not necessarily contain all available information.

Mainly there are two challenge types:

1. Location based challenges
2. Quest book based challenges

When recording a Location-Based Challenge, only the relevant fields required to track its completion are captured. Similarly, when documenting a Quest Book Challenge, only the data specific to the quest book challenge is recorded.

```
# change NAN values to 0 and other values to 1
df["deleted"] = df["deleted"].notna().astype(int)
```

```
# It's safer to delete the deleted = 1 data set
print(df[df["deleted"] == 1].shape[0], "rows with deleted true")
print(df[df["deleted"] == 0].shape[0], "rows with deleted false")
```

```
248 rows with deleted true
17659 rows with deleted false
```

```
df.shape
```

```
(17907, 28)
```

```
df = df[df["deleted"] != 1]
```

```
df.shape
```

```
(17659, 28)
```

In this dataset, missing values (NaN) are interpreted as unset data. To standardize the "deleted" column, it is converted to an integer type, replacing NaN values with 0 and setting all other values to 1.

The dataset is filtered to retain only the records where the "deleted" column is not equal to 1, ensuring that only valid (non-deleted) entries are considered for further analysis.

After filtering, the data set contains 17659 rows.

```python
# fill missing values before conversion data types
df['location_zone'] = df['location_zone'].fillna(0).astype(int)
df['MaxPerDay'] = df['MaxPerDay'].fillna(0).astype(int)
df['MaxPerTTT'] = df['MaxPerTTT'].fillna(0).astype(int)

df.dtypes
```

```
team_name              object
challenge_type         object
location_name          object
location_zone           int64
location_area          object
questbook              object
MaxPerDay               int64
MaxPerTTT               int64
questbook_name         object
questbook_category     object
questbook_type         object
points                  int64
seekers_portion        object
partner_team_name      object
media_type             object
caption                object
deleted                 int64
created                object
updatedAt              object
comment_status         object
report_comment         object
comment_team_name      object
admin_comment          object
day                    object
comments_count          int64
initiated              object
latitude              float64
longitude             float64
dtype: object
```

Missing values in specific columns (location_zone, MaxPerDay, and MaxPerTTT) are filled with 0 before converting their data types to integers. This ensures consistency and avoids errors when performing further analysis.

```
# convert initiated, created, updatedAt columns to date time type
```

```
df['initiated'] = pd.to_datetime(df['initiated'], format='%d/%m/%Y %I:%M%p')
df['created'] = pd.to_datetime(df['created'], format='%d/%m/%Y %I:%M%p')
df['updatedAt'] = pd.to_datetime(df['updatedAt'], format='%d/%m/%Y %I:%M%p')
df.dtypes
```

```
team_name                    object
challenge_type               object
location_name                object
location_zone                 int64
location_area                object
questbook                    object
MaxPerDay                     int64
MaxPerTTT                     int64
questbook_name               object
questbook_category           object
questbook_type               object
points                        int64
seekers_portion              object
partner_team_name            object
media_type                   object
caption                      object
deleted                       int64
created              datetime64[ns]
updatedAt            datetime64[ns]
comment_status               object
report_comment               object
comment_team_name            object
admin_comment                object
day                          object
comments_count                int64
initiated            datetime64[ns]
latitude                    float64
longitude                   float64
dtype: object
```

The columns initiated, created, and updatedAt are converted from object (string) format to datetime64 format. This ensures proper handling of date and time values for accurate time-based analysis.

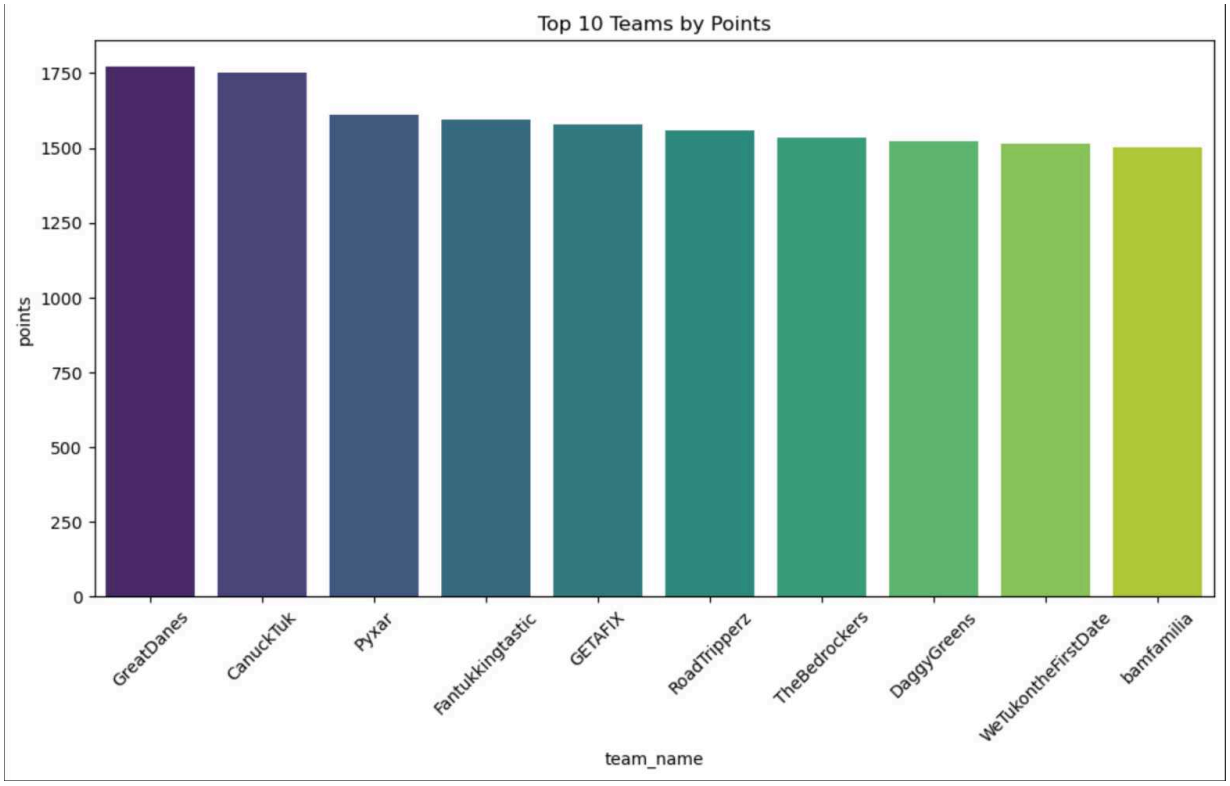
Descriptive statistics and visualizations to uncover patterns or anomalies

```
# top 10 teams
top_teams = df.groupby('team_name')['points'].sum().reset_index()
.sort_values(by='points', ascending=False)
print("Top 10 Teams:\n", top_teams.head(10))
```

```
Top 10 Teams:
              team_name  points
19          GreatDanes    1772
8            CanuckTuk    1752
34               Pyxar    1610
15    Fantukkingtastic    1593
17             GETAFIX    1577
39        RoadTripperz    1559
47       TheBedrockers    1533
11         DaggyGreens    1523
70  WeTukontheFirstDate    1514
75           bamfamilia    1503
```

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

plt.figure(figsize=(12,6))
sns.barplot(x=top_teams.head(10)['team_name'], y=top_teams.head(10)['points'], palette='viridis')
plt.xticks(rotation=45)
plt.title("Top 10 Teams by Points")
plt.show()
```



Visualise top 10 performing teams.

Top 10 teams visualisation:

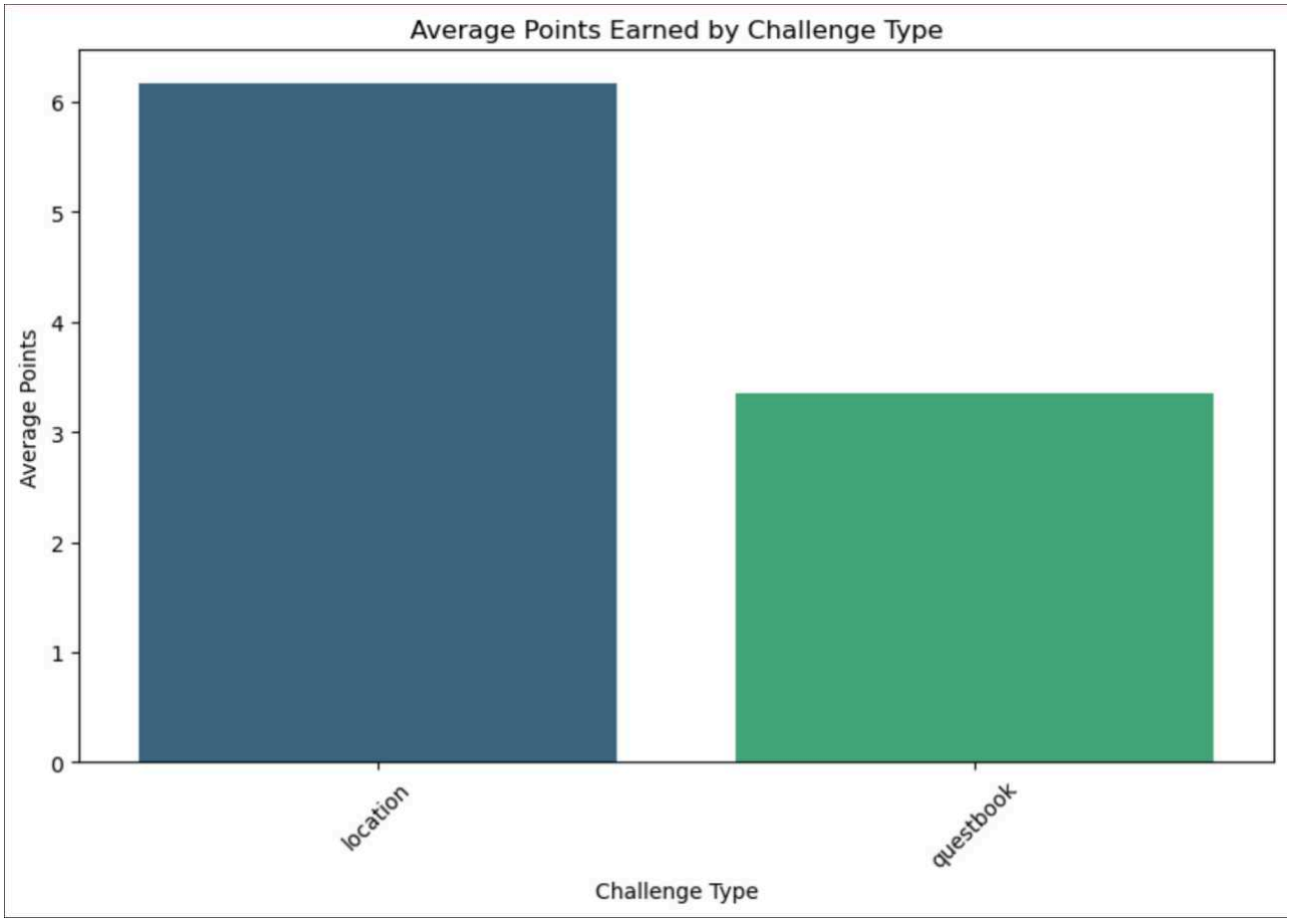
- The dataset was grouped by team_name to calculate the total points scored by each team.
- The highest-scoring team is GreatDanes, followed closely by CanuckTuk and Pyxar.
- The top 5 teams have relatively close scores, indicating strong competition.
- The score difference between GreatDanes (1772 points) and bamfamilia (1503 points, 10th place) is not significantly large.

Next let's find out do teams earn more points from location-based challenges or quest book challenges.

```
challenge_avg_points = df.groupby('challenge_type')['points'].mean().reset_index()

# Plotting the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x='challenge_type', y='points', data=challenge_avg_points, palette='viridis')

# Adding labels and title
plt.xlabel("Challenge Type")
plt.ylabel("Average Points")
plt.title("Average Points Earned by Challenge Type")
plt.xticks(rotation=45)
plt.show()
```



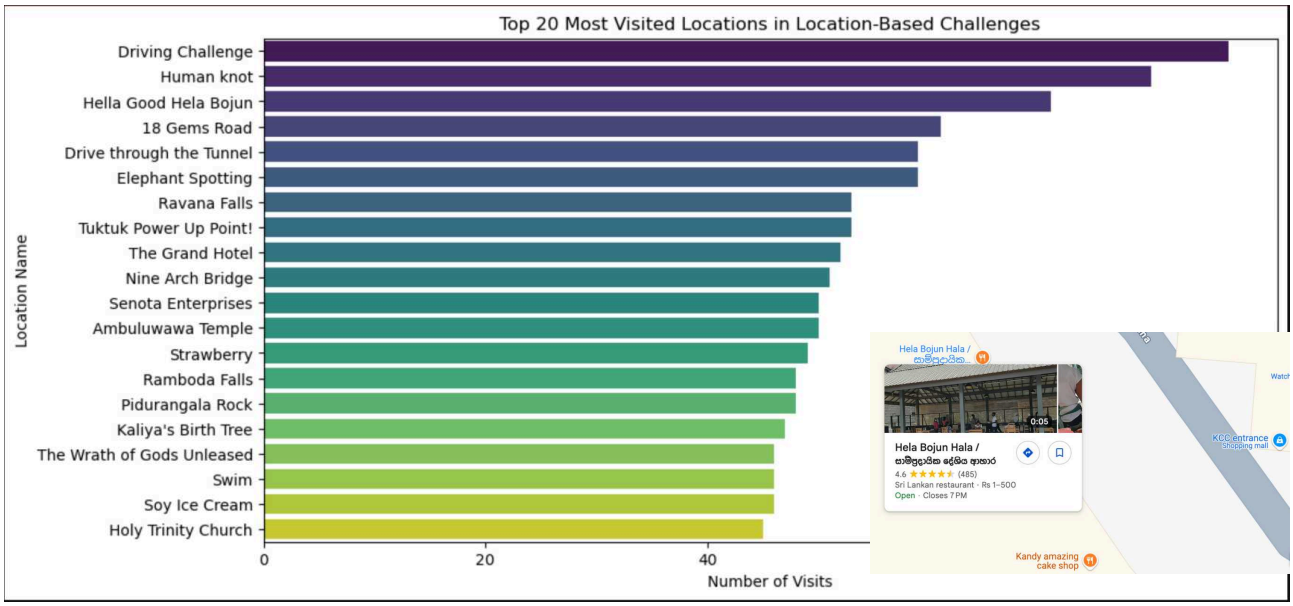
Teams has scored more points through completing location based challenges. This is interesting.

```python
# Filtering dataset for location-based challenges
location_challenges = df[df['challenge_type'] == 'location']

# Counting occurrences of each location
location_counts = location_challenges['location_name'].value_counts().reset_index()
location_counts.columns = ['location_name', 'count']

# Selecting top 20 most visited locations
top_locations = location_counts.head(20)

# Plotting bar chart for most visited locations
plt.figure(figsize=(12, 6))
sns.barplot(y=top_locations['location_name'], x=top_locations['count'], palette='viridis')
plt.xlabel("Number of Visits")
plt.ylabel("Location Name")
plt.title("Top 20 Most Visited Locations in Location-Based Challenges")
plt.show()
```



Key findings:

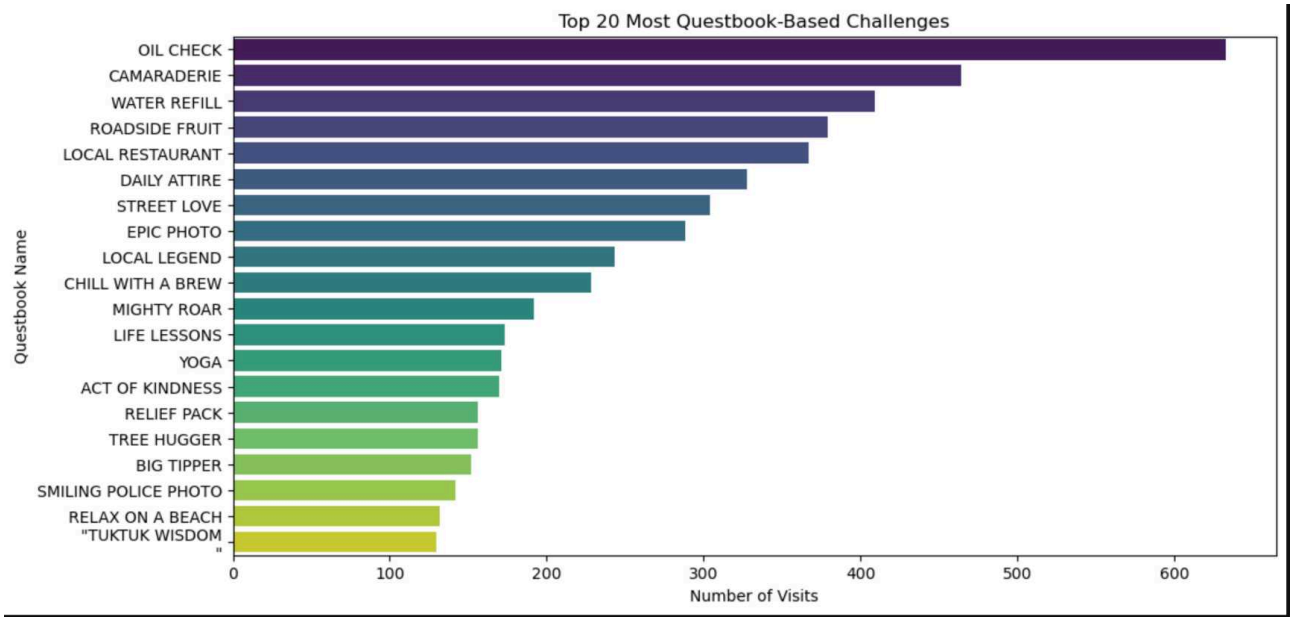
- The most visited location in location-based challenges is "Driving Challenge", followed by "Human Knot" and "Hella Good Hela Bojun". Found out that "Driving Challenge"and "Human Knot" are not really a location, it was an team building event has taken place in the very beginning of the tournament.
- Most significant location visited was "Hella Good Hela Bojun", this location is in Kandy with local food varieties in a beautiful location.
- These locations have significantly more visits than others, indicating their popularity among participants.
- The top locations include a mix of natural attractions (Ravana Falls, Nine Arch Bridge, Ramboda Falls, Pidurangala Rock) and cultural/experiential sites (Holy Trinity Church, Ambuluwa Temple, Tuktuk Power Up Point).

```
# Filtering dataset for location-based challenges
questbook_challenges = df[df['challenge_type'] == 'questbook']

# Counting occurrences of each location
questbook_counts = questbook_challenges['questbook'].value_counts().reset_index()
questbook_counts.columns = ['questbook', 'count']

# Selecting top 20 most visited locations
top_challenges = questbook_counts.head(20)

# Plotting bar chart for most visited locations
plt.figure(figsize=(12, 6))
sns.barplot(y=top_challenges['questbook'], x=top_challenges['count'], palette='viridis')
plt.xlabel("Number of Visits")
plt.ylabel("Questbook Name")
plt.title("Top 20 Most Questbook-Based Challenges")
plt.show()
```



Top 20 Quest book challenges completed.

- The most completed questbook-based challenge is "OIL CHECK", followed by "CAMARADERIE" and "WATER REFILL".
- Many challenges revolve around community engagement (e.g., "ACT OF KINDNESS", "RELIEF PACK") and cultural experiences (e.g., "LOCAL RESTAURANT", "LIFE LESSONS").
- The participation rate gradually decreases after the top two challenges, but most of the top 20 challenges still have high engagement. This could be due to that top three challenges are must do ones daily.

2. Problem Solving:

Find out the least popular location areas.

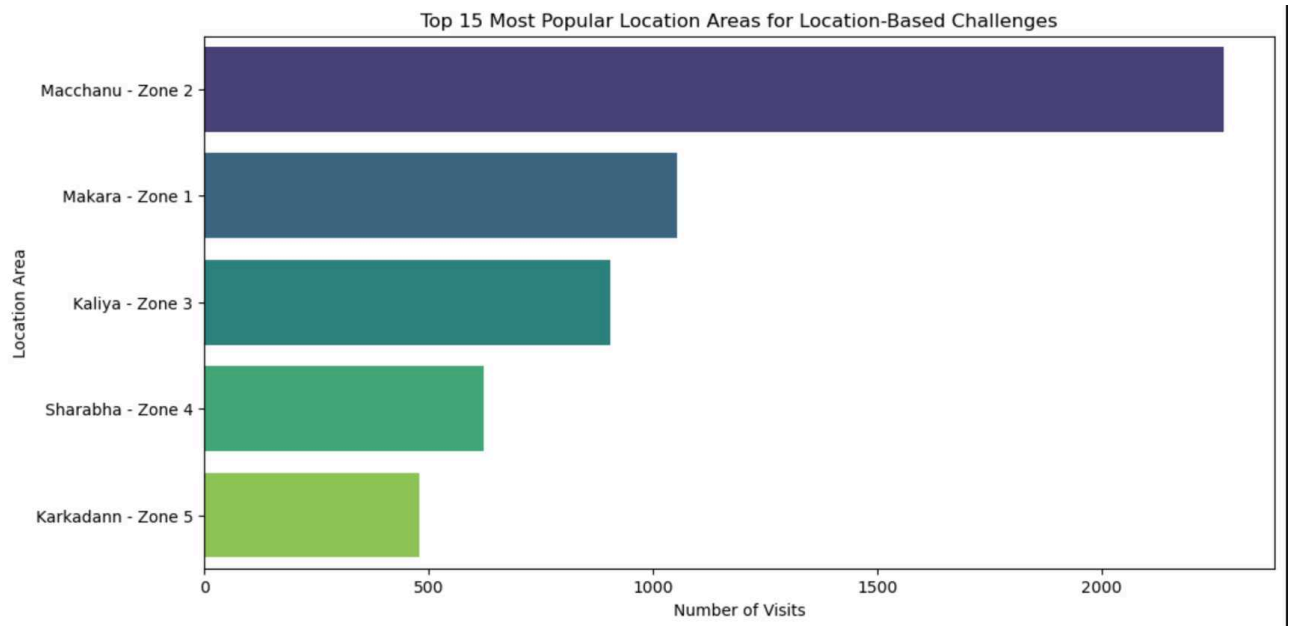
```
import matplotlib.pyplot as plt
import seaborn as sns

# Filtering dataset for location-based challenges
location_challenges = df[df['challenge_type'] == 'location']

# Counting occurrences of each location_area
location_area_counts = location_challenges['location_area'].value_counts().reset_index()
location_area_counts.columns = ['location_area', 'count']

# Selecting top 15 most popular location areas
top_location_areas = location_area_counts.head(15)

# Plotting bar chart for most popular location areas
plt.figure(figsize=(12, 6))
sns.barplot(y=top_location_areas['location_area'], x=top_location_areas['count'], palette='viridis')
plt.xlabel("Number of Visits")
plt.ylabel("Location Area")
plt.title("Top 15 Most Popular Location Areas for Location-Based Challenges")
plt.show()
```



Popular location areas might contain high-value challenges, well-known attractions, or be more accessible for participants.

In the dataset found out that Zone 2 which is the most popular area has 158 attractions and most least popular location Zone 5 has listed only 69 attractions.

Event organizers could find more attractions in these areas and promote them with higher points could balance the travel and promote around the country to attract more tourists.

Question 2:

Yahoo's Decline: From Internet Giant to a Struggling Player.

Yahoo was once a dominant force in the internet industry, leading in search, email, news, and digital advertising. However, despite its early success, the company struggled to adapt to changing market dynamics and ultimately lost its competitive edge. This report examines the key reasons for Yahoo's decline, supported by business statistics, industry trends, and strategic missteps.

Findings from the Internet:

- Founded in 1994 as a web directory, Yahoo became the gateway to the internet.
- By 2000, Yahoo was valued at over $125 billion and was the most-visited website globally.
- Revenue peaked due to advertising, Yahoo Mail, Yahoo Finance, and Yahoo News.
- Yahoo was a pioneer in display advertising, generating millions from banner ads.
- Failed Acquisition of Google (1998): Yahoo had the chance to buy Google for $1 million but rejected the offer.
- Rejected Microsoft's $44.6B Buyout (2008): Yahoo's refusal led to further market decline.
- Lost the Social Media Race: Failed acquisitions of Facebook (2006) and YouTube (2006).
- Tumblr Failure (2013): Acquired for $1.1B, but mismanagement led to a massive value drop.
- Rise of Google: Google dominated search (92% market share by 2020), leaving Yahoo behind.
- Over 7 CEOs between 2007-2017, leading to inconsistent strategies.
- 2013-2014 Data Breaches: Over 3 billion Yahoo accounts were compromised.

Dataset and Visualisation:

Companies market cap provide the market capitalisation of Yahoo Inc. Downloaded the dataset and saved in the repository and local drive for visualisation .

File name: yahoo_end_of_year_market_cap.csv

Notebook: yahoo.ipynb

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

```
import os
os.chdir("/Users/jam/msc/course-works")
print(os.getcwd())  # Verify the change
```

```
/Users/jam/msc/course-works
```

```
df = pd.read_csv("yahoo_end_of_year_market_cap.csv")
df.head()
```

	Year	Marketcap	Change
0	2017	50.38	36.55%
1	2016	36.89	17.57%
2	2015	31.38	-34.42%
3	2014	47.85	16.65%
4	2013	41.01	74.29%

```
df.dtypes
```

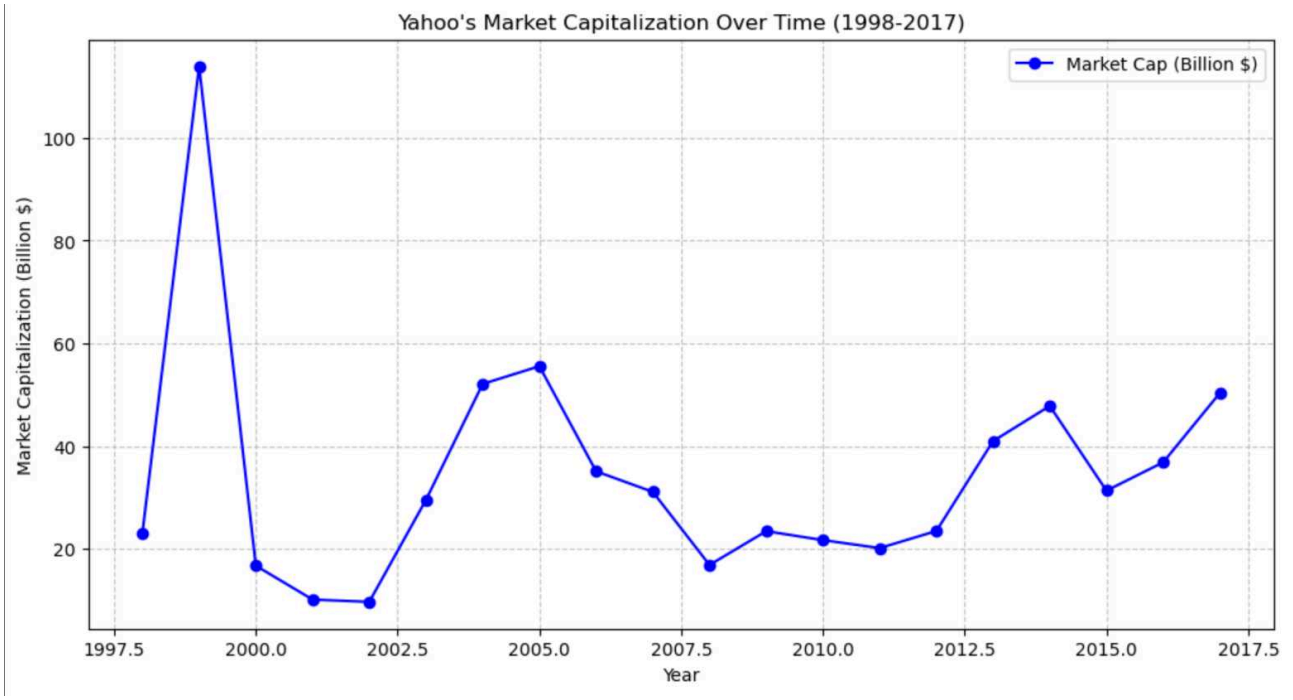
```
Year           int64
Marketcap    float64
Change        object
dtype: object
```

```python
# Sorting the data by Year in ascending order
df = df.sort_values(by="Year")

# Plotting Yahoo's Market Capitalization Over Time
plt.figure(figsize=(12, 6))
plt.plot(df["Year"], df["Marketcap"], marker='o', linestyle='-', color='b', label="Market Cap (Billion $)")

# Adding title and labels
plt.title("Yahoo's Market Capitalization Over Time (1998-2017)")
plt.xlabel("Year")
plt.ylabel("Market Capitalization (Billion $)")
plt.grid(True, linestyle="--", alpha=0.7)
plt.legend()

# Display the chart
plt.show()
```



Here is the bar chart visualizing Yahoo's market capitalization from 1998 to 2017.

Key Insights from the Chart:

- Yahoo peaked in 1999 (~$113.9 billion) during the dot-com boom before crashing.
- Drastic decline in 2000-2002, dropping below $10 billion after the dot-com bubble burst.
- Recovery in 2003-2006 (~$55 billion) as Yahoo expanded into advertising and media.
- Gradual decline from 2008-2015 as Google, Facebook, and YouTube overtook Yahoo.
- Sale to Verizon (2017) occurred after stabilizing around $50 billion.