AI Data Powered Analysis Early Internship - Sub-Group 7

Week-2 Final Report

Report Title: “Week-2: Exploratory Data Analysis, Insight Generation and Hypothesis Development Report ”

Associate Members:

Name	Email
Kolawole Oparinde	kolawole@vempower.org
Akanksha Choudhary	akanksha@vempower.org
Sakshi Sahu	sakshisahu@vempower.org
Shishir Jaiswal	shishir@vempower.org
Shruti Mishra	shrutimishra@vempower.org

Team Members:

Name	Role	Email
Sumaiya Tasnim	Team Lead	sumaiyaa.tasnim.18@gmail.com
Saniya Dantal	Project Scribe	dantalsb04@gmail.com
Zaheer 123	Project Manager	mianzaheer4195@gmail.com
Devadharshini T	Project Lead	dharshinideva83@gmail.com
Peter Macharia	Team Member	mpeter778@gmail.com
Bokka Hamsini	Team Member	bokkahamsini@gmail.com

Introduction:

In the second week of the internship, we will focus on Exploratory Data Analysis (EDA), which is an essential step in understanding datasets before moving toward advanced AI-driven analytics. Through EDA, we will explore data to uncover patterns, identify anomalies, and summarize key characteristics using both statistical methods and visual techniques. The insights gained during this phase will form the foundation for building predictive models and driving meaningful AI-powered decisions in the coming weeks.

Objectives:

- Explore the dataset to understand its structure, key variables, and potential challenges.
- Perform statistical analysis to summarize data distributions, trends, and correlations.
- Apply visualization techniques to represent data clearly and identify hidden patterns.
- Generate preliminary insights and hypotheses for guiding advanced analysis.
- Compile findings into a structured EDA report that documents trends, anomalies, and recommendations.

Assigned Dataset : "SLU_Opportunity_Wise_Data.csv"
Cleaned Dataset was exported as : "Cleaned_Preprocessed_Dataset.csv"
Applied Features in dataset was renamed as : "Featured_Dataset.csv"

Tools We Will Use:

- **Excel / Google Sheets** – For basic data exploration, cleaning, and simple visualizations (if required).
- **Python** – The primary programming language for advanced data analysis and modeling.
- **Pandas** – For efficient data manipulation, cleaning, and exploration.
- **NumPy** – For numerical computations and handling arrays.
- **Matplotlib & Seaborn** – For creating clear and meaningful charts, graphs, and plots.
- **Plotly Express** – For interactive visualizations and dashboards.
- **SciPy** – For statistical analysis and hypothesis testing.
- **Jupyter Notebook / Google Colab** – Interactive environments for performing, documenting, and sharing analysis.
- **Warnings Library** – To manage and suppress unnecessary warning messages for cleaner outputs.

Expected Outcomes:

- ✓ Review and Verification of Week-1 Final Dataset.
- ✓ A comprehensive EDA report summarizing the dataset's structure, trends, and anomalies.
- ✓ Clear visualizations (charts, graphs, heatmaps, plots) that highlight key patterns.
- ✓ Identification of missing values, outliers, and inconsistencies for data cleaning.
- ✓ Hypotheses and preliminary insights that will guide subsequent AI-powered modeling.
- ✓ A dataset that is well-understood and ready for predictive analytics.

Learning Outcomes:

1. Developing a strong understanding of EDA principles and their role in data science.
2. Gaining hands-on experience in data exploration and visualization using Excel and Python.
3. Building proficiency in identifying trends, correlations, and anomalies within datasets.
4. Learning to generate insights and form hypotheses from raw data.
5. Improving skills in documenting and reporting findings effectively through an EDA report.

Review & Verification of Dataset

Dataset column's Review:

Column Name	Previous Data Cleaning	Latest Data Cleaning	Updates	Purpose / Reason
Learner SignUp DateTime	Filled missing values with most frequent date; corrected corrupted date formats; standardized datetime64	Filled missing values with the most frequent date in dataset; kept consistent datetime format "MM/DD/YYYY HH:MM:SS AM/PM"	Updated	To handle missing entries, ensure temporal consistency for feature engineering and analysis
Opportunity Id	Verified for duplicates	Already clean	Already Cleaned	Unique identifier verified, ensures integrity of records
Opportunity Name	Standardized categorical data, corrected capitalization	Already clean	Already Cleaned	Consistent naming for analysis and grouping
Opportunity Category	Standardized categorical data, corrected capitalization	Already clean	Already Cleaned	Uniform representation for statistical and visual analysis
Opportunity End Date	Fixed corrupted date formats, standardized datetime64, verified sequences	Filled missing values using median duration added to start date; ensured datetime consistency	Updated	To handle missing values and maintain accurate temporal calculations (duration, engagement)
First Name	Verified consistency	Already clean	Already Cleaned	Ensures correct identification without missing entries
Date of Birth	Verified formats, converted to datetime64	Already clean	Already Cleaned	Required for age calculation, verified consistency
Gender	Standardized categorical data, corrected variations	Already clean	Already Cleaned	Uniform gender representation for analysis
Country	Standardized categorical data, corrected typos and variations, harmonized values	Already clean	Already Cleaned	Uniform country representation to avoid misclassification
Institution Name	Corrected typos, expanded abbreviations, replaced mismatched entries, standardized text	Filled missing values with "Not Applicable"	Updated	To handle missing entries and ensure consistency in educational institution field
Current/Intended Major	Corrected typos, standardized text, harmonized categories	Filled missing values with "Other"; corrected misc/others categories	Updated	To handle missing entries and ensure consistent representation of majors
Entry created at	Standardized datetime64 format	Already clean	Already Cleaned	Ensures temporal consistency for analysis
Status Description	Standardized categorical data, corrected capitalization	Already clean	Already Cleaned	Uniform representation of status for analysis
Status Code	Checked numeric outliers, ensured valid range	Already clean	Already Cleaned	Numeric validation ensures consistency
Apply Date	Fixed corrupted date formats, standardized datetime64	Already clean	Already Cleaned	Required for engagement calculations and temporal analysis
Opportunity Start Date	Fixed corrupted date formats, standardized datetime64	Filled missing values per Opportunity Name median; ensured datetime consistency	Updated	To handle missing entries and maintain accurate temporal calculations for analysis

Features columns- Review:

Feature Column's Name	Applied Formula	Formula Representation	Explanation	Updates	Why Used / Benefit
Age	=DATEDIF(G2, TODAY(), "Y")	G2 → Date of Birth	Calculates learner's current age.	Same as previous	Helps analyze engagement across age groups; demographic insights.
Opportunity Duration	=E2 - P2	E2 → Opportunity End Date, P2 → Opportunity Start Date	Measures duration of each opportunity.	Same as previous	Useful for understanding if longer or shorter opportunities affect participation and satisfaction.
Normalized Age	=(Q2-MIN(Q$2:Q$100))/(MAX(Q$2:Q$100)-MIN(Q$2:Q$100))	Q2 → Age	Scales age to 0–1 range for comparability.	Updated (Newly Added Feature)	Ensures Age contributes fairly in models without dominating due to larger numeric scale.
Normalized Status Code	=(N2 - MIN(N$2:N$100)) / (MAX(N$2:N$100) - MIN(N$2:N$100))	N2 → Status Code	Converts status codes to 0–1 scale.	Same as previous	Enables fair comparison in analysis and prevents scale distortion.
Normalized Opportunity Duration	=IF(MAX(R$2:R$100)-MIN(R$2:R$100)=0, 0, (R2-MIN(R$2:R$100))/(MAX(R$2:R$100)-MIN(R$2:R$100)))	R2 → Opportunity Duration	Scales opportunity durations between 0–1.	Updated	Makes opportunity duration comparable and suitable for ML or statistical analysis.
Encoded Gender	=IF(H2="Male", 1, IF(H2="Female", 2, IF(H2="Prefer Not To Say", 3, IF(H2="Other", 4, ""))))	H2 → Gender	Converts gender categories to numeric codes.	Updated	Prepares gender for ML models and prevents inconsistencies with text data.
Encoded Opportunity Category	=IF(D2="Internship", 1, IF(D2="Course", 2, IF(D2="Event", 3, IF(D2="Competition", 4, IF(D2="Engagement", 5, "")))))	D2 → Opportunity Category	Converts opportunity types to numeric codes.	Updated	Enables predictive analysis of different opportunity types and their impact on engagement.
Encoded Country	=IF(I2="United States", 1, IF(I2="India", 2, IF(I2="Nigeria", 3, IF(I2="Ghana", 4, IF(I2="Pakistan", 5, IF(I2="Bangladesh", 6, IF(I2="Egypt", 7, IF(I2="Ethiopia", 8, IF(I2="Kenya", 9, IF(I2="Rwanda", 10, 11))))))))))	I2 → Country	Converts countries to numeric codes. Used Top 10 Countries in encoding.	Updated	Simplifies country analysis, enables grouping and numeric computation for models.
Extracted SignUp Month	=MONTH(A2)	A2 → Learner SignUp DateTime	Extracts the month number from the sign-up date.	Same as previous	Useful for analyzing trends, seasonality, and monthly patterns in engagement.
Extracted SignUp Year	=YEAR(A2)	A2 → Learner SignUp DateTime	Extracts the year from sign-up date.	Same as previous	Helps detect long-term trends and yearly growth in engagement.
Extracted SignUp Day	=DAY(A2)	A2 → Learner SignUp DateTime	Extracts the day of the month from sign-up date.	Same as previous	Allows detailed daily pattern analysis of engagement.

Feature Column's Name	Applied Formula	Formula Representation	Explanation	Updates	Why Used / Benefit
Weekly Patterns	=TEXT(A2, "ddd")	A2 → Learner SignUp DateTime	Shows abbreviated day of the week (Sun, Mon, Tue…).	Updated	Helps detect weekday vs weekend engagement patterns.
Seasonal Patterns	=IF(OR(Y2=12,Y2=1,Y2=2),"Winter", IF(OR(Y2=3,Y2=4,Y2=5),"Spring", IF(OR(Y2=6,Y2=7,Y2=8),"Summer", IF(OR(Y2=9,Y2=10,Y2=11),"Fall","""))))	Y2 → Extracted Month	Assigns season based on month number.	Same as previous	Detects seasonal trends, peaks, and troughs in engagement.
Engagement Days	=O2 - P2	O2 → Apply Date, P2 → Opportunity Start Date	Measures lag between application and opportunity start.	Same as previous	Identifies quick vs delayed engagement, uncovering friction or motivation factors.
Duration × Age	=Q2 * R2	Q2 → Age, R2 → Opportunity Duration	Multiplies age by opportunity duration to create an interaction metric.	Same as previous	Explores combined effects of learner age and opportunity length on engagement.
Engagement Score	=U2**0.4* + *Q2**0.2 + AD2**0.3* + *AE2**0.1	U2 → Normalized Opportunity Duration, Q2 → Age, AD2 → Engagement Days, AE2 → Duration × Age	Weighted composite score combining multiple engagement metrics.	Same as previous	Provides a holistic engagement measure combining duration, age, and interaction features.

Important Note: *We took out the ABS() function from the formulas and kept the original calculations, since ABS was distorting values and creating false outliers. Real outliers were then handled using* ***winsorization****, which pulls extreme values back into a reasonable range without removing any data. As a result, the dataset is now clean — with no nulls, NaNs, duplicates, inconsistencies, or negative values — and all rows remain intact.*

Note: *The dataset has undergone comprehensive cleaning and feature engineering. All missing or null values have been addressed, and inconsistencies or errors in the data have been resolved. The integrity of the dataset has been maintained, ensuring that no information has been altered incorrectly. With all features accurately prepared and validated, the dataset is now fully ready for Exploratory Data Analysis (EDA) to extract meaningful insights.*

Renamed the final Verified Dataset as : "Verified_Processed_Dataset.csv"
Total Records: 8246 rows
Total columns: 32

You can view the "Verified_Processed_Dataset.csv" at the following link:
https://drive.google.com/file/d/1es7BV21UQYphGL5-NuV1uG1TVxAHTNNp/view?usp=sharing

Exploratory Data Analysis

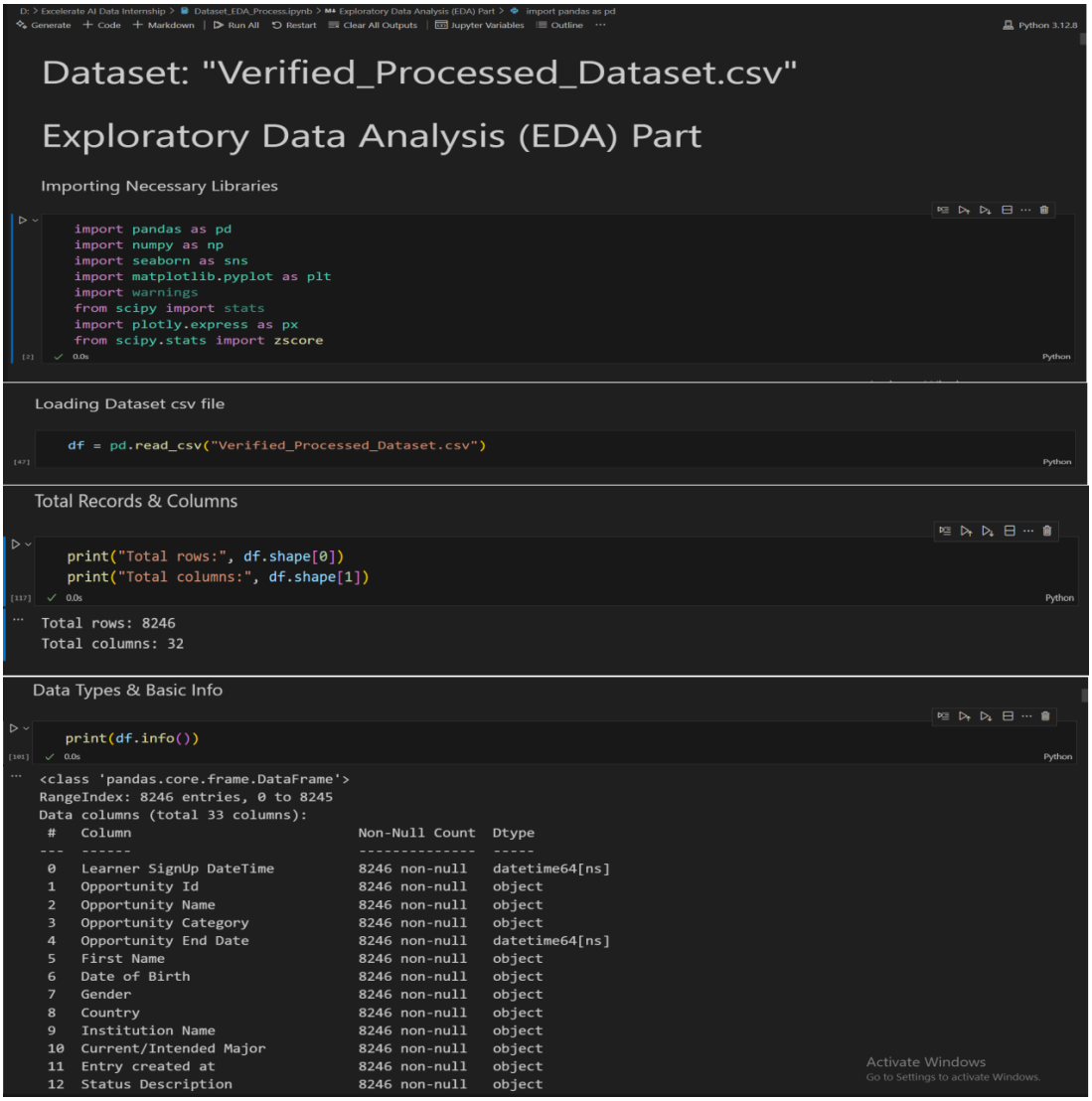
Exploratory Data Analysis (EDA) is a crucial process used to examine datasets, summarize their key characteristics, and identify patterns or irregularities through statistical measures and visualizations. It helps transform raw data into meaningful insights that guide further modeling and decision-making.

Benefits of EDA:

- ► Detects missing values, duplicates and outliers
- ► Provides understanding of data distributions
- ► Validates assumptions and highlights inconsistencies
- ► Uncovers hidden relationships between variables
- ► Ensures data quality for reliable analysis
- ► Improves accuracy and performance of predictive models

Performing EDA of our Dataset:

1) Data Understanding and Exploration:



```
 13  Status Code                     8246 non-null   int64
 14  Apply Date                      8246 non-null   object
 15  Opportunity Start Date          8246 non-null   datetime64[ns]
 16  Age                             8246 non-null   int64
 17  Opportunity Duration            8246 non-null   float64
 18  Normalized Age                  8246 non-null   float64
 19  Normalized Status Code          8246 non-null   float64
 20  Normalized Opportunity Duration 8246 non-null   int64
 21  Encoded Gender                  8246 non-null   int64
 22  Encoded Opportunity Category    8246 non-null   int64
 23  Encoded Country                 8246 non-null   int64
 24  Extracted SignUp month          8246 non-null   int64
 25  Extracted SignUp Year           8246 non-null   int64
 26  Extracted SignUp Day            8246 non-null   int64
 27  Weekly Patterns                 8246 non-null   object
 28  Seasonal Patterns               8246 non-null   object
 29  Engagement Days                 8246 non-null   float64
 30  Duration × Age                  8246 non-null   float64
 31  Engagement Score                8246 non-null   float64
```

Identifying missing (NaN / null) values

```python
# Show rows with missing values
missing_rows = df[df.isnull().any(axis=1)]
print(missing_rows)
```

[105] ✓ 0.0s Python

```
Empty DataFrame
Columns: [Learner SignUp DateTime, Opportunity Id, Opportunity Name, Opportunity Category, Opportunity End Date, First Name,
Index: []

[0 rows x 33 columns]
```

Checking Duplicates

```python
print("Duplicate rows:", df.duplicated().sum())
```

[107] ✓ 0.0s Python

```
Duplicate rows: 0
```

Checking Outliers

Outliers usually matter in numeric columns.

NEEDED: The most important for real outlier analysis will be- Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score.

NO NEEDED: For encoded categorical variables (Encoded Gender, Encoded Country, etc.), outlier detection usually isn't meaningful since they're discrete codes. Also textual / categorical / IDs / dates → won't have meaningful outliers. Examples: Opportunity Id, Opportunity Name, Country, Institution Name, First Name, Gender, Date of Birth, Opportunity Start Date, etc.

Generate + Code + Markdown

Add Markdown Cell

```python
# Select important numeric columns (exact names from your dataset)
cols_to_check = [                       # has trailing space
    "Opportunity Duration",
    "Normalized Age",
    "Normalized Opportunity Duration",
    "Duration × Age",                   # special multiplication sign
    "Engagement Score"
]

# Calculate IQR
Q1 = df[cols_to_check].quantile(0.25)
Q3 = df[cols_to_check].quantile(0.75)
IQR = Q3 - Q1

# Outlier condition
outliers = ((df[cols_to_check] < (Q1 - 1.5 * IQR)) |
            (df[cols_to_check] > (Q3 + 1.5 * IQR)))

# Outlier counts per column
print("Outlier counts per column (IQR method):")
print(outliers.sum())
```

[108] ✓ 0.0s Python

Activate Windows
Go to Settings to activate Windows.

```
Outlier counts per column (IQR method):
Opportunity Duration              0
Normalized Age                    0
Normalized Opportunity Duration   0
Duration × Age                    0
Engagement Score                  0
dtype: int64
```

Data Structure and Initial Checks summary:

At the start, the dataset contains 8,246 rows and 32 columns covering various data types including dates, categorical, and numerical fields.

- All columns have complete data with no missing values detected so far.
- Data types were reviewed using pandas info function:
    - Date columns such as 'Learner SignUp DateTime', 'Opportunity End Date', and others were converted into datetime format for consistency.
    - Categorical columns like 'Gender', 'Country', and 'Opportunity Category' are identified as object/string types.
    - Numeric columns include 'Age', 'Opportunity Duration', 'Engagement Score', and normalized or encoded versions of several features.

Data quality checks show:

✓ No missing/NaN/NULL values
✓ No duplicate rows
✓ No outliers in key numeric columns (Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score)
✓ No inconsistencies; all remaining data are valid and unchanged.

Overall, the dataset is clean and ready for analysis of patterns, engagement, and opportunity participation.

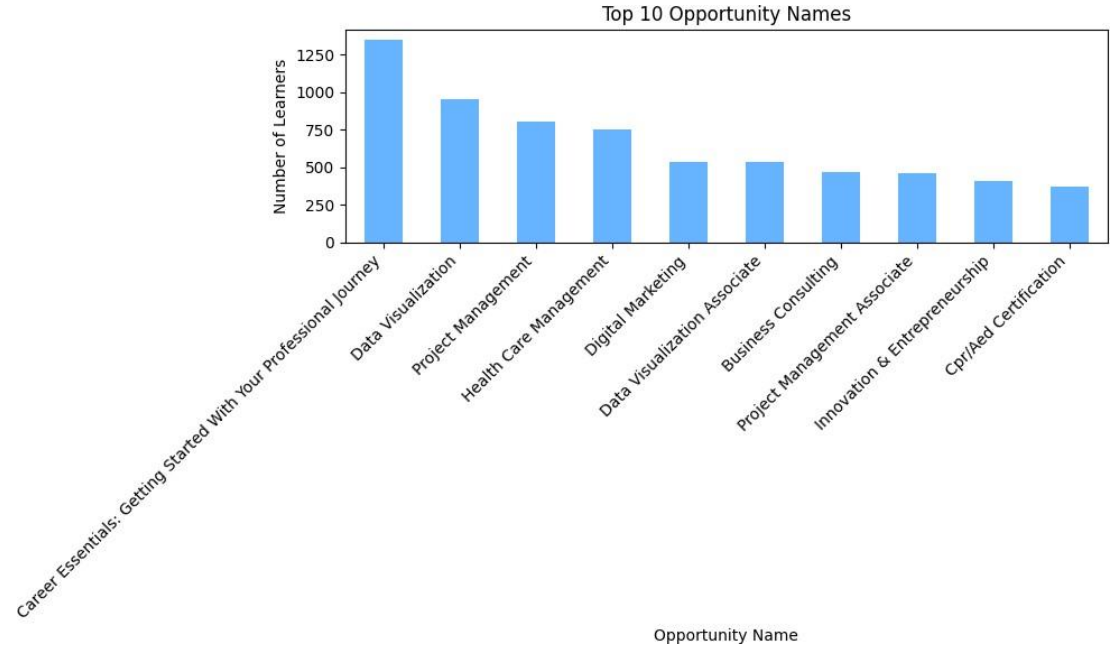
DATA CATEGORY ANALYSIS

Category	Columns
Identifiers	Opportunity Id, Opportunity Name, First Name
Dates	Learner SignUp DateTime, Opportunity End Date, Date of Birth, Entry created at, Apply Date, Opportunity Start Date
Numeric	Age, Opportunity Duration, Status Code, Normalized Age, Normalized Status Code, Normalized Opportunity Duration, Engagement Days, Duration × Age, Engagement Score
Categorical	Opportunity Category, Gender, Country, Institution Name, Current/Intended Major, Status Description
Encoded	Encoded Gender, Encoded Opportunity Category, Encoded Country
Seasonal Patterns	Weekly Patterns, Seasonal Patterns, Extracted SignUp Month, Extracted SignUp Year, Extracted SignUp Day

2) Highlighing Basic Key Insights:

Identifiers

(Visualized in Excel)



Aspect	Details
Figure	Opportunity ID by Learners (Bar Chart) Opportunity Names by Learners (Bar Chart) Top 10
Columns used	- Opportunity ID - Opportunity Name
Why used	- **Opportunity ID:** To visualize the number of learners per opportunity, identifying popular or less-attended courses, products, or events. - **Opportunity Name:** To focus on the **top 10 most-attended opportunities**, highlighting learner preferences.

Aspect	Details
How it matters	- Shows which opportunities attract more learners.- Helps in **resource allocation and planning**.- Identifies underperforming opportunities for improvement.- Highlights **learner trends and preferences** for the most popular offerings.
Importance	Important for understanding **learner engagement at the opportunity level**, enabling targeted interventions and optimizing content offerings.

Categorical

Categorical Variable Counts

```
#Gender - Pie Char
gender_counts = df['Gender'].value_counts(normalize=False)   # actual counts
gender_percent = df['Gender'].value_counts(normalize=True) * 100   # percentages

print("Gender Distribution (Counts & Percentages):")
for gender, count in gender_counts.items():
    print(f"{gender}: {count} ({gender_percent[gender]:.2f}%)")

# Pie Chart
plt.figure(figsize=(6,6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
        startangle=90, colors=['#66b3ff','#ff9999'])
plt.title("Gender Distribution")
plt.show()
```

```
# Opportunity Category - Bar Chart
plt.figure(figsize=(10,5))
ax = sns.countplot(
    data=df,
    x='Opportunity Category',
    order=df['Opportunity Category'].value_counts().index,
    palette="viridis"
)

plt.xticks(rotation=45)
plt.title("Opportunity Category Counts")

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height}',
                xy=(p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom')

plt.show()
```

```
#Country - Bar Chart for Top 10
top_countries = df['Country'].value_counts().nlargest(10)
plt.figure(figsize=(12,6))
sns.barplot(x=top_countries.index, y=top_countries.values, palette="magma")
plt.title("Top 10 Countries by Count")
plt.ylabel("Count")
plt.xticks(rotation=45)

for i, val in enumerate(top_countries.values):
    plt.text(i, val + 5, str(val), ha='center', va='bottom')

plt.show()
```

```
# Status description - Row Chart
plt.figure(figsize=(12,6))
ax = sns.countplot(
    data=df,
    y='Status Description',
    order=df['Status Description'].value_counts().index,
    palette="viridis"
)
plt.title("Status Description Counts")

for p in ax.patches:
    width = p.get_width()
    ax.annotate(f'{width}', xy=(width, p.get_y() + p.get_height()/2),
                ha='left', va='center')

plt.show()
```

```
# Opportunity Name - Row Chart
top_opportunities = df['Opportunity Name'].value_counts().nlargest(10)

plt.figure(figsize=(12,6))
ax = sns.barplot(x=top_opportunities.values, y=top_opportunities.index, palette="viridis")
plt.title("Top 10 Opportunity Names by Count")
```

Activate Windows
Go to Settings to activate Windows.

```python
for i, val in enumerate(top_opportunities.values):
    ax.text(val, i, str(val), va='center', ha='left')

plt.show()

# Top 10 Institution Names - Row Chart
top_institutions = df['Institution Name'].value_counts().nlargest(10)

plt.figure(figsize=(12,6))
ax = sns.barplot(x=top_institutions.values, y=top_institutions.index, palette='viridis')
plt.title("Top 10 Institutions by Learner Count")
plt.xlabel("Number of Learners")
plt.ylabel("Institution Name")

for i, val in enumerate(top_institutions.values):
    ax.text(val + 1, i, str(val), va='center', ha='left')

plt.show()
```

[329] ✓ 1.4s Python

```python
# Top 10 Current/Intended Major - Pie Chart
top_Major = df['Current/Intended Major'].value_counts().nlargest(10)

plt.figure(figsize=(8, 8))
plt.pie(
    top_Major.values,
    labels=top_Major.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=sns.color_palette("viridis", len(top_Major))
)
plt.title("Top 10 Current/Intended Major Distribution", fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

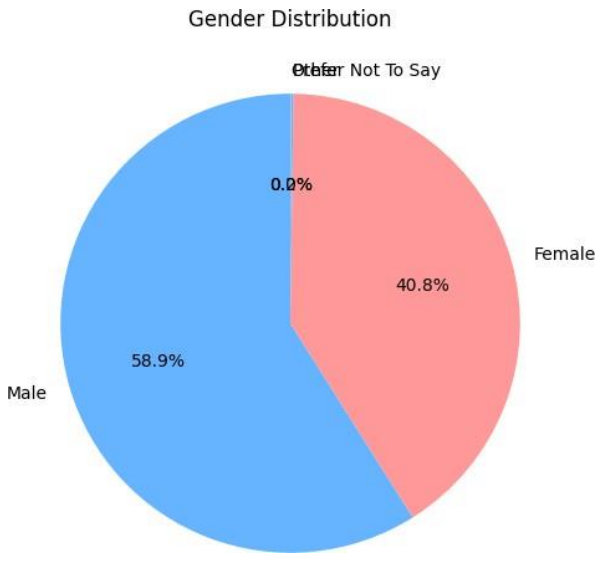
(function) fontweight: Literal['bold']

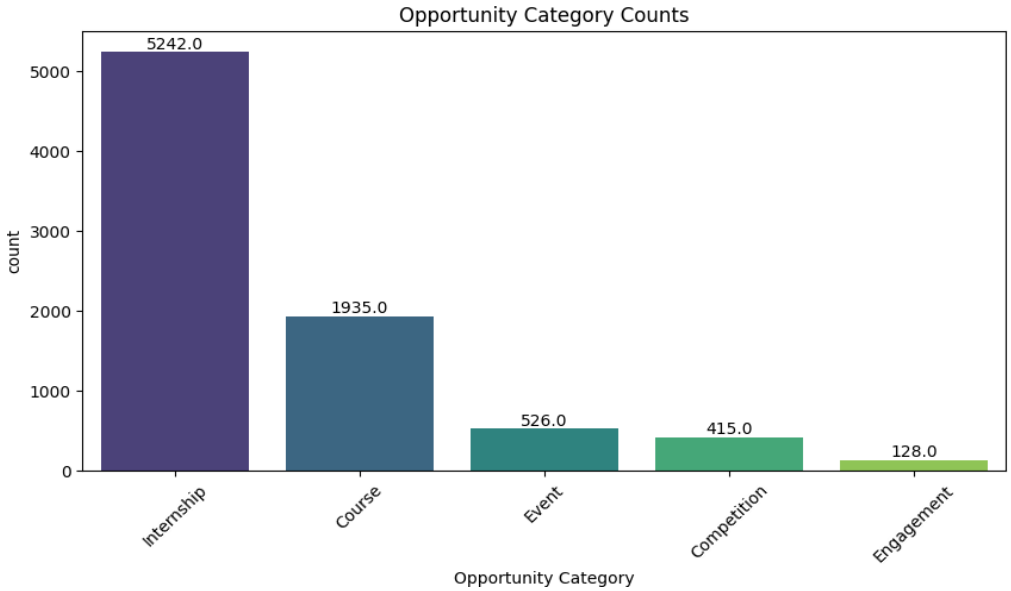
[8] ✓ 1.5s Python

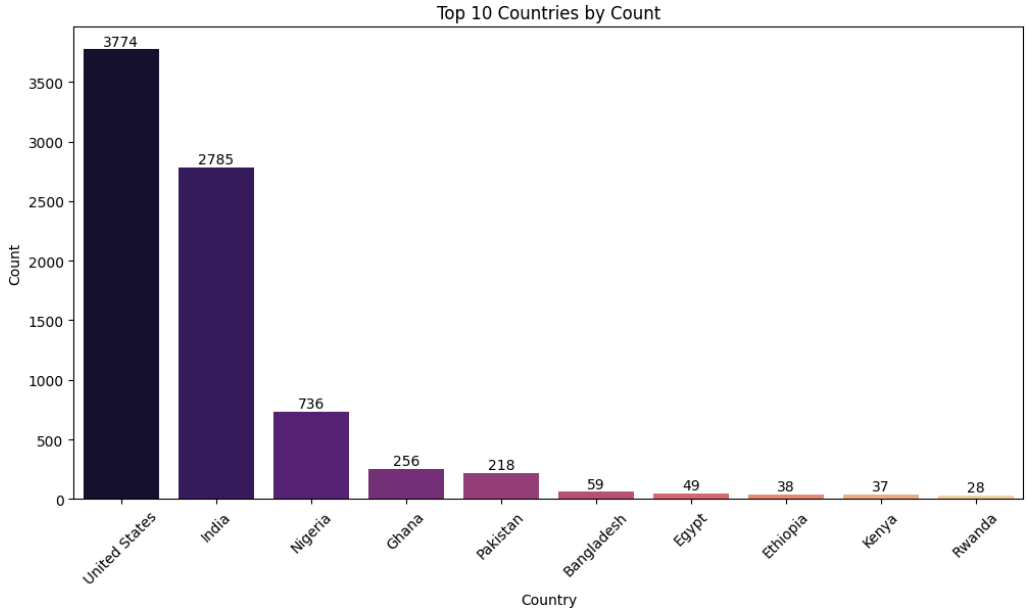
```python
# Top 10 most common first names
top_names = df['First Name'].value_counts().head(10)

plt.figure(figsize=(10,6))
ax = sns.barplot(x=top_names.values, y=top_names.index, palette="mako")
plt.title("Top 10 Most Common First Names", fontsize=16)
plt.xlabel("Frequency", fontsize=12)
plt.ylabel("First Name", fontsize=12)

# Add counts on bars
for i, val in enumerate(top_names.values):
    ax.text(val + 0.5, i, str(val), va='center', ha='left', fontsize=10)

plt.show()
```

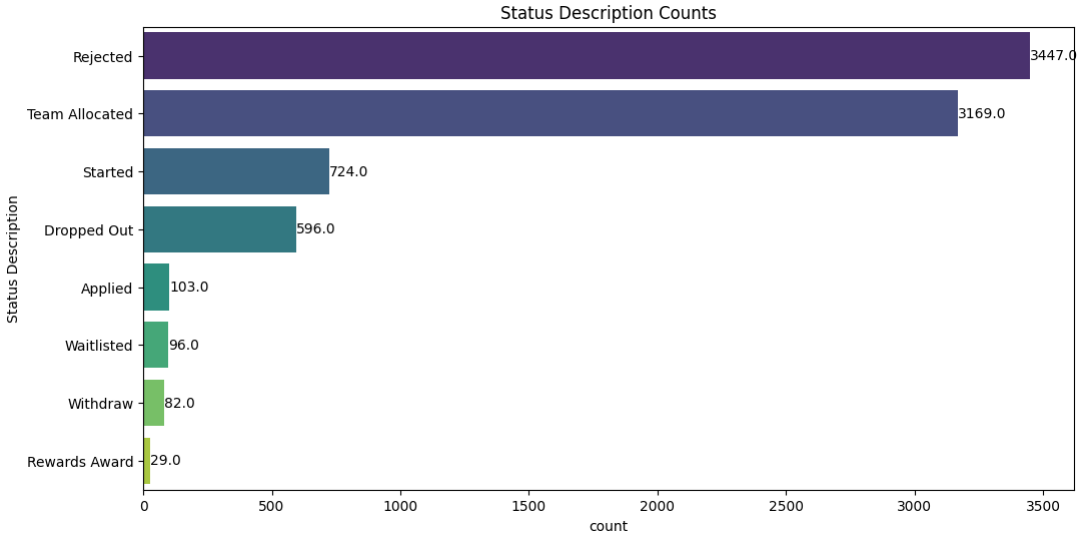
[24] ✓ 0.8s Python

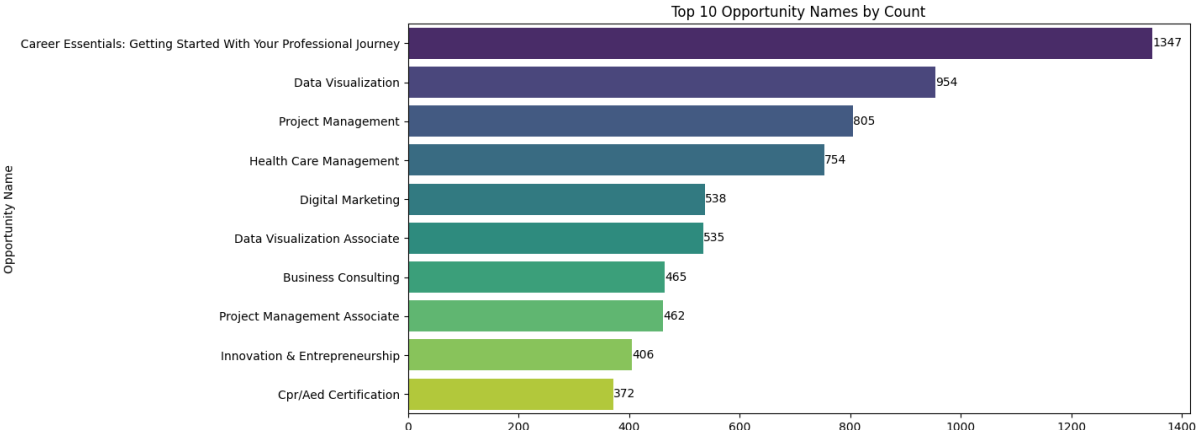
```
Gender Distribution (Counts & Percentages):
Male: 4861 (58.95%)
Female: 3367 (40.83%)
Prefer Not To Say: 15 (0.18%)
Other: 3 (0.04%)
```

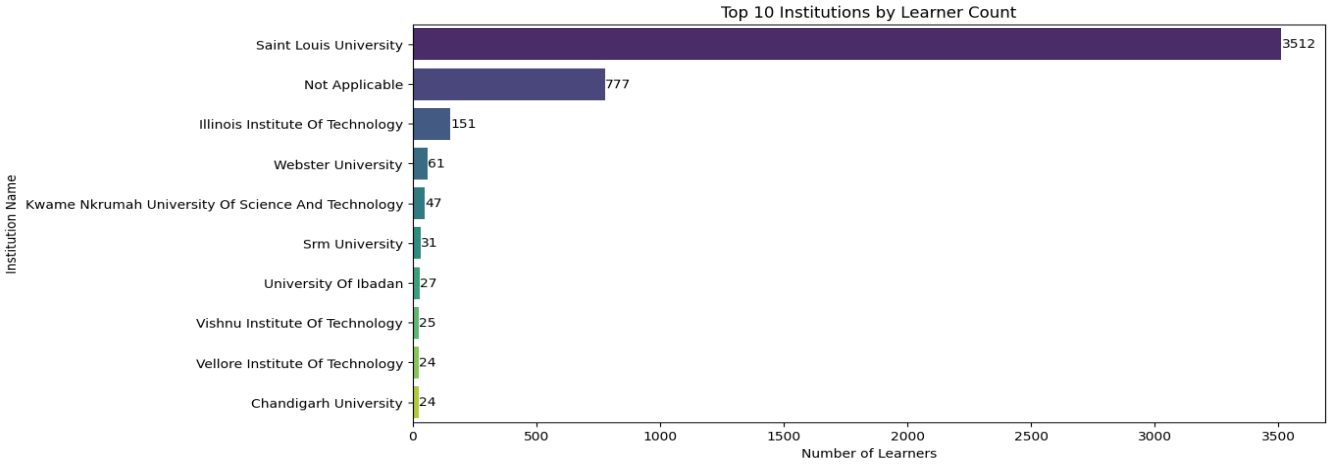


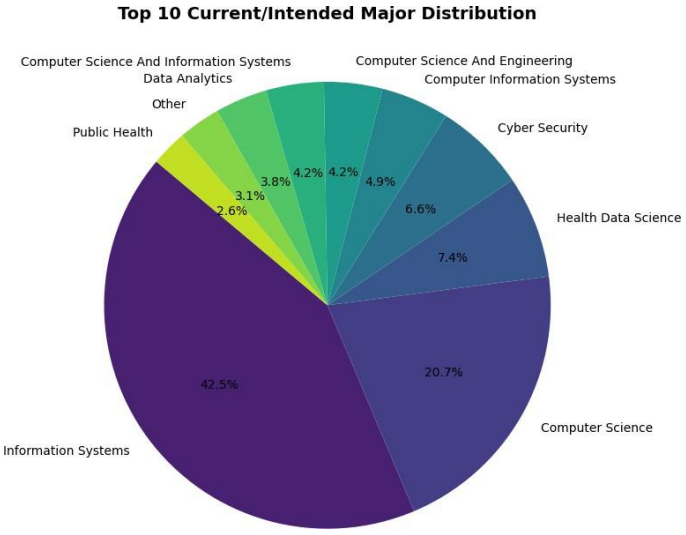

Opportunity Category Counts
5242.0
1935.0
526.0
415.0
128.0
count
Internship
Course
Event
Competition
Engagement
Opportunity Category

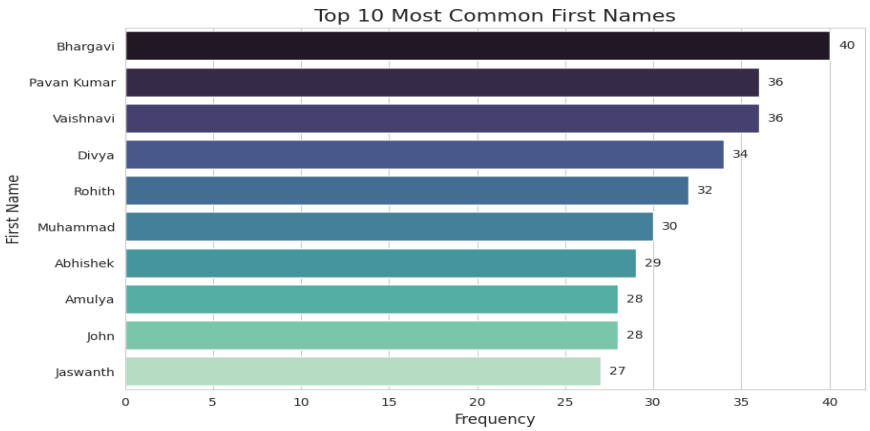

Top 10 Countries by Count
3774
2785
736
256
218
59
49
38
37
28
Count
United States
India
Nigeria
Ghana
Pakistan
Bangladesh
Egypt
Ethiopia
Kenya
Rwanda
Country


Status Description Counts
Rejected
Team Allocated
Started
Dropped Out
Applied
Waitlisted
Withdraw
Rewards Award
3447.0
3169.0
724.0
596.0
103.0
96.0
82.0
29.0
Status Description
count


Top 10 Opportunity Names by Count
Career Essentials: Getting Started With Your Professional Journey
Data Visualization
Project Management
Health Care Management
Digital Marketing
Data Visualization Associate
Business Consulting
Project Management Associate
Innovation & Entrepreneurship
Cpr/Aed Certification
1347
954
805
754
538
535
465
462
406
372
Opportunity Name
0
200
400
600
800
1000
1200
1400


Top 10 Institutions by Learner Count
Saint Louis University
Not Applicable
Illinois Institute Of Technology
Webster University
Kwame Nkrumah University Of Science And Technology
Srm University
University Of Ibadan
Vishnu Institute Of Technology
Vellore Institute Of Technology
Chandigarh University
3512
777
151
61
47
31
27
25
24
24
Institution Name
0
500
1000
1500
2000
2500
3000
3500
Number of Learners


Top 10 Current/Intended Major Distribution
Computer Science And Information Systems
Data Analytics
Other
Public Health
Computer Science And Engineering
Computer Information Systems
Cyber Security
Health Data Science
Computer Science
Information Systems
42.5%
20.7%
7.4%
6.6%
4.9%
4.2%
4.2%
3.8%
3.1%
2.6%


Top 10 Most Common First Names
Bhargavi
Pavan Kumar
Vaishnavi
Divya
Rohith
Muhammad
Abhishek
Amulya
John
Jaswanth
40
36
36
34
32
30
29
28
28
27
First Name
0
5
10
15
20
25
30
35
40
Frequency

Aspect	Details
Figure	Gender Distribution (Pie Chart) , Opportunity Category Counts (Bar Chart) , Top 10 Countries by Count (Bar Chart), Status Description Counts (Row Chart), Top 10 Opportunity Names by Count (Row Chart), Top 10 Institutions by Learner Count (Row Chart), Top 10 Current/Intended Major (Pie Chart), Top 10 Most Common First Names (Bar Chart)
Columns used	Gender, Opportunity Category, Country, Status Description, Opportunity Name, Institution Name, Current/Intended Major, First Name
Why used	To visualize categorical distributions across key features, highlighting patterns in learner demographics, participation, institutional representation, academic interest areas, and common names. Pie charts, bar charts, and row charts provide an intuitive understanding of counts and proportions.
How it matters	- Gender Pie Chart shows the proportion of male and female learners - Opportunity Category and Status Description charts identify popular categories and statuses. - Top 10 Countries, Opportunity Names, and Institutions highlight key contributors and trends. - Top 10 Majors Pie Chart highlights learners' academic preferences, showing the most common fields of study. - Top 10 First Names Bar Chart shows the most frequent names among learners, which can be useful for personalization or demographic insights. - Helps in decision-making, resource allocation, and targeting strategies.
Importance	These visualizations are important for stakeholder insights, program planning, and understanding learner engagement patterns across demographics, institutions, and academic interests. They also help in detecting imbalances or gaps in participation that may require attention, and provide insight into learner naming trends for personalization or community engagement.

3) Visualizing Data Distributions:

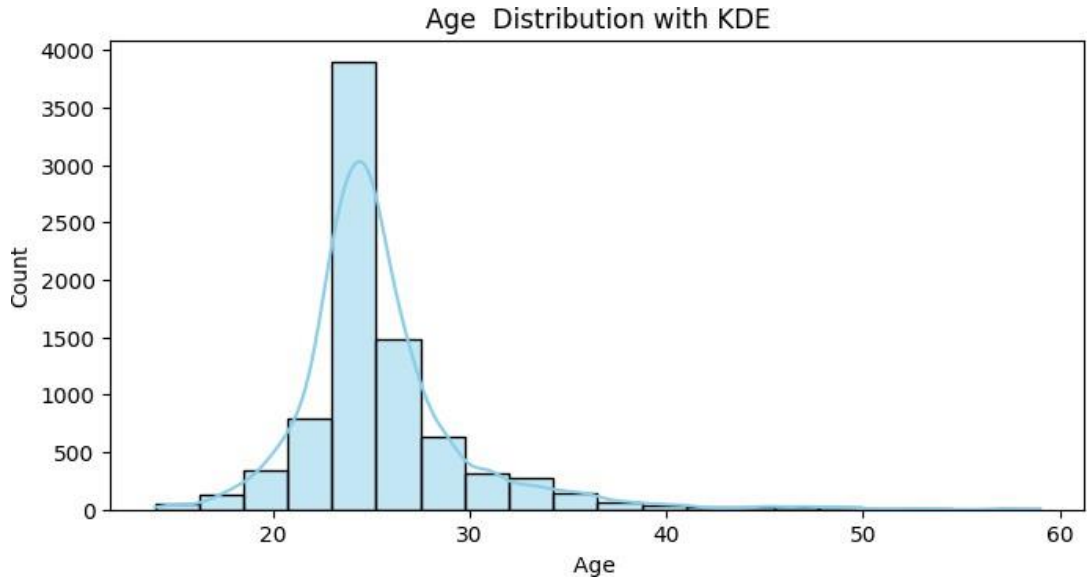
Numerics

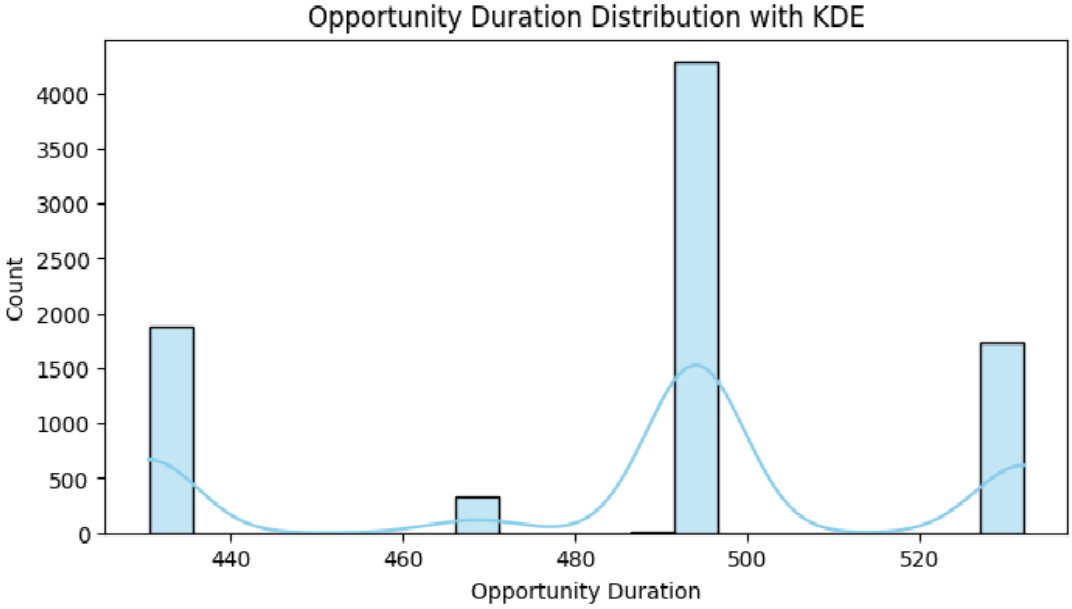
Distribution of Numeric Variables

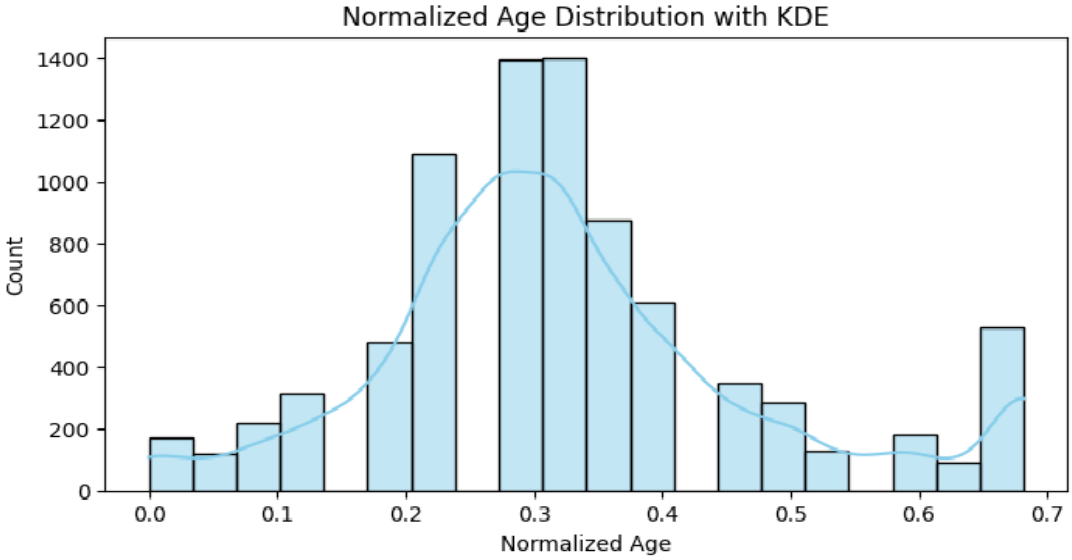
Combined Histograms with Density Overlay

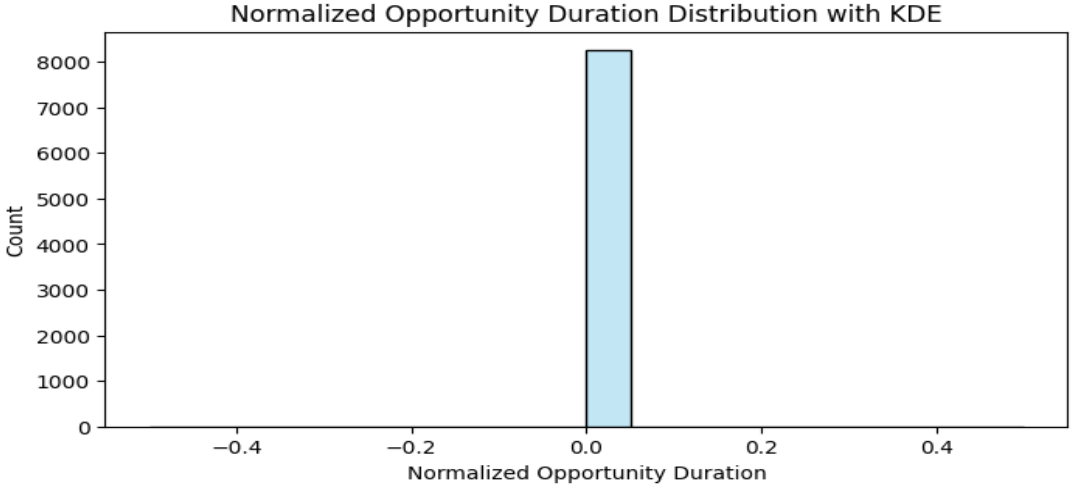
```
for col in numeric_cols:
    plt.figure(figsize=(8,4))
    sns.histplot(df[col], bins=20, kde=True, color='skyblue')
    plt.title(f"{col} Distribution with KDE")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.show()
```

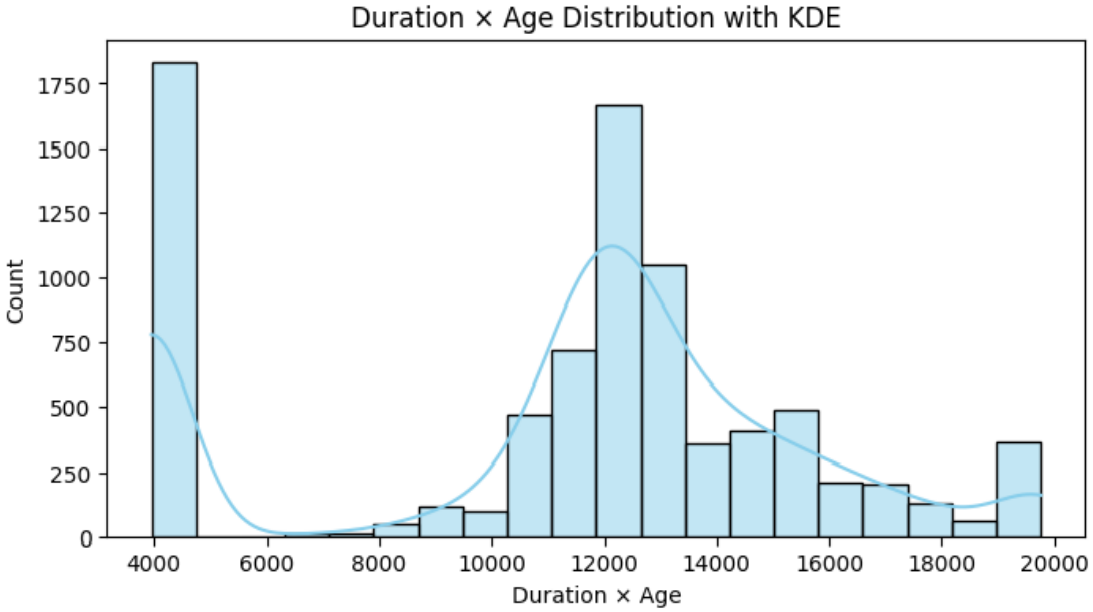
Histograms: Detects unusual peaks or gaps.

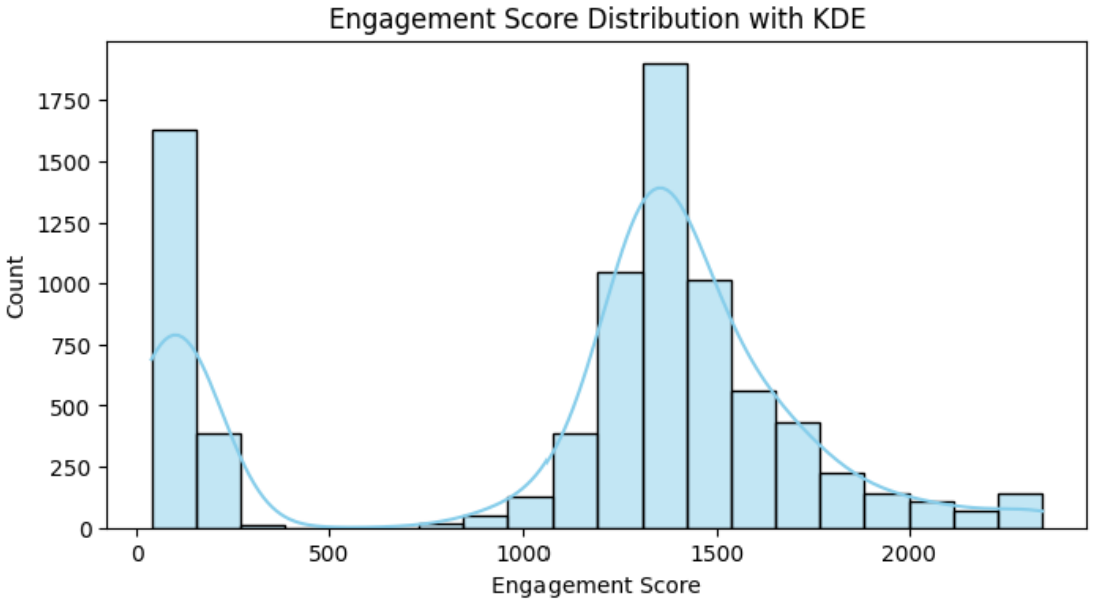



Opportunity Duration Distribution with KDE
Count
Opportunity Duration
0
500
1000
1500
2000
2500
3000
3500
4000
440
460
480
500
520


Normalized Age Distribution with KDE
Count
Normalized Age
0
200
400
600
800
1000
1200
1400
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Normalized Opportunity Duration Distribution with KDE
Count
Normalized Opportunity Duration
0
1000
2000
3000
4000
5000
6000
7000
8000
−0.4
−0.2
0.0
0.2
0.4

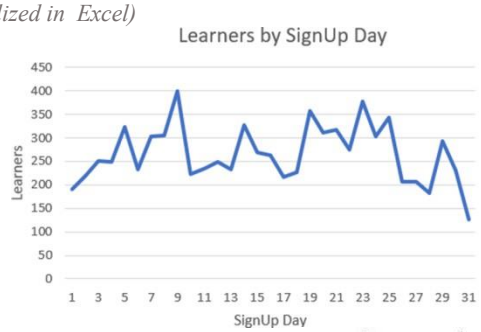


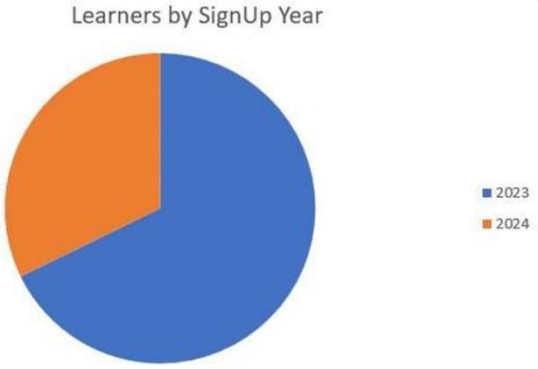


Aspect	Details
Figure	Distribution of Numeric Variables with KDE (Histogram)
Columns used	Age, Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score
Why used	To visualize the **distribution and density** of key numeric features. Histograms show frequency, while KDE overlays show the estimated probability density of the data.
How it matters	- Helps understand the **spread, central tendency, and skewness** of numeric features. - Identifies **potential outliers or unusual patterns**. - Useful for **data preprocessing** and selecting appropriate statistical methods. - Assists stakeholders in understanding the **overall behavior** of numeric variables.
Importance	KDE curves provide a **smooth view of data distribution**, making patterns easier to interpret than raw histograms alone.

Dates

(Visualized in Excel)





Aspect	Learners by Signup Day	Learners by Signup Month	Learners by Signup Year
Figure	Bar chart showing learner counts per day of the month	Line chart showing monthly learner signups	Line chart showing yearly learner signups
Columns used	Extracted SignUp Day	Extracted SignUp Month	Extracted SignUp Year
Why used	To visualize daily trends in learner signups	To identify seasonal patterns and peak months	To identify yearly growth or decline trends
How it matters	Shows which days have higher learner activity, useful for short-term planning	Helps allocate resources or campaigns during high-demand months	Supports long-term planning and trend analysis
Importance	Useful for **day-level engagement insights**	Highlights **seasonal engagement patterns**	Enables **strategic decision-making over years**

Reveal Patterns and Trends: (DATES)

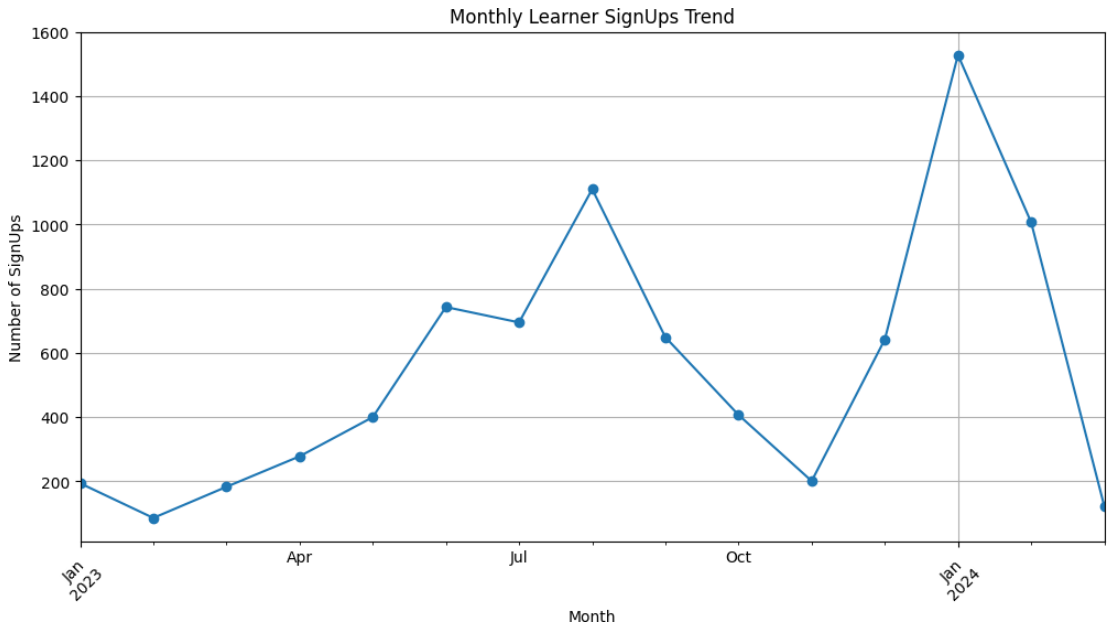
Line chart of Monthly Learner SignUps Trend

```
# Convert to datetime if not already
df['Learner SignUp DateTime'] = pd.to_datetime(df['Learner SignUp DateTime'])

# Count of signups per month
monthly_signups = df.groupby(df['Learner SignUp DateTime'].dt.to_period('M')).size()

# Plot line chart
plt.figure(figsize=(12,6))
monthly_signups.plot(kind='line', marker='o')
plt.title("Monthly Learner SignUps Trend")
plt.xlabel("Month")
plt.ylabel("Number of SignUps")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

[57] ✓ 0.2s

Activate Windows
Go to Settings to activate Windows.

Python

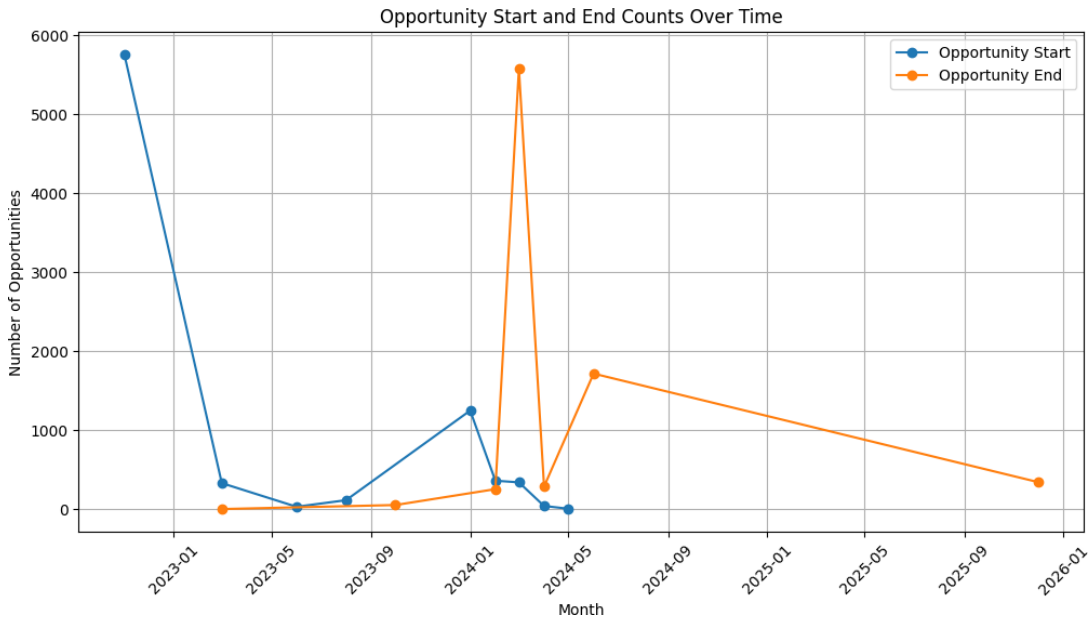


Aspect	Details
Figure	Monthly Learner SignUps Trend (Line Chart)
Columns used	- Learner SignUp DateTime
Why used	To visualize **temporal trends** in learner sign-ups over time. Line charts help track patterns, growth, or seasonal fluctuations in registrations.
How it matters	- Identifies **peak months and trends** in learner registration. - Helps in **planning resources, programs, and campaigns** according to demand. - Useful for detecting **seasonal patterns** and growth rates.
Importance	Line charts provide a **clear visual of trends over time**, allowing stakeholders to make informed decisions for scheduling and outreach strategies.

Opportunity Start Date and Opportunity End Date, a line chart can reveal trends over time

```python
# Convert to datetime if not already
df['Opportunity Start Date'] = pd.to_datetime(df['Opportunity Start Date'])
df['Opportunity End Date'] = pd.to_datetime(df['Opportunity End Date'])

# Aggregate by month
start_counts = df.groupby(df['Opportunity Start Date'].dt.to_period('M')).size()
end_counts = df.groupby(df['Opportunity End Date'].dt.to_period('M')).size()

# Convert PeriodIndex to datetime for plotting
start_counts.index = start_counts.index.to_timestamp()
end_counts.index = end_counts.index.to_timestamp()

# Plot line chart
plt.figure(figsize=(12,6))
plt.plot(start_counts.index, start_counts.values, marker='o', label='Opportunity Start')
plt.plot(end_counts.index, end_counts.values, marker='o', label='Opportunity End')
plt.title("Opportunity Start and End Counts Over Time")
plt.xlabel("Month")
plt.ylabel("Number of Opportunities")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.show()
```

Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	Opportunity Start and End Counts Over Time (Line Chart)
Columns used	Opportunity Start Date, Opportunity End Date
Why used	To visualize **temporal trends** in when opportunities begin and end. Line charts help track scheduling patterns and detect overlaps or gaps.
How it matters	- Shows trends in **opportunity availability** over time. - Helps in **planning resources** and forecasting periods of high or low activity. - Useful for detecting **seasonal or monthly patterns** in opportunity scheduling.
Importance	Comparing start and end counts provides insights into **opportunity durations and distribution**, helping stakeholders manage timing and workload efficiently.

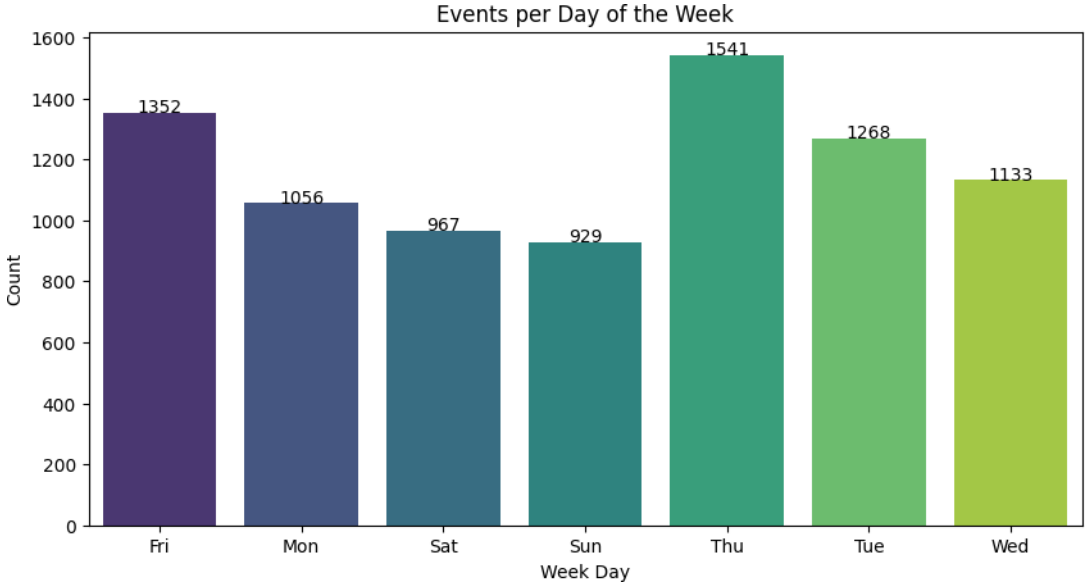
4) Enhance Communication:

Seasonal Patterns

Bar chart → counts per week day

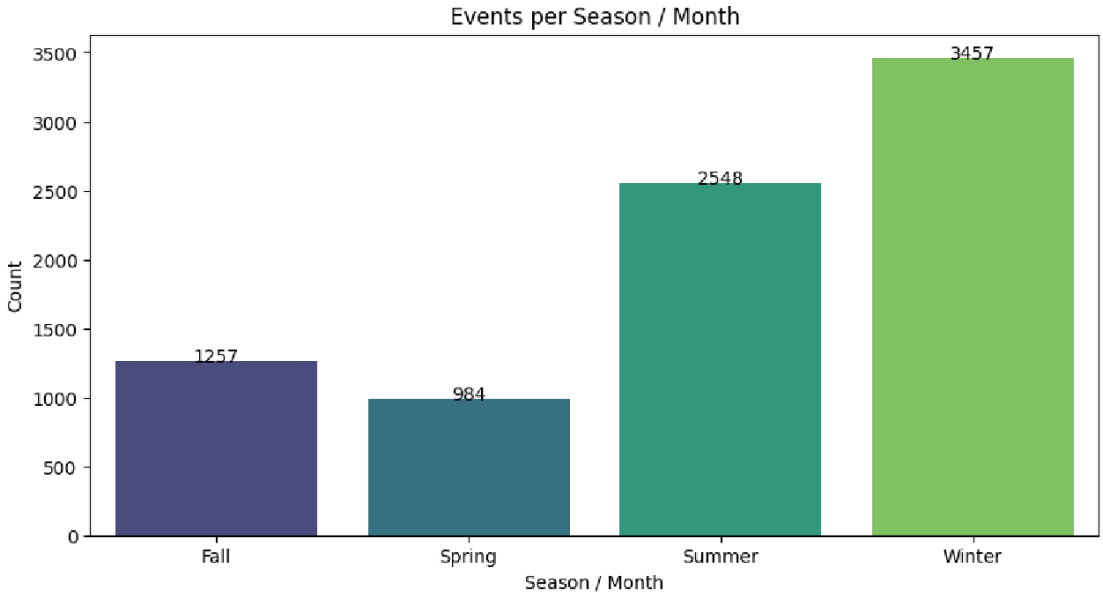
```python
# Count of events per week day
weekly_counts = df['Weekly Patterns'].value_counts().sort_index()  # assuming Monday=0 ... Sunday=6

plt.figure(figsize=(10,5))
sns.barplot(x=weekly_counts.index, y=weekly_counts.values, palette='viridis')
plt.title("Events per Day of the Week")
plt.xlabel("Week Day")
plt.ylabel("Count")

# Annotate counts
for i, val in enumerate(weekly_counts.values):
    plt.text(i, val+1, str(val), ha='center')

plt.show()
```

0.1s Python



Bar chart → counts per season

```python
# Count per season/month
season_counts = df['Seasonal Patterns'].value_counts().sort_index()  # e.g., Jan-Dec or Winter, Spring, Summer, Autumn

plt.figure(figsize=(10,5))
sns.barplot(x=season_counts.index, y=season_counts.values, palette='viridis')
plt.title("Events per Season / Month")
plt.xlabel("Season / Month")
plt.ylabel("Count")

# Annotate counts
for i, val in enumerate(season_counts.values):
    plt.text(i, val+1, str(val), ha='center')

plt.show()
```



Aspect	Details
Figure	Events per Day of the Week (Bar Chart) Events per Season / Month (Bar Chart)
Columns used	- Weekly Patterns (Weekdays) - Seasonal Patterns (Seasons/Months)
Why used	To visualize **temporal distribution of events** across weekdays and seasons/months, helping identify patterns in learner activity or opportunity scheduling.

Aspect	Details
How it matters	- Shows which **days of the week** have higher or lower event counts.- Reveals **seasonal trends or monthly peaks** in events.- Useful for **planning and resource allocation** based on temporal activity patterns.- Helps detect patterns or gaps in engagement over time.
Importance	Annotating counts on bars provides a **clear numeric perspective**, making trends easier to interpret and communicate to stakeholders.

5) Calculating initial summary statistics:

Calculating initial summary statistics

```
# Select numeric columns
numeric_cols = ["Age ", "Opportunity Duration", "Normalized Age",
                "Normalized Opportunity Duration", "Duration x Age", "Engagement Score"]

# Basic summary statistics
summary_stats = df[numeric_cols].describe().T  # Transpose for easier reading
summary_stats['range'] = summary_stats['max'] - summary_stats['min']  # Add range column
summary_stats['median'] = df[numeric_cols].median()  # Add median column

# Reorder columns for clarity
summary_stats = summary_stats[['count', 'mean', 'median', 'std', 'min', 'max', 'range']]

print(summary_stats)
```

[56] ✓ 0.0s Python

```
                                count          mean        median  \
Age                            8246.0     25.456706     25.000000
Opportunity Duration           8246.0    486.503292    493.978715
Normalized Age                 8246.0      0.326891      0.318182
Normalized Opportunity Duration  8246.0      0.000000      0.000000
Duration x Age                 8246.0  11255.051332  12080.305560
Engagement Score               8246.0   1120.988366   1326.818928

                                     std          min           max  \
Age                             4.336149    14.000000     59.000000
Opportunity Duration           34.580794   430.470874    532.083420
Normalized Age                  0.151775     0.000000      0.681818
Normalized Opportunity Duration     0.000000     0.000000      0.000000
Duration x Age                4477.947762  3951.829720  19759.148616
Engagement Score               625.582652    37.893238   2345.694043

                                    range
Age                             45.000000
Opportunity Duration           101.612546
Normalized Age                   0.681818
Normalized Opportunity Duration      0.000000
Duration x Age                15807.318896
Engagement Score               2307.800804
```

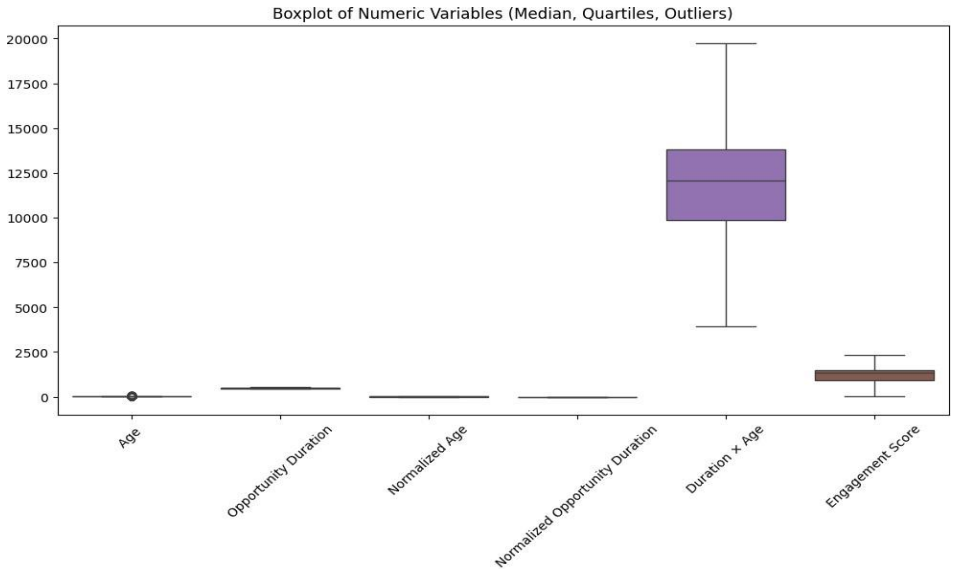
Activate Windows
Go to Settings to activate Windows.

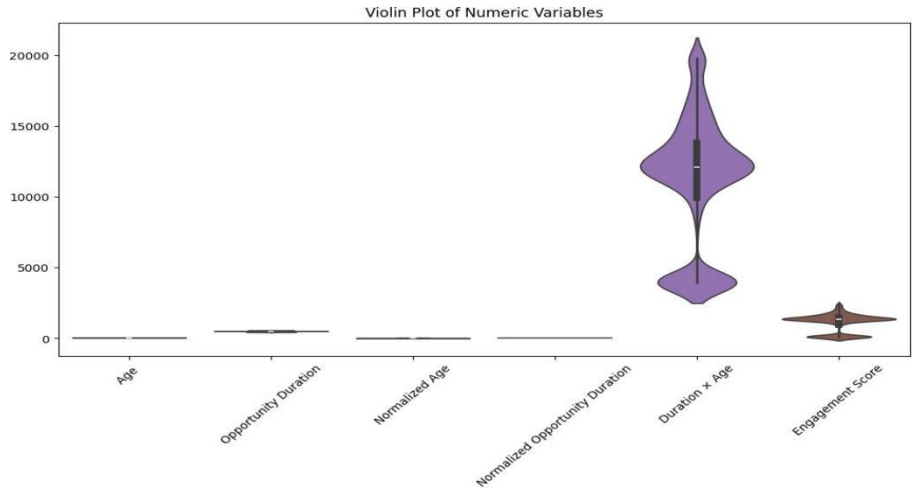
Boxplots (for spotting outliers and spread)

Generate + Code + Markdown

```
plt.figure(figsize=(12,6))
sns.boxplot(data=df[numeric_cols])
plt.title("Boxplot of Numeric Variables (Median, Quartiles, Outliers)")
plt.xticks(rotation=45)
plt.show()
```

[55] ✓ 0.3s Python





Aspect	Details
Figure	Boxplot and Violin Plot of Numeric Variables (Median, Quartiles, Outliers, and Distribution)
Columns used	Age, Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score
Why used	To visualize the **distribution, central tendency, spread, and outliers** of all numeric features. Boxplots highlight medians, quartiles, and outliers, while violin plots show the **full distribution and density** of each variable.
How it matters	- Boxplots show **medians, quartiles, and outliers**, helping detect variability and extreme values. - Violin plots reveal **density and skewness**, showing the shape of the data. - Together, they allow **comparison across multiple features** and help identify potential data transformations.- Useful for understanding patterns before modeling or statistical analysis.
Importance	Combining boxplots and violin plots provides a **comprehensive overview** of numeric data, highlighting both summary statistics and detailed distributions for informed decision-making.

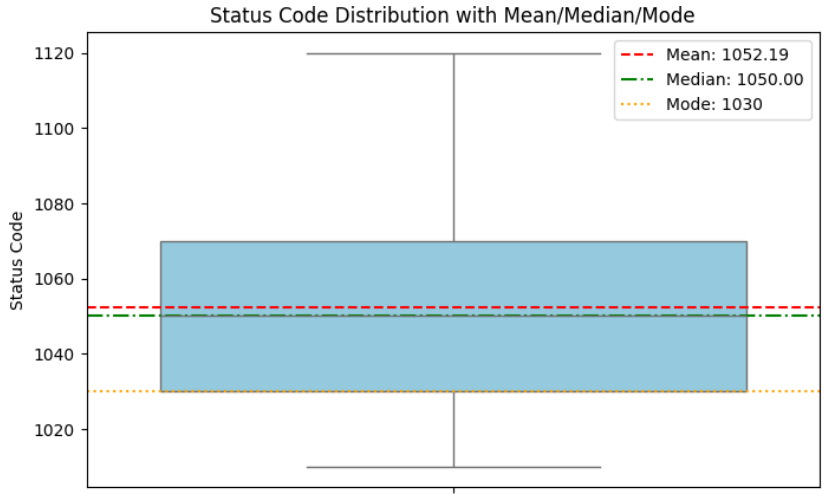
A box plot with mean/median/mode overlay of Status Code

```python
# Calculate statistics
mean_status = df['Status Code'].mean()
median_status = df['Status Code'].median()
mode_status = df['Status Code'].mode()[0]

print(f"Mean: {mean_status}")
print(f"Median: {median_status}")
print(f"Mode: {mode_status}")

# Boxplot to visualize distribution and outliers
plt.figure(figsize=(8,5))
sns.boxplot(y=df['Status Code'], color='skyblue')
plt.title("Status Code Distribution with Mean/Median/Mode")

# Overlay mean, median, mode
plt.axhline(mean_status, color='red', linestyle='--', label=f'Mean: {mean_status:.2f}')
plt.axhline(median_status, color='green', linestyle='-.', label=f'Median: {median_status:.2f}')
plt.axhline(mode_status, color='orange', linestyle=':', label=f'Mode: {mode_status:.0f}')

plt.ylabel("Status Code")
plt.legend()
plt.show()
```

Activate Windows
Go to Settings to activate Windows.

✓ 0.2s Python

```
Mean: 1052.1865146737812
Median: 1050.0
Mode: 1030
```



Aspect	Details
Figure	Status Code Distribution with Mean, Median, and Mode (Boxplot)
Columns used	- Status Code
Why used	To summarize the central tendency and distribution of the Status Code. Mean, median, and mode provide key statistical measures, while the boxplot visualizes the spread and potential outliers.
How it matters	- Helps understand the **typical status code** and how values are distributed. - Identifies **outliers** that may require attention. - Comparing mean, median, and mode shows **skewness** or asymmetry in the data. - Useful for monitoring overall **status trends** in opportunities.
Importance	Overlaying mean, median, and mode on the boxplot provides an intuitive visual summary of the data's central tendencies and variability.

Insight Generation and Hypothesis Development

Insight generation and hypothesis development play a vital role in transforming raw data into meaningful knowledge. By carefully analyzing patterns, trends, and anomalies, organizations can generate actionable insights that guide decision-making. Hypothesis development further allows the formulation of testable assumptions, ensuring that strategies and solutions are grounded in evidence rather than intuition.

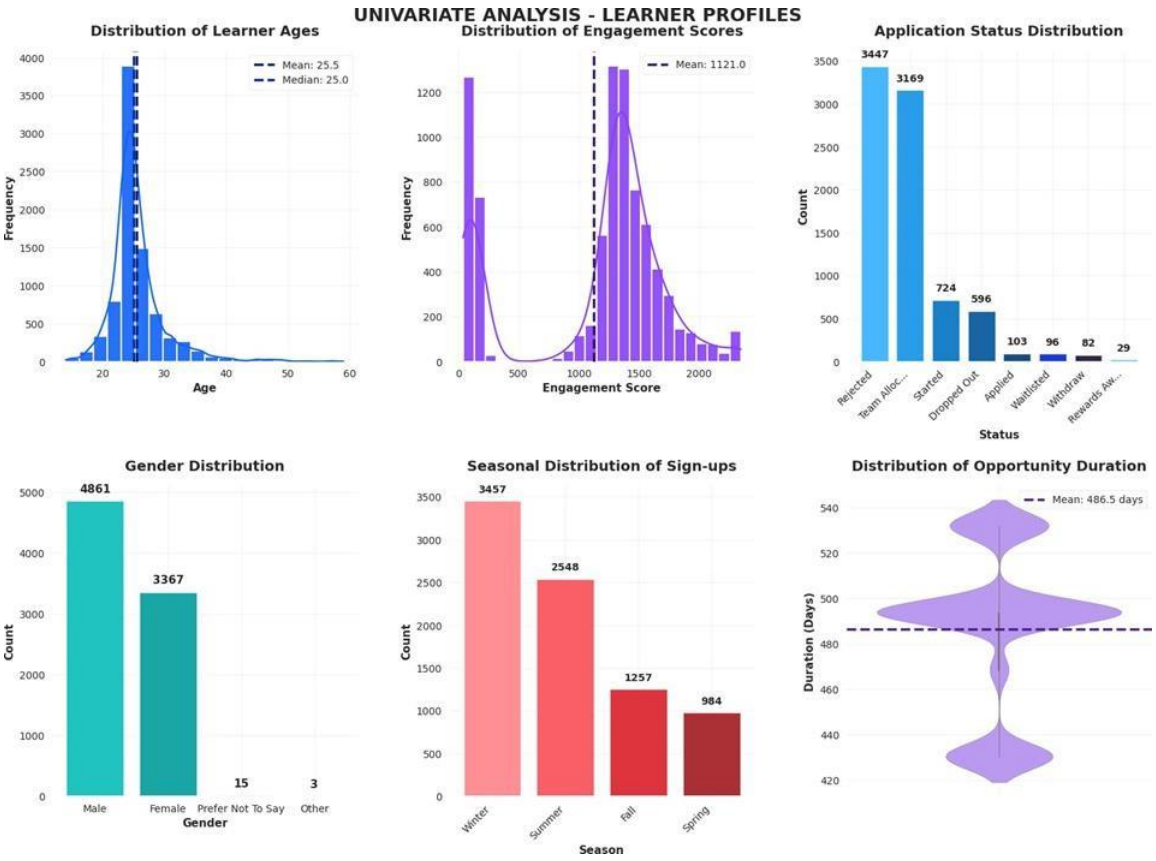
Importance

- Provides a structured approach to exploring and understanding complex data.
- Helps identify potential causes, correlations, and opportunities.
- Ensures research and business decisions are evidence-based.
- Strengthens problem-solving through systematic testing and validation.

Benefits

- Improves decision-making accuracy and reduces risks.
- Supports innovation by uncovering hidden opportunities.
- Enhances efficiency in experiments and analyses.
- Builds a foundation for predictive modeling and advanced analytics.
- Promotes data-driven culture across teams and organizations.

1) Understanding Data Through Univariate and Bivariate Analysis



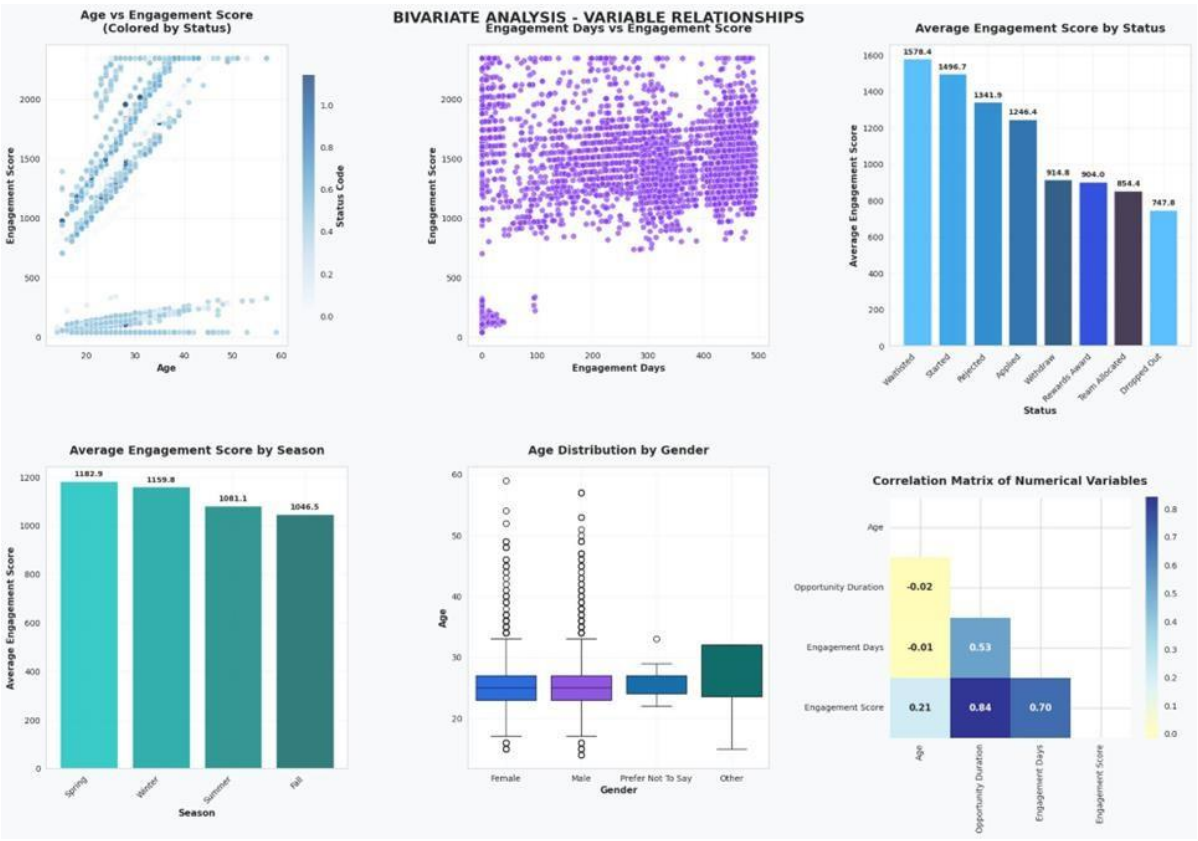


Table 1: Univariate Analysis

Aspect	Details
Figure	1. Distribution of Learner Ages 2. Distribution of Engagement Scores 3. Application Status Distribution 4. Gender Distribution 5. Seasonal Patterns 6. Opportunity Duration Distribution
Columns used	- Age - Engagement Score - Status Description - Gender - Seasonal Patterns - Opportunity Duration
Why used	To explore individual variable distributions and identify patterns, outliers, and central tendencies (mean, median).
How it matters	- Understand **learner demographics and engagement patterns**.- Reveal trends in **opportunity duration, seasonal participation, and status distribution**.- Guides feature selection and further analyses.
Importance	Provides a **baseline understanding** of dataset characteristics, enabling better decisions for targeted interventions and program optimization.

Table 2: Bivariate Analysis

Aspect	Details
Figure	1. Age vs Engagement Score (Colored by Status) 2. Engagement Days vs Engagement Score 3. Average Engagement Score by Status 4. Average Engagement Score by Season 5. Age Distribution by Gender 6. Correlation Matrix of Numerical Variables
Columns used	- Age - Engagement Score - Normalized Status Code - Engagement Days - Status Description - Seasonal Patterns - Gender - Opportunity Duration
Why used	To explore **relationships between two variables** and identify correlations, patterns, and trends across different categories or groups.
How it matters	- Highlights **key relationships** influencing engagement.- Identifies **strongly correlated variables**, supporting feature selection.- Provides insight into **demographic and temporal patterns** affecting engagement.
Importance	Supports **data-driven decision-making** by revealing interaction effects, helping optimize content delivery and learner engagement strategies.

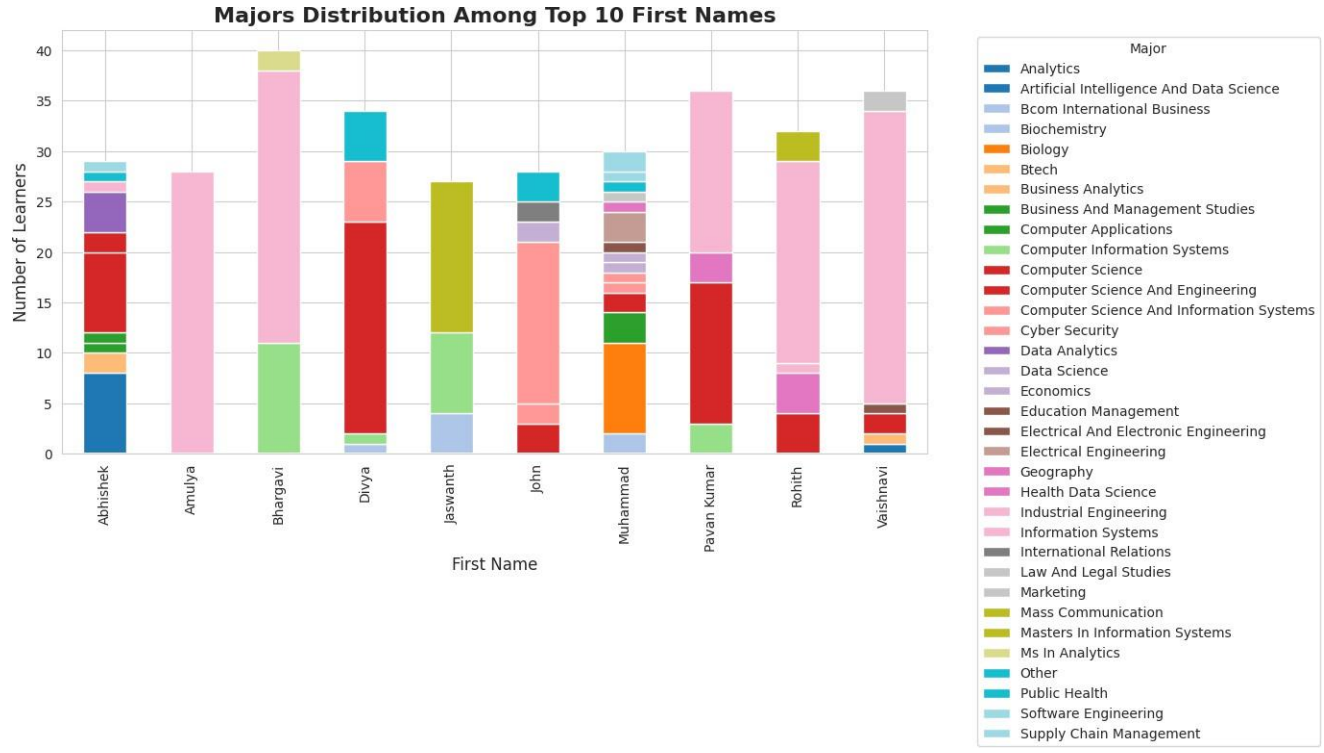
2) Compare Data Groups:

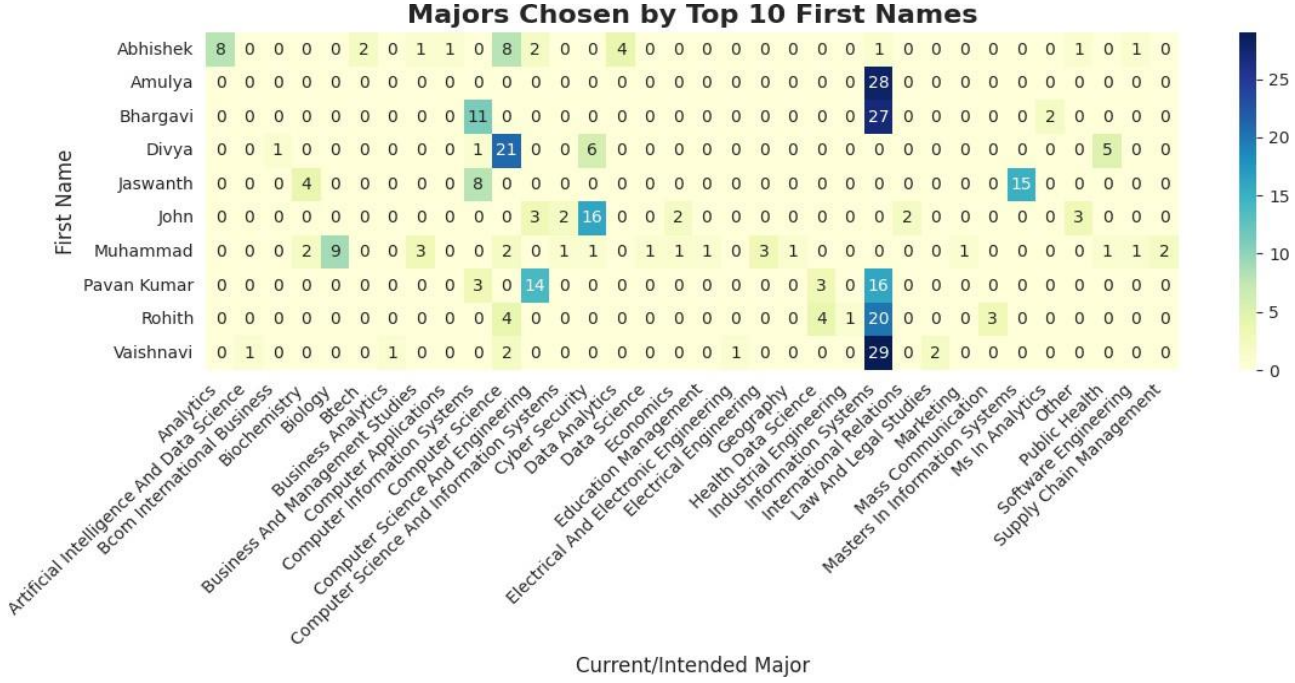
```
# ================================================================
# 📊 First Name vs Major Analysis
# ================================================================

# Ensure clean font/style
sns.set_style("whitegrid")
plt.rcParams['font.family'] = 'DejaVu Sans'

# 1. Get Top 10 First Names
top_names = df['First Name'].value_counts().head(10).index

# 2. Filter dataset for those names
filtered = df[df['First Name'].isin(top_names)]

# 3. Cross-tab: First Name vs Current/Intended Major
name_major = pd.crosstab(filtered['First Name'], filtered['Current/Intended Major'])

# ================================================================
# 🔹 Visualization 1: Stacked Bar Chart
# ================================================================
plt.figure(figsize=(14,7))
name_major.plot(kind='bar', stacked=True, figsize=(14,7), colormap='tab20')
```

Activate Windows
Go to Settings to activate Windows.

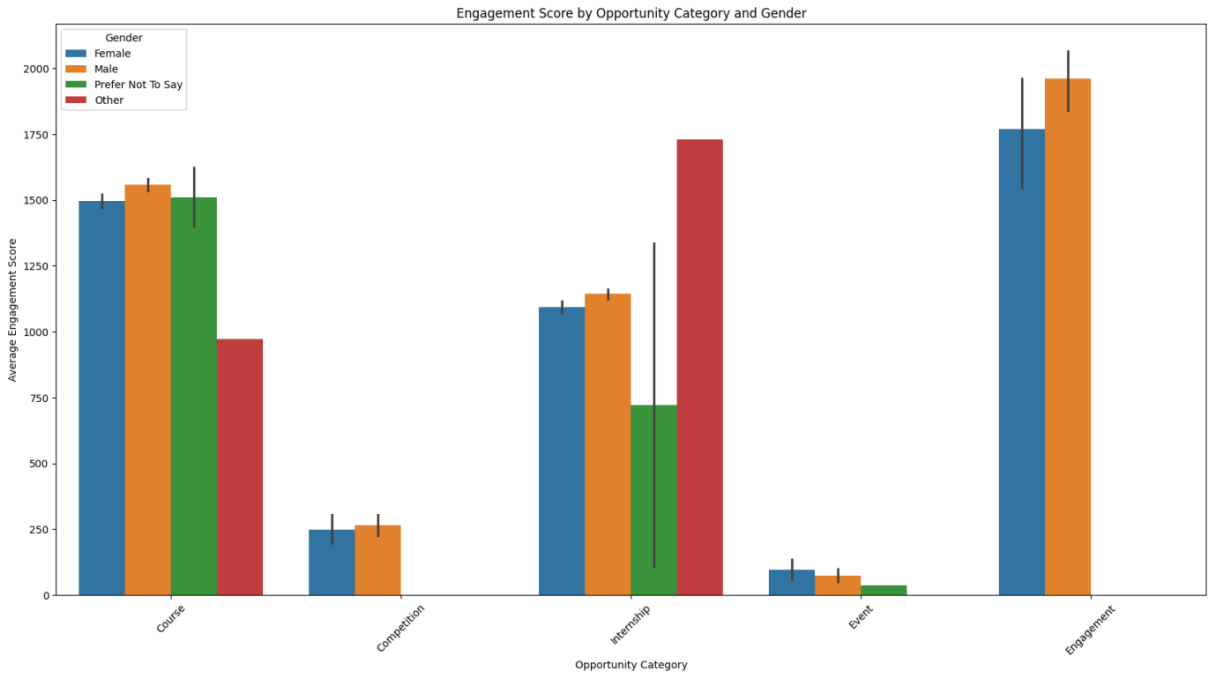
```
plt.title("Majors Distribution Among Top 10 First Names", fontsize=16, fontweight='bold')
plt.xlabel("First Name", fontsize=12)
plt.ylabel("Number of Learners", fontsize=12)
plt.legend(title="Major", bbox_to_anchor=(1.05,1), loc='upper left')
plt.tight_layout()
plt.show()

# ================================================================
# 🔹 Visualization 2: Heatmap
# ================================================================
plt.figure(figsize=(12,6))
sns.heatmap(name_major, cmap="YlGnBu", annot=True, fmt="d")

plt.title("Majors Chosen by Top 10 First Names", fontsize=16, fontweight='bold')
plt.xlabel("Current/Intended Major", fontsize=12)
plt.ylabel("First Name", fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Activate Windows





Aspect	Details
Figure	1. Stacked Bar Chart – Majors Distribution Among Top 10 First Names 2. Heatmap – Majors Chosen by Top 10 First Names
Columns used	First Name, Current/Intended Major
Why used	To analyze how the most common first names among learners relate to their chosen or intended majors. Stacked bar charts show proportional distributions across majors for each name, while the heatmap highlights frequency patterns with intensity.
How it matters	- Reveals if certain first names (possibly linked to cultural/regional groups) have concentrated preferences for particular majors. - Stacked bar chart helps compare distribution visually across names. - Heatmap allows easy spotting of strong/weak associations between names and majors.
Importance	Provides demographic insights into naming trends and academic interests. Useful for identifying cultural or regional influences in field selection, which can inform outreach strategies, diversity analysis, and targeted engagement.



Aspect	Details
Figure	Engagement Score by Opportunity Category and Gender (Bar Chart)
Columns used	Opportunity Category, Engagement Score, Gender
Why used	To compare **average engagement scores** across opportunity categories for different genders. Helps identify disparities or trends in participation and engagement.
How it matters	- Highlights which opportunity categories are **more engaging for male or female learners**. - Supports **targeted interventions** to improve engagement. - Useful for understanding **demographic differences** in participation.
Importance	This visualization is important for **decision-making and program optimization**, ensuring equitable and effective engagement across genders.

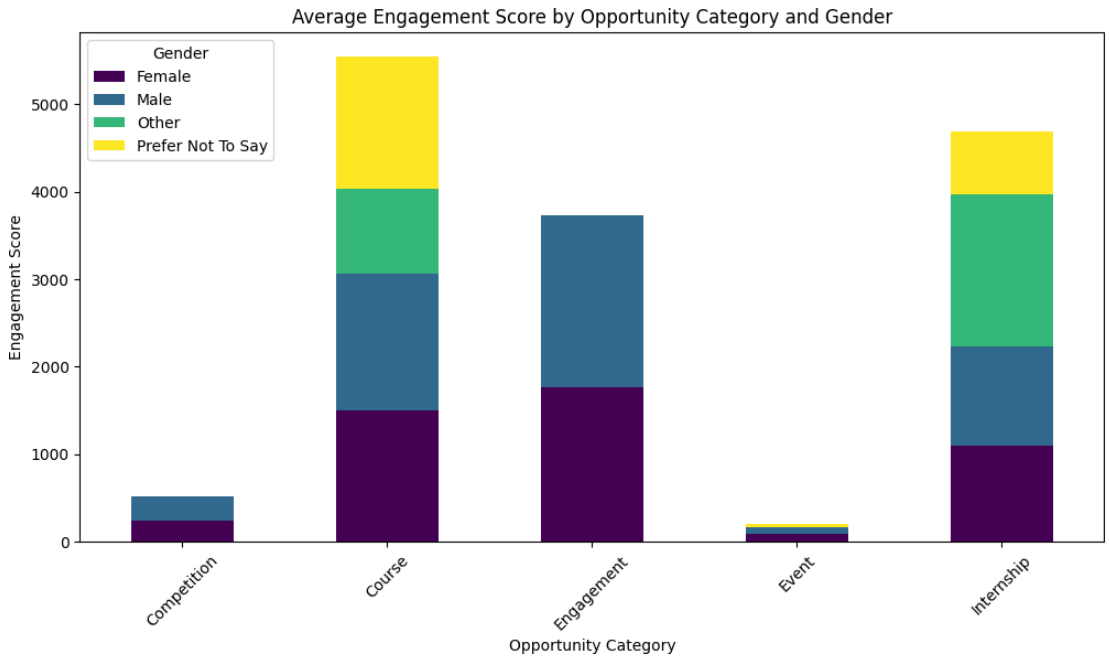
Stacked Bar Charts (Comparison Across Multiple Categories)

Comparing multiple categorical variables in one chart.

Example: Engagement Score distribution across Opportunity Category and Gender.

```
# Aggregate data: mean Engagement Score by Opportunity Category and Gender
agg_df = df.groupby(['Opportunity Category', 'Gender'])['Engagement Score'].mean().unstack()

# Stacked bar chart
agg_df.plot(kind='bar', stacked=True, figsize=(12,6), colormap='viridis')
plt.title("Average Engagement Score by Opportunity Category and Gender")
plt.ylabel("Engagement Score")
plt.xticks(rotation=45)
plt.legend(title='Gender')
plt.show()
```

0.3s Python

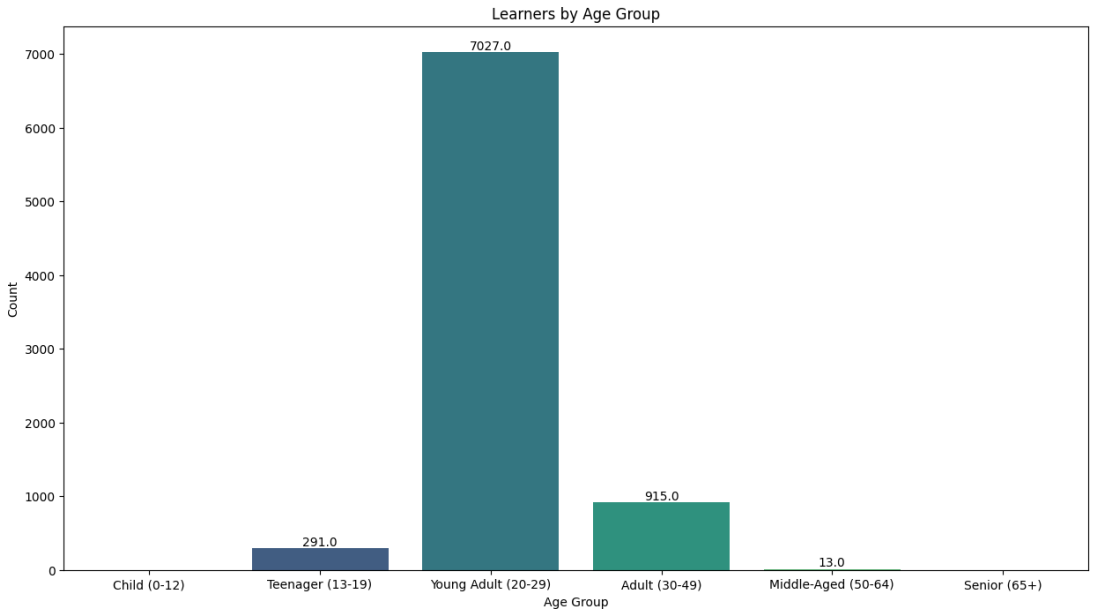


Aspect	Details
Figure	Average Engagement Score by Opportunity Category and Gender (Stacked Bar Chart)
Columns used	Opportunity Category, Gender, Engagement Score
Why used	To show **combined contributions** of male and female learners to the average engagement score across opportunity categories. Stacked bars make it easy to compare totals and subgroup contributions.
How it matters	- Highlights **gender-wise differences** in engagement within each opportunity category. - Helps in **identifying categories where one gender is more engaged**. - Useful for planning **targeted interventions** and improving overall engagement.
Importance	Stacked bar charts provide a **clear visual of subgroup contributions**, making it easier to communicate patterns to stakeholders and guide decision-making.

Learners by Age Group - Bar Chart

```
# Define meaningful age categories
bins = [0, 12, 19, 29, 49, 64, 120]
labels = ['Child (0-12)', 'Teenager (13-19)', 'Young Adult (20-29)',
          'Adult (30-49)', 'Middle-Aged (50-64)', 'Senior (65+)']

df['Age Group'] = pd.cut(df['Age '], bins=bins, labels=labels, right=True)

# Plot count of each age group
plt.figure(figsize=(15,8))
ax = sns.countplot(data=df, x='Age Group', palette='viridis', order=labels)
plt.title("Learners by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Count")

# Annotate counts on top of bars
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height}', xy=(p.get_x() + p.get_width()/2, height),
                ha='center', va='bottom')

plt.show()
```

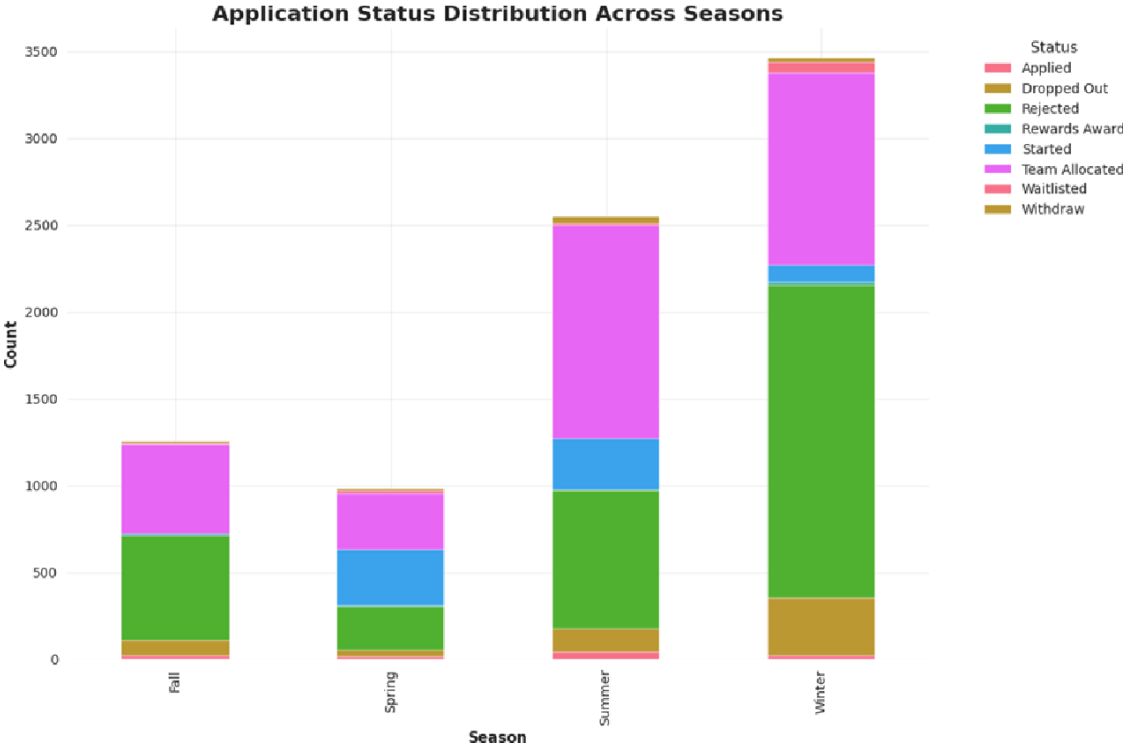
Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	Learners by Age Group (Bar Chart)
Columns used	Age, Age Group (categorized)
Why used	To visualize the **distribution of learners across meaningful age groups**, providing insight into the demographic composition of the dataset.
How it matters	- Identifies which age groups have the **highest or lowest participation**. - Helps in **targeting programs** and designing age-appropriate opportunities. - Useful for understanding **learner demographics** and planning engagement strategies.
Importance	Annotating counts on bars makes it easier to **quantify each group**, which is important for reporting and stakeholder insights.

```python
# Plot 1: Violin Plots for Engagement Score by Status
plt.figure(figsize=(14, 8))
sns.violinplot(x='Status Description', y='Engagement Score', data=df, palette='Set2')
plt.title('Engagement Score Distribution by Application Status', fontsize=16)
plt.xlabel('Status Description')
plt.ylabel('Engagement Score')
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()


# Plot 2: Stacked Bar Chart - Status by Season
status_season = pd.crosstab(df['Seasonal Patterns'], df['Status Description'])
status_season.plot(kind='bar', stacked=True, figsize=(12, 8))
plt.title('Application Status Distribution Across Seasons', fontsize=16, fontweight='bold')
plt.xlabel('Season')
plt.ylabel('Count')
plt.legend(title='Status', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	1. Engagement Score Distribution by Application Status (Violin Plot) 2. 2. Application Status Distribution Across Seasons (Stacked Bar Chart)
Columns used	Status Description, Engagement Score , Seasonal Patterns
Why used	- Violin Plot: To visualize the **distribution and spread** of engagement scores across different application statuses. - Stacked Bar Chart: To show how **application statuses vary across seasons**, highlighting seasonal trends.
How it matters	- Reveals which **statuses are associated with higher or lower engagement**. - Shows **seasonal patterns in applications**, helping plan targeted interventions. - Helps identify **status-specific engagement trends** over time.
Importance	Enables **strategic decision-making** by connecting engagement levels with application statuses and seasonal trends, improving program planning and learner retention.

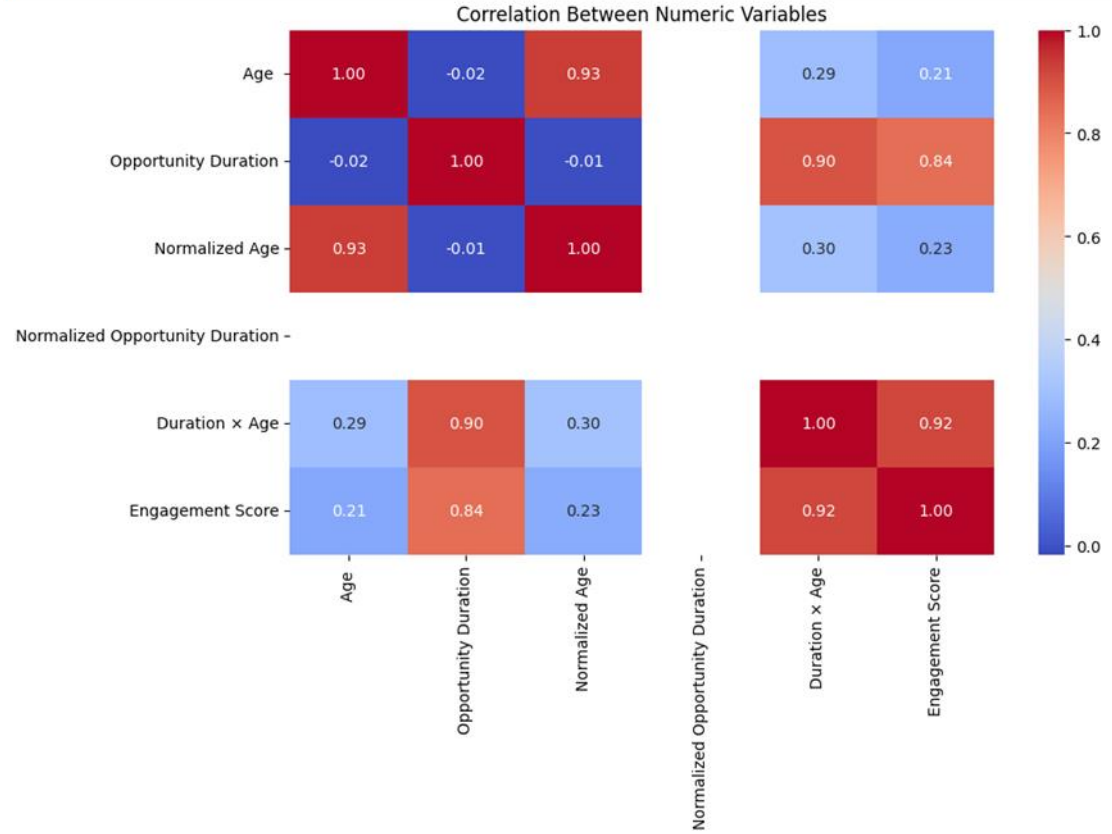
3) Correlation Matrix Analysis:

A correlation matrix was generated to identify relationships between numeric features. The correlation coefficient values range from -1 (strongly negative) to +1 (strongly positive), with values near zero indicating little to no linear relationship.

Correlation Analysis (Numeric Variables)

Heatmaps (Highlight Correlations or Anomalies)

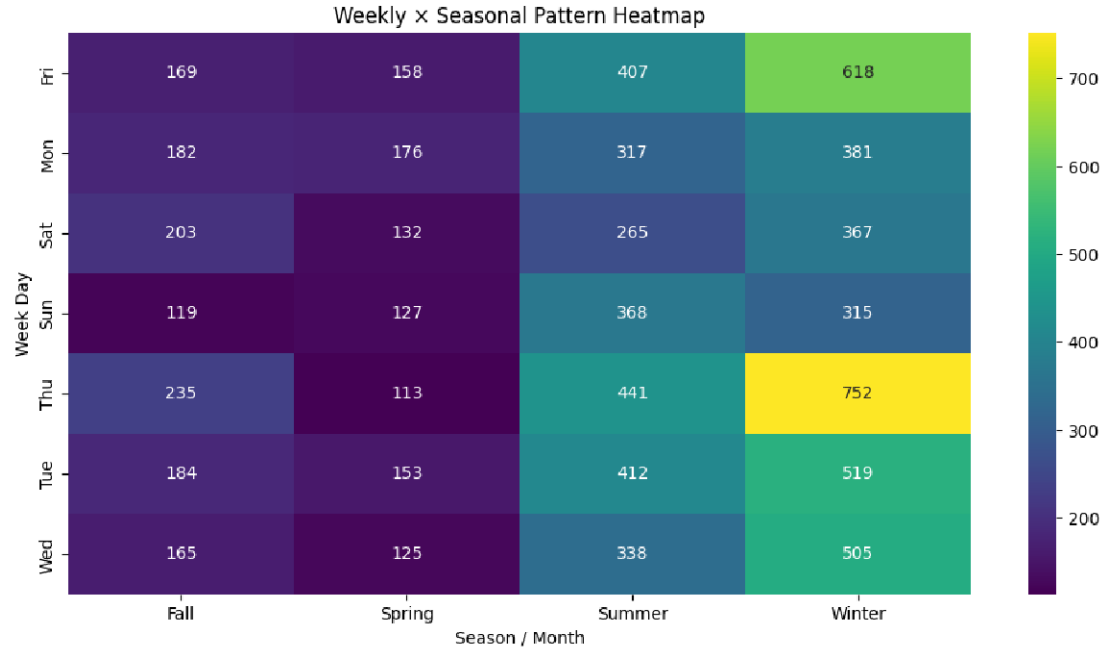
```
# Select numeric columns for correlation
numeric_cols = ["Age ", "Opportunity Duration", "Normalized Age",
                "Normalized Opportunity Duration", "Duration × Age", "Engagement Score"]

# Compute correlation matrix
corr_matrix = df[numeric_cols].corr()

# Plot heatmap
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Between Numeric Variables")
plt.show()
```



Aspect	Details
Figure	Correlation Heatmap of Numeric Variables
Columns used	Age, Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score
Why used	To visually represent relationships and correlations between key numeric features. Helps identify which variables are strongly related or independent, useful for feature analysis and model building.
How it matters	- Understanding correlations guides feature selection and avoids redundancy. - Strong correlations may indicate potential predictors for engagement or opportunity outcomes. - Weak or negative correlations highlight features that contribute unique information.
Importance	Provides an intuitive way to communicate patterns to stakeholders, making it easier to interpret numerical relationships.

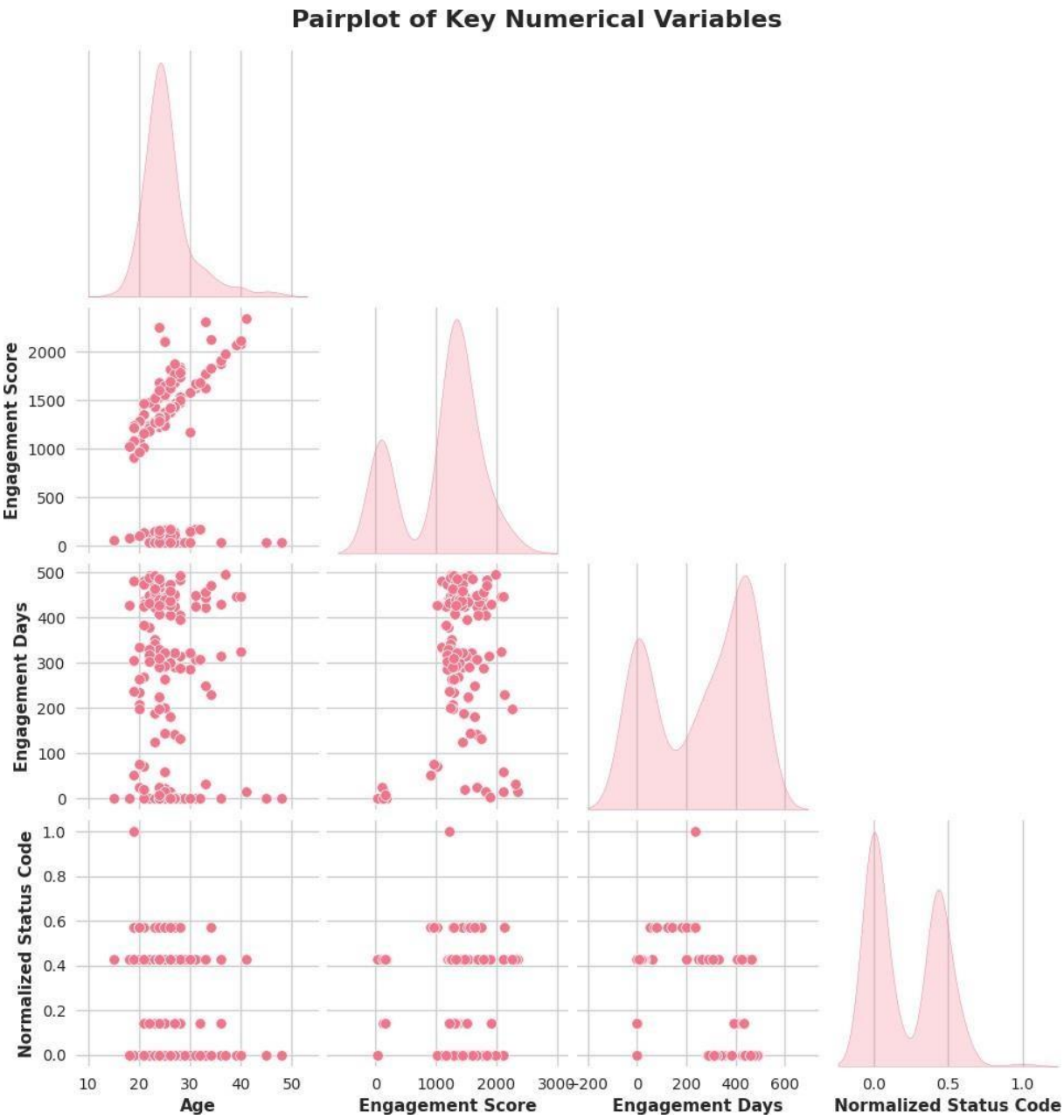
Heatmap → combined view of week × season/month for detailed pattern spotting

```python
# Create a pivot table: index=Weekday, columns=Season/Month, values=counts
pivot = df.pivot_table(index='Weekly Patterns', columns='Seasonal Patterns', aggfunc='size', fill_value=0)

plt.figure(figsize=(12,6))
sns.heatmap(pivot, annot=True, fmt='d', cmap='viridis')
plt.title("Weekly × Seasonal Pattern Heatmap")
plt.ylabel("Week Day")
plt.xlabel("Season / Month")
plt.show()
```



Aspect	Details
Figure	Weekly × Seasonal Pattern Heatmap
Columns used	Weekly Patterns (Weekdays), Seasonal Patterns (Seasons)
Why used	To visualize how opportunities or learner engagements are distributed across weekdays and seasons/months.
How it matters	- Identifies peak days and months for engagement - Helps in planning and scheduling opportunities- Reveals seasonal effects on activity
Importance	Allows quick comparison of daily vs seasonal trends, helping stakeholders understand temporal patterns in the data

```python
# ----------------------------
# 1. Pairplot for key numerical variables
# ----------------------------
numerical_cols = ['Age', 'Engagement Score', 'Engagement Days', 'Normalized Status Code']
pairplot_cols = [col for col in numerical_cols if col in df.columns]

if len(pairplot_cols) > 1:
    print("Creating Pairplot... (this may take a moment)")
    sns.pairplot(df[pairplot_cols], diag_kind='kde', corner=True)
    plt.suptitle('Pairplot of Key Numerical Variables', y=1.02, fontsize=16, fontweight='bold')
    plt.show()
else:
    print("Not enough numerical columns available for Pairplot.")
```



4) Patterns and Correlations:

Signup vs. Completion: While signups fluctuate monthly, completions follow a similar but more subdued trend. There is a clear positive correlation, but the gap between signups and completions remains wide, highlighting a key challenge in converting registered users into successful participants.

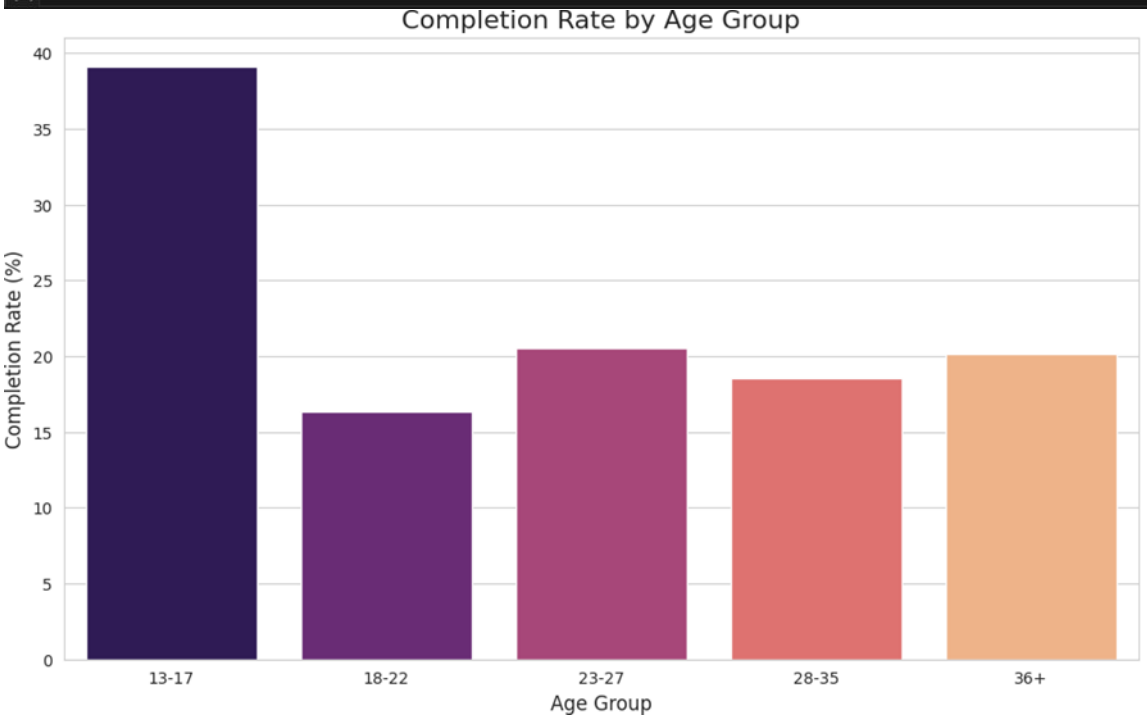
Demographics:

Age: The user base is predominantly young, with a strong peak in the 23-27 age group. This group also exhibits the highest completion rate

Grouping age into discrete bins can help us see non-linear relationships between age and completion.

```
# Define the age bins and labels
age_bins = [13, 17, 22, 27, 35, 60]
age_labels = ['13-17', '18-22', '23-27', '28-35', '36+']

# Create the Age_Group column
df['Age_Group'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels, right=True)

# Visualize the completion rate by the new Age_Group
plt.figure(figsize=(12, 7))
age_group_completion = df.groupby('Age_Group', as_index=False)['Is_Completed'].mean()
age_group_completion['Completion Rate'] = age_group_completion['Is_Completed'] * 100

sns.barplot(x='Age_Group', y='Completion Rate', data=age_group_completion, palette='magma', hue='Age_Group', dodge=False
plt.title('Completion Rate by Age Group', fontsize=16)
plt.xlabel('Age Group', fontsize=12)
plt.ylabel('Completion Rate (%)', fontsize=12)
plt.legend([],[], frameon=False)
plt.show()
```

Activate Windows
Go to Settings to activate Windows.



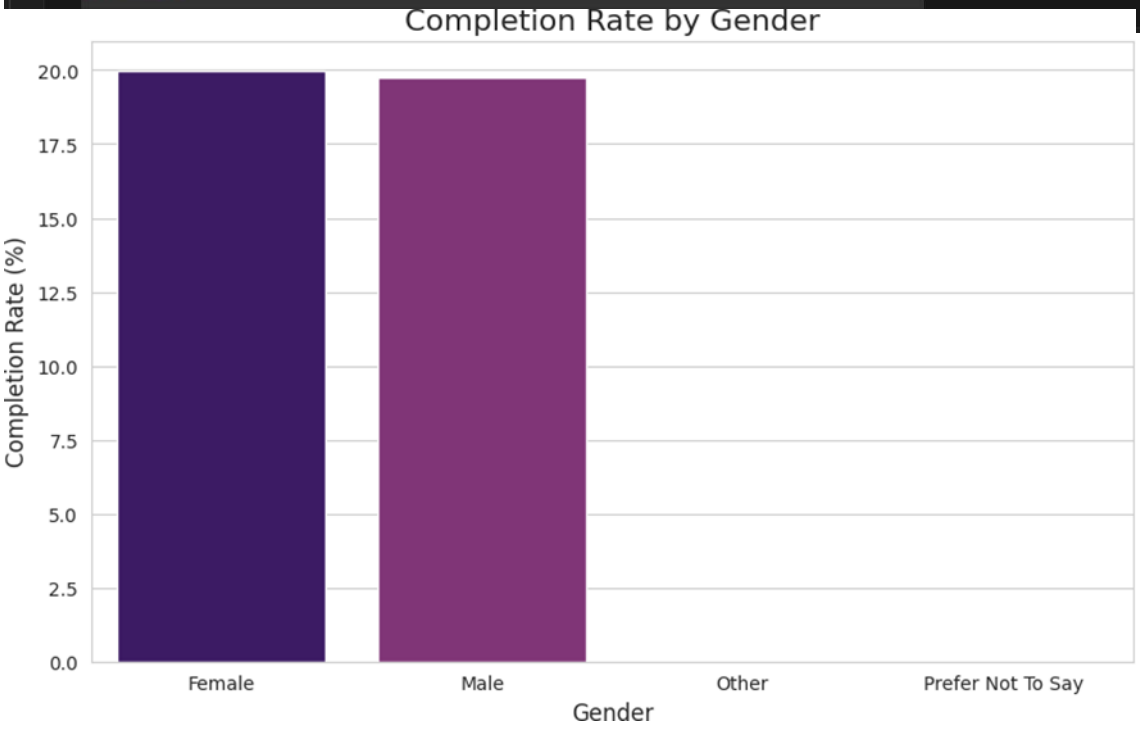
Aspect	Details
Figure	Completion Rate by Age Group -Bar chart
Columns Used	Age, Age_Group, Is_Completed
Why Used	To evaluate how completion varies across different age groups.
How it Matters	Shows which age groups are more likely to complete applications, highlighting user engagement patterns.
Importance	Guides targeted interventions and engagement strategies based on age demographics.

Gender: While there are more male than female users, the ANOVA test (p=0.1125) showed no statistically significant difference in engagement scores between genders. However, completion rates for females are slightly lower.

Here, I check for any gender disparity in completion rates.

```python
plt.figure(figsize=(10, 6))
gender_completion = df.groupby('Gender', as_index=False)['Is_Completed'].mean()
gender_completion['Completion Rate'] = gender_completion['Is_Completed'] * 100

sns.barplot(x='Gender', y='Completion Rate', data=gender_completion.sort_values('Completion Rate', ascending=False), pal
plt.title('Completion Rate by Gender', fontsize=16)
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Completion Rate (%)', fontsize=12)
plt.legend([],[], frameon=False)
plt.show()
```

Python



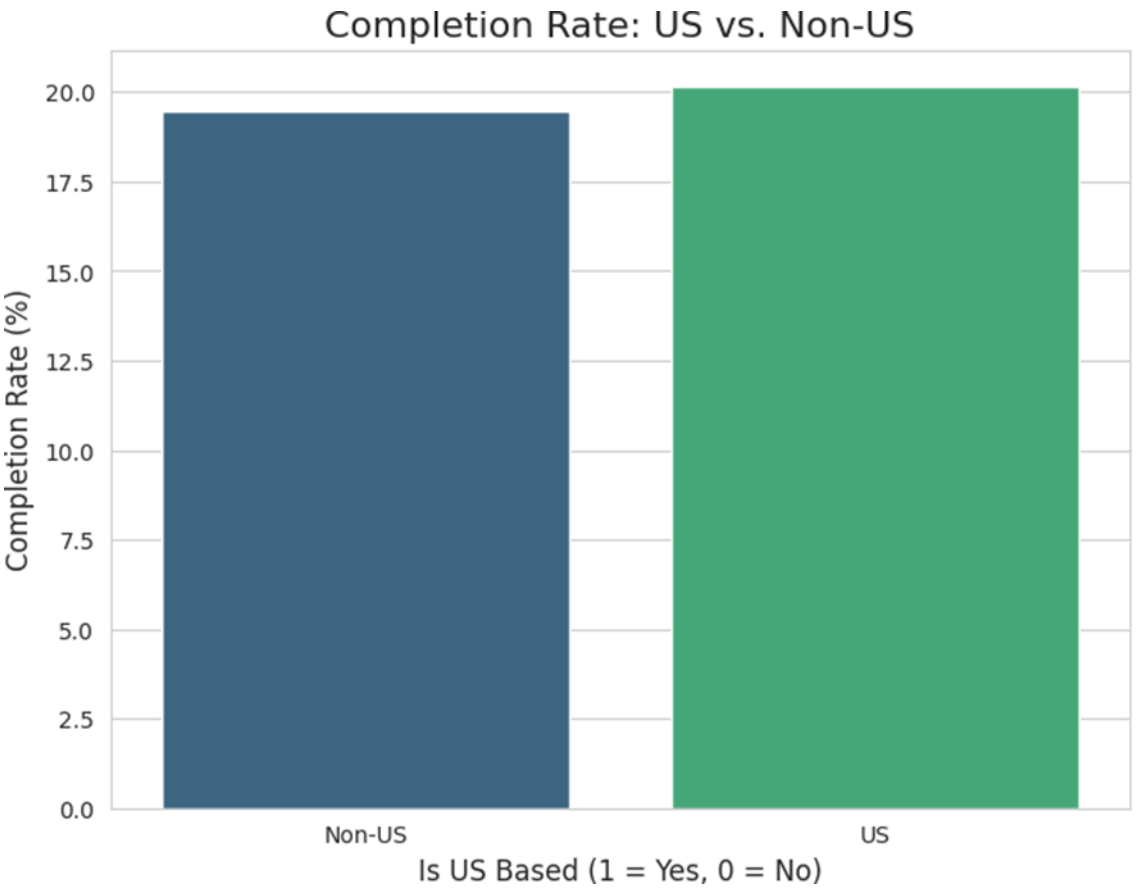
Aspect	Details
Figure	Completion Rate by Gender - Bar chart
Columns Used	Gender, Is_Completed
Why Used	To compare completion rates between different genders.
How it Matters	Highlights gender-based differences in engagement and completion patterns.
Importance	Supports targeted interventions to improve completion rates across genders.

Geography:

The user base is dominated by the United States and India. However, completion rates do not vary significantly between US and non-US users, and there is no strong pattern among the top 10 countries.

```python
# --- Visualize Completion Rate by Is_US_Based ---
plt.figure(figsize=(8, 6))
us_completion = df.groupby('Is_US_Based', as_index=False)['Is_Completed'].mean()
us_completion['Completion Rate'] = us_completion['Is_Completed'] * 100

sns.barplot(x='Is_US_Based', y='Completion Rate', data=us_completion, palette='viridis', hue='Is_US_Based', dodge=False)
plt.title('Completion Rate: US vs. Non-US', fontsize=16)
plt.xlabel('Is US Based (1 = Yes, 0 = No)', fontsize=12)
plt.ylabel('Completion Rate (%)', fontsize=12)
plt.xticks([0, 1], ['Non-US', 'US'])
plt.legend([],[], frameon=False)
plt.show()
```

[47] Python



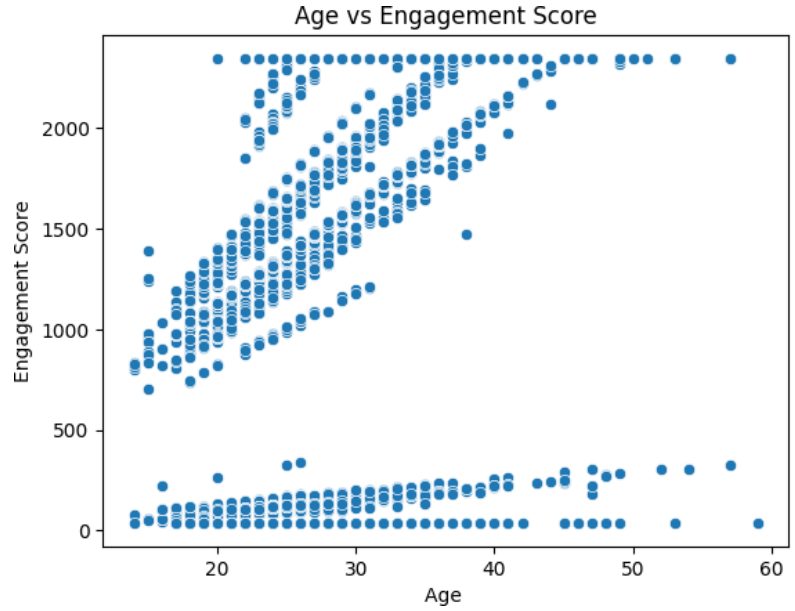
Aspect	Details
Figure	Completion Rate: US vs Non-US - Bar chart
Columns Used	Country, Is_US_Based, Is_Completed
Why Used	To compare completion rates between US-based and non-US learners.
How it Matters	Reveals geographic differences in engagement and completion patterns.
Importance	Helps tailor regional strategies and interventions to improve completion rates.

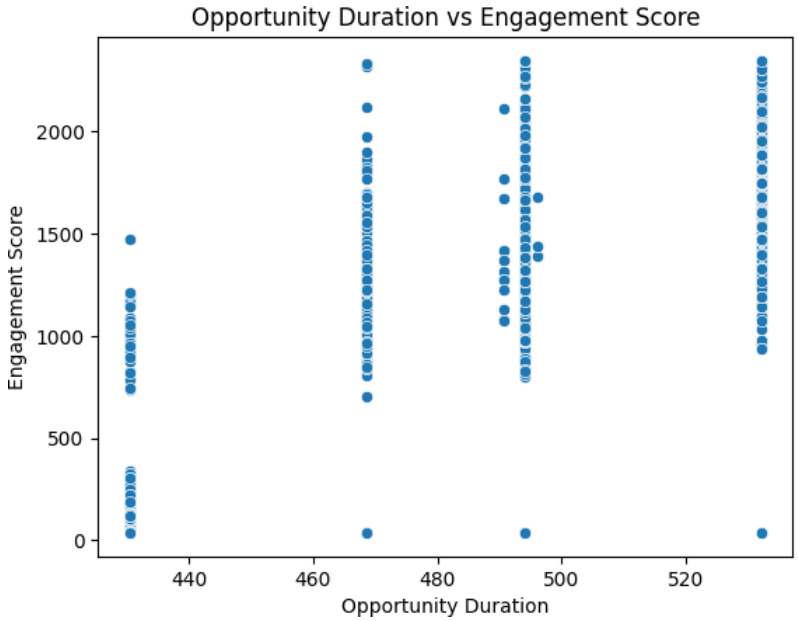
5) Uncovering Relationships Between Variables:

Scatter Plots (Visual Relationships)

```
# Age vs Engagement Score
sns.scatterplot(data=df, x="Age ", y="Engagement Score")
plt.title("Age vs Engagement Score")
plt.show()

# Opportunity Duration vs Engagement Score
sns.scatterplot(data=df, x="Opportunity Duration", y="Engagement Score")
plt.title("Opportunity Duration vs Engagement Score")
plt.show()
```

[12] ✓ 0.3s Python

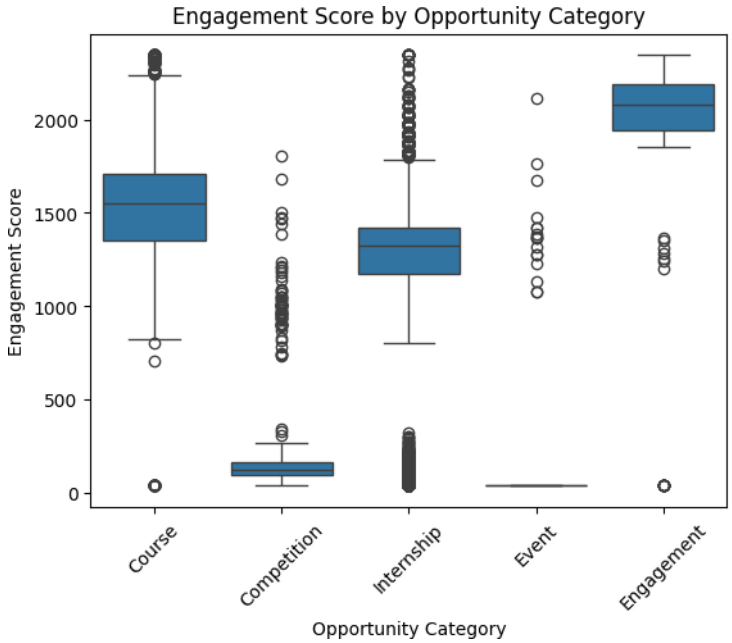




Aspect	Details
Figure	Age vs Engagement Score (Scatterplot) Opportunity Duration vs Engagement Score (Scatterplot)
Columns used	- Age (for first plot) - Engagement Score - Opportunity Duration (for second plot)
Why used	To visually examine the **relationship between age and engagement**, and **opportunity duration and engagement**. Scatterplots help detect trends, patterns, and possible outliers in the data.
How it matters	- Identifies whether age or opportunity duration has an impact on engagement. - Helps understand **learner behavior** in relation to time and demographics. - Useful for **feature selection** in predictive modeling. - Detects clusters, trends, or anomalies in engagement data.
Additional point	These plots allow stakeholders to **quickly see potential influences** on engagement and make informed decisions for planning opportunities.

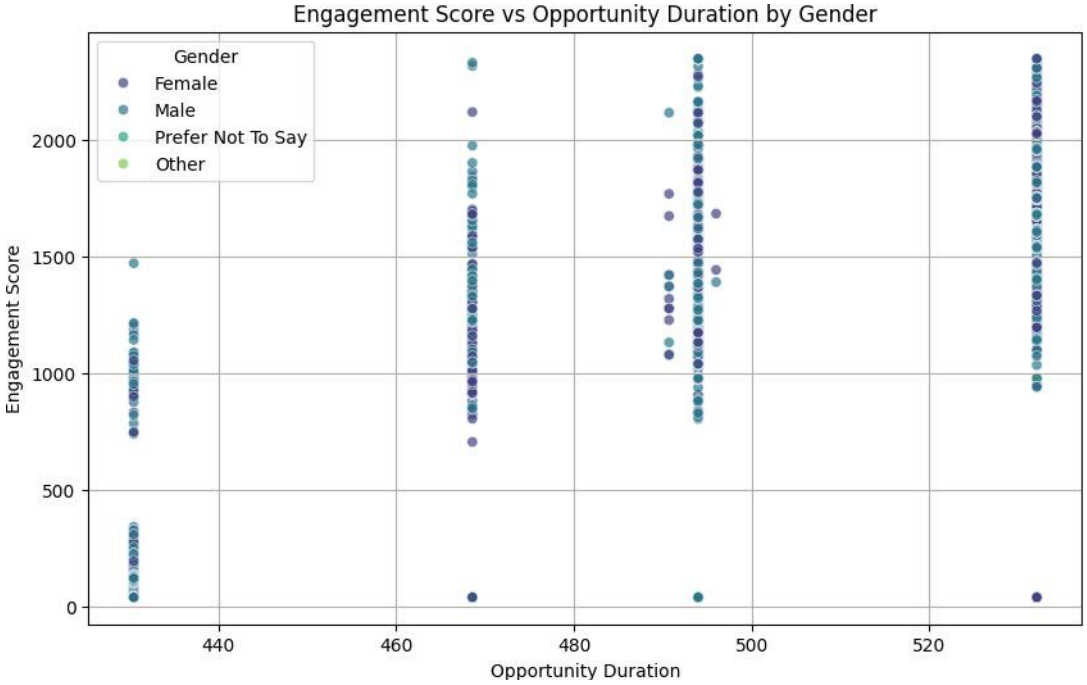
Boxplots (Categorical vs Numeric Relationships)

```
# Engagement Score across Opportunity Categories
sns.boxplot(data=df, x="Opportunity Category", y="Engagement Score")
plt.xticks(rotation=45)
plt.title("Engagement Score by Opportunity Category")
plt.show()
```



Aspect	Details
Figure	Engagement Score by Opportunity Category (Boxplot)
Columns used	Opportunity Category, Engagement Score
Why used	To visualize the **distribution of engagement scores** across different opportunity categories. Boxplots help identify medians, variability, and outliers within each category.
How it matters	- Shows which categories have higher or lower engagement. - Highlights **variability and consistency** of engagement in each category. - Helps in **identifying categories that perform well** or need improvement. - Useful for decision-making and planning targeted interventions.
Importance	Outliers in engagement scores can reveal exceptional learner responses or unusual activity patterns.

```
plt.figure(figsize=(10,6))
sns.scatterplot(
    data=df,
    x='Opportunity Duration',
    y='Engagement Score',
    hue='Gender',
    palette='viridis',
    alpha=0.7
)
plt.title("Engagement Score vs Opportunity Duration by Gender")
plt.xlabel("Opportunity Duration")
plt.ylabel("Engagement Score")
plt.legend(title='Gender')
plt.grid(True)
plt.show()
```

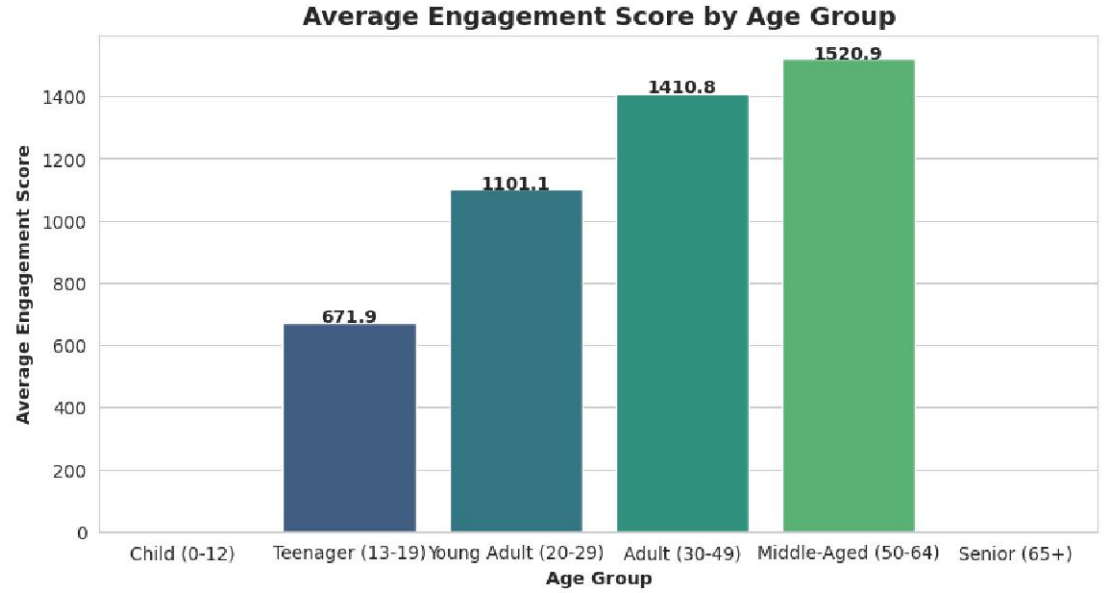


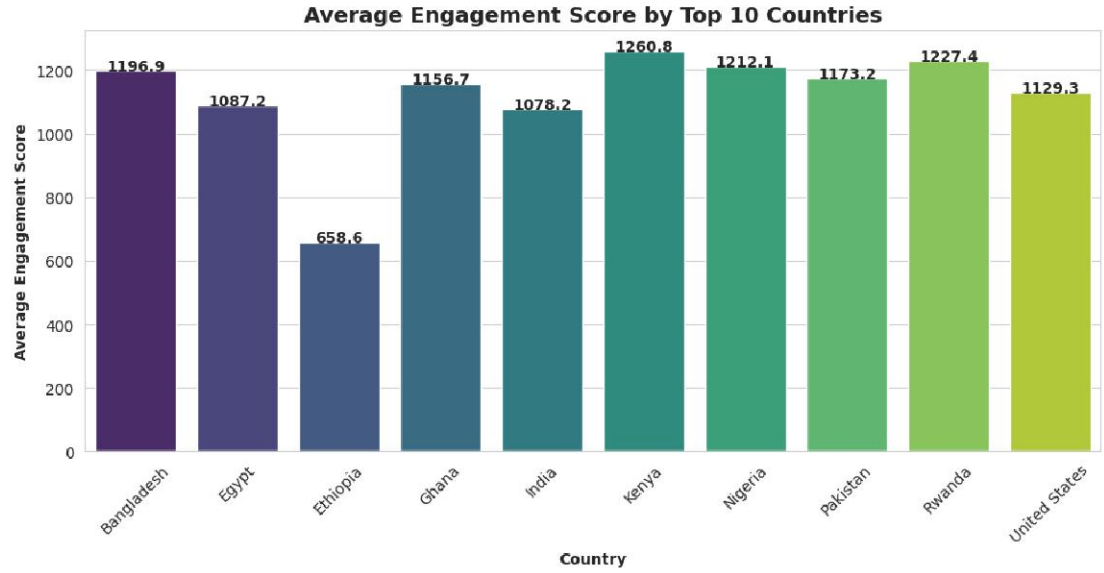
Aspect	Details
Figure	Engagement Score vs Opportunity Duration by Gender (Scatter Plot)
Columns used	- Opportunity Duration - Engagement Score - Gender
Why used	To visualize **how engagement varies with opportunity duration** and whether gender influences this relationship.
How it matters	- Shows **trends and patterns** between opportunity length and engagement.- Highlights **gender-specific differences** in engagement.- Helps identify which opportunities may need targeted interventions based on gender.
Importance	Supports **data-driven insights** for optimizing course duration and engagement strategies, while considering gender differences in participation or outcomes.

```python
# Ensure 'Age Group' exists
if 'Age Group' not in df.columns:
    bins = [0, 12, 19, 29, 49, 64, 120]
    labels = ['Child (0-12)', 'Teenager (13-19)', 'Young Adult (20-29)',
              'Adult (30-49)', 'Middle-Aged (50-64)', 'Senior (65+)']
    df['Age Group'] = pd.cut(df['Age '], bins=bins, labels=labels, right=True)

# -------------------------
# Engagement by Age Group
# -------------------------
age_engagement = df.groupby('Age Group')['Engagement Score'].mean().reindex(labels)

plt.figure(figsize=(10,5))
sns.barplot(x=age_engagement.index, y=age_engagement.values, palette='viridis')
plt.title("Average Engagement Score by Age Group", fontsize=14, fontweight='bold')
plt.xlabel("Age Group", fontweight='bold')
plt.ylabel("Average Engagement Score", fontweight='bold')

# Annotate values
for i, v in enumerate(age_engagement.values):
    plt.text(i, v + 0.5, f"{v:.1f}", ha='center', fontweight='bold')
```

Activate Windows
Go to Settings to activate Windows.

```python
plt.show()

# -------------------------
# Engagement by Country (Top 10)
# -------------------------
top_countries = df['Country'].value_counts().head(10).index
country_engagement = df[df['Country'].isin(top_countries)].groupby('Country')['Engagement Score'].mean()

plt.figure(figsize=(12,5))
sns.barplot(x=country_engagement.index, y=country_engagement.values, palette='viridis')
plt.title("Average Engagement Score by Top 10 Countries", fontsize=14, fontweight='bold')
plt.xlabel("Country", fontweight='bold')
plt.ylabel("Average Engagement Score", fontweight='bold')
plt.xticks(rotation=45)

# Annotate values
for i, v in enumerate(country_engagement.values):
    plt.text(i, v + 0.5, f"{v:.1f}", ha='center', fontweight='bold')

plt.show()
```



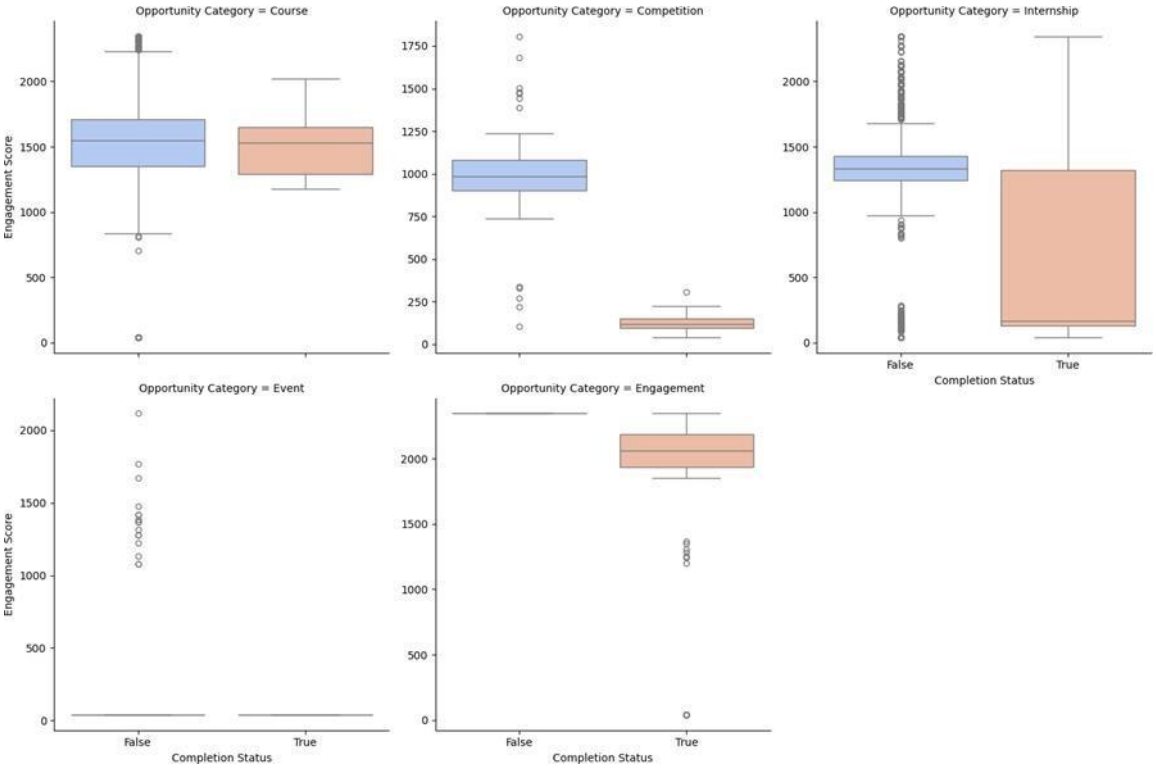


Aspect	Average Engagement by Age Group	Average Engagement by Top 10 Countries
Figure	Bar chart – Average Engagement Score by Age Group	Bar chart – Average Engagement Score by Top 10 Countries
Columns used	Age Group, Engagement Score	Country, Engagement Score
Why used	To measure engagement patterns across different age groups	To measure engagement patterns across top 10 countries
How it matters	Identifies which age brackets are more engaged	Identifies which countries have higher engagement
Importance	Helps target strategies or interventions by age	Helps target strategies or interventions by geography

6) Statistical & Outliers Analysis:

Engagement Score Analysis: *(Statistical)*

By Category: ANOVA results ($p < 0.0001$) confirm a significant difference in engagement scores across opportunity categories. 'Course' opportunities are associated with the highest median engagement scores.

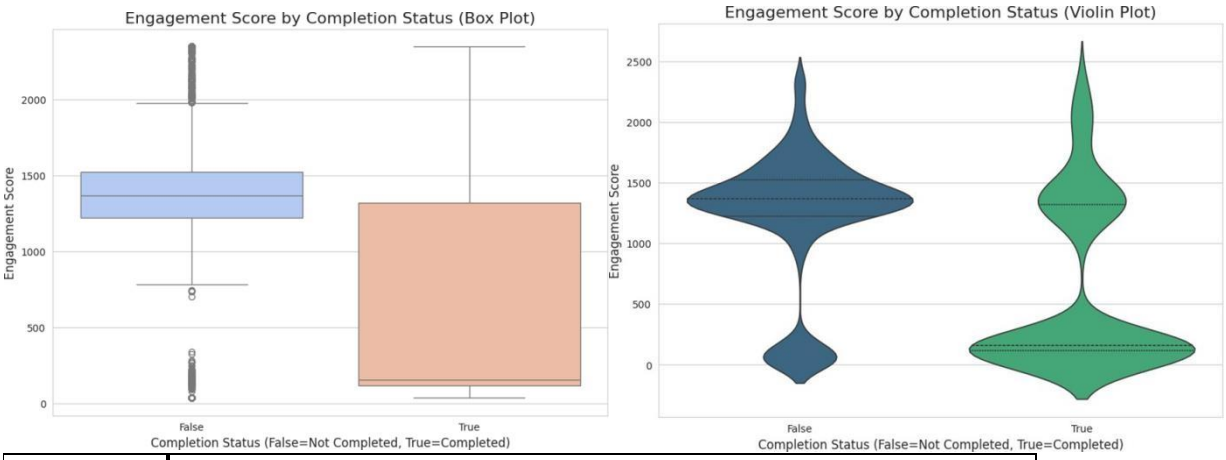


By Completion Status: Surprisingly, users who did *not* complete opportunities have a higher median 'Engagement Score'. This suggests the metric may be rewarding prolonged, unsuccessful effort rather than efficient completion.

A key finding I noted was that learners who completed opportunities had lower average engagement scores. I'll visualize this surprising relationship more clearly using both box plots and violin plots.

```
engagement_metrics = ['Engagement Days', 'Duration × Age', 'Engagement Score']

# Box Plots
for metric in engagement_metrics:
    plt.figure(figsize=(10, 7))
    sns.boxplot(x='Is_Completed', y=metric, data=df, palette='coolwarm')
    plt.title(f'{metric} by Completion Status (Box Plot)', fontsize=16)
    plt.xlabel('Completion Status (False=Not Completed, True=Completed)', fontsize=12)
    plt.ylabel(metric, fontsize=12)
    plt.show()

# Violin Plots
for metric in engagement_metrics:
    plt.figure(figsize=(10, 7))
    sns.violinplot(x='Is_Completed', y=metric, data=df, palette='viridis', inner='quartile')
    plt.title(f'{metric} by Completion Status (Violin Plot)', fontsize=16)
    plt.xlabel('Completion Status (False=Not Completed, True=Completed)', fontsize=12)
    plt.ylabel(metric, fontsize=12)
    plt.show()
```

Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	Engagement Metrics by Completion Status (Box & Violin Plots)
Columns Used	Is_Completed, Engagement Days, Duration × Age, Engagement Score
Why Used	To compare engagement behaviors between completed and non-completed learners
How it Matters	Shows distribution, spread, and central tendency differences for key engagement metrics
Importance	Helps identify engagement patterns that may predict completion and inform interventions

Outliers and Anomalies Analysis: *(Outliers)*

Outliers are data points that deviate significantly from the norm, possibly indicating errors, unique cases, or significant anomalies.

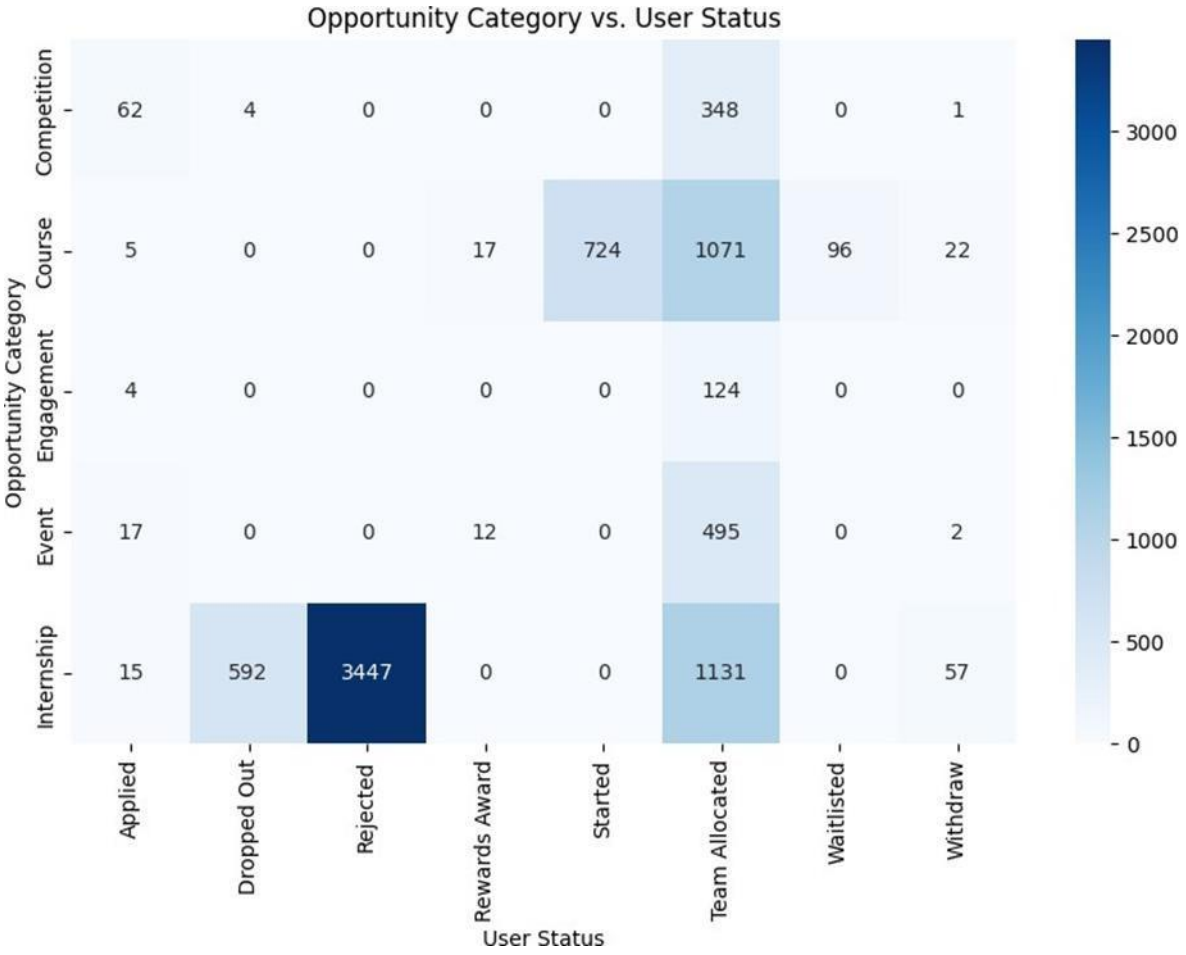
Completion Time Outliers: The 'Time_to_Apply' metric has significant outliers, with some users taking over a year to apply. These users are almost exclusively in the non-completed group.

Low Completion Days & Categories: The Chi-squared test ($p < 0.0001$) confirms a strong association between 'Opportunity Category' and 'User Status'. The most significant anomaly is in the 'Internship' category, which accounts for a staggering 3,447 'Rejected' users. This is the single largest bottleneck in the user journey.

A matrix showing counts of users for each combination of Opportunity Category (rows) and Status Description (columns). This is essentially a cross-tabulation heatmap.

```
# Ensure the columns exist
if 'Opportunity Category' in df.columns and 'Status Description' in df.columns:

    # Create a cross-tabulation table
    category_status_matrix = pd.crosstab(df['Opportunity Category'], df['Status Description'])

    # Plot heatmap
    plt.figure(figsize=(14, 8))
    sns.heatmap(category_status_matrix, annot=True, fmt='d', cmap='YlGnBu', cbar=True,
                linewidths=0.5, linecolor='gray')
    plt.title('User Status by Opportunity Category', fontsize=16, fontweight='bold')
    plt.xlabel('Status Description', fontsize=12, fontweight='bold')
    plt.ylabel('Opportunity Category', fontsize=12, fontweight='bold')
    plt.xticks(rotation=45, ha='right')
    plt.yticks(rotation=0)
    plt.tight_layout()
    plt.show()
```

Activate Windows



Aspect	Details
Figure	User Status by Opportunity Category (Heatmap)
Columns Used	Opportunity Category, Status Description
Why Used	To visualize the distribution of user statuses across opportunity types
How it Matters	Highlights which opportunity categories have higher or lower completion or other status types
Importance	Helps prioritize opportunities for engagement improvements or targeted interventions

Few important column's Outliers Detection:

```python
# OUTLIER DETECTION AND ANALYSIS - With Borders
# Detect outliers using IQR method
def detect_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data < lower_bound) | (data > upper_bound)]
    return outliers

# Check for outliers in key numerical columns
outlier_analysis = {}
for col in ['Age', 'Engagement Score', 'Engagement Days']:
    outliers = detect_outliers_iqr(df[col])
    outlier_analysis[col] = {
        'count': len(outliers),
        'percentage': (len(outliers) / len(df)) * 100,
        'values': outliers.values
    }
    print(f"{col}: {outlier_analysis[col]['count']} outliers ({outlier_analysis[col]['percentage']:.2f}%)")
```

Activate Windows
Go to Settings to activate Windows.

```python
# Visualize outliers with borders
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle('OUTLIER DETECTION ANALYSIS', fontsize=16, fontweight='bold', y=1.05)

# Plot 1: Age Outliers
ax1 = axes[0]
sns.boxplot(y=df['Age'], ax=ax1, color='#ff6b6b')
ax1.set_title('Age - Outlier Detection', fontsize=14, fontweight='bold', pad=10)
ax1.set_ylabel('Age', fontweight='bold')
# Add border to Age plot
for spine in ax1.spines.values():
    spine.set_edgecolor('#e0e0e0')
    spine.set_linewidth(2)

# Plot 2: Engagement Score Outliers
ax2 = axes[1]
sns.boxplot(y=df['Engagement Score'], ax=ax2, color='#4ecdc4')
ax2.set_title('Engagement Score - Outlier Detection', fontsize=14, fontweight='bold', pad=10)
ax2.set_ylabel('Engagement Score', fontweight='bold')
# Add border to Engagement Score plot
for spine in ax2.spines.values():
    spine.set_edgecolor('#e0e0e0')
    spine.set_linewidth(2)
```

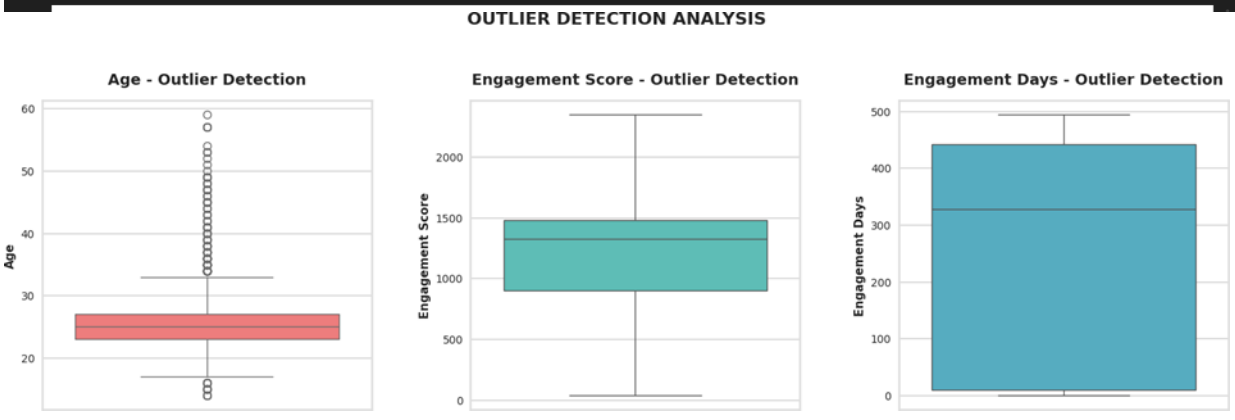
Activate Windows
Go to Settings to activate Windows.

```python
# Plot 3: Engagement Days Outliers
ax3 = axes[2]
sns.boxplot(y=df['Engagement Days'], ax=ax3, color='#45b7d1')
ax3.set_title('Engagement Days - Outlier Detection', fontsize=14, fontweight='bold', pad=10)
ax3.set_ylabel('Engagement Days', fontweight='bold')
# Add border to Engagement Days plot
for spine in ax3.spines.values():
    spine.set_edgecolor('#e0e0e0')
    spine.set_linewidth(2)

plt.tight_layout()
plt.show()
```

0.4s Python

```
Age: 470 outliers (5.70%)
Engagement Score: 0 outliers (0.00%)
Engagement Days: 0 outliers (0.00%)
```

Activate Windows
Go to Settings to activate Windows.



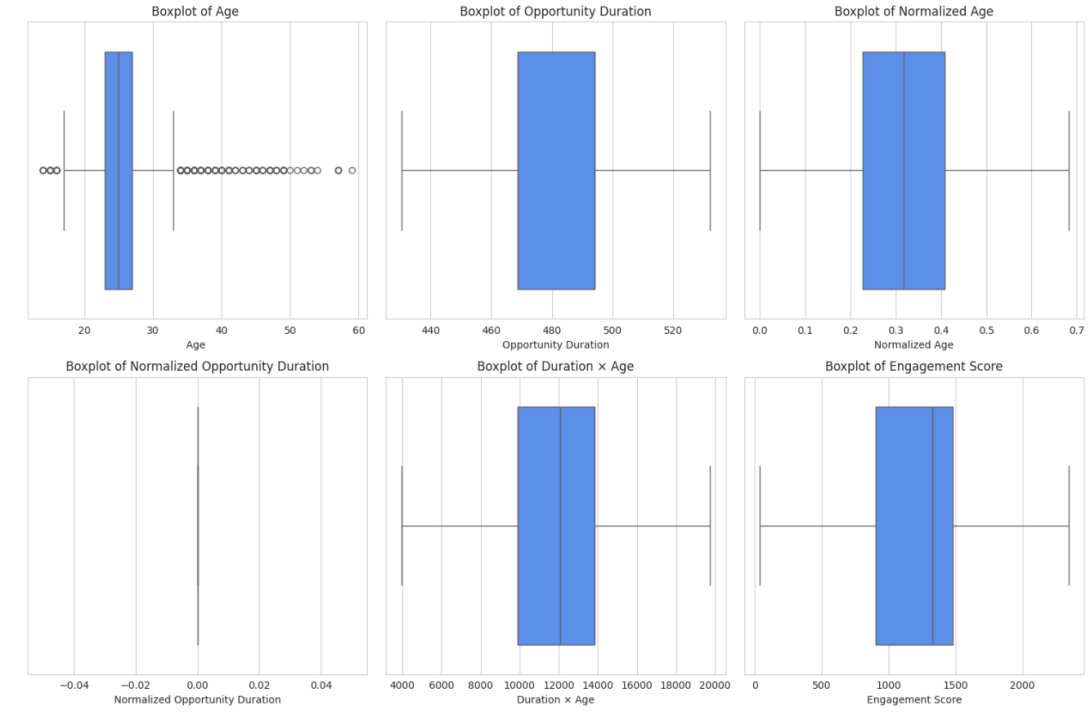
Aspect	Age Outliers	Engagement Score Outliers	Engagement Days Outliers
Figure	Boxplot with border	Boxplot with border	Boxplot with border
Columns Used	Age	Engagement Score	Engagement Days
Why Used	To detect unusually low or high ages in dataset	To detect unusually low or high engagement scores	To detect unusually low or high engagement days
How it Matters	Identifies learners outside typical age range	Identifies extreme engagement behaviors	Identifies learners with abnormal engagement duration
Importance	Helps handle anomalies or segment analysis by age	Supports cleaner modeling and reliable insights	Supports cleaner modeling and reliable insights

Global Outliers: Far from all other points.

Statistical & Outliers Analysis

```python
# Numeric columns to check
numeric_cols = ["Age ", "Opportunity Duration", "Normalized Age",
                "Normalized Opportunity Duration", "Duration x Age", "Engagement Score"]

# ------------------------------
# 1. Z-Score Method
# ------------------------------
print("🔹 Z-Score Outliers")
z_scores = df[numeric_cols].apply(zscore, nan_policy='omit')
z_outliers = (np.abs(z_scores) > 3)
for col in numeric_cols:
    count = z_outliers[col].sum()
    print(f"{col}: {count} potential outliers (|Z| > 3)")

# ------------------------------
# 2. IQR Method
# ------------------------------
print("\n🔹 IQR Outliers")
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers = df[(df[col] < Q1 - 1.5*IQR) | (df[col] > Q3 + 1.5*IQR)]
    print(f"{col}: {len(outliers)} potential outliers (IQR method)")

# ------------------------------
# 3. Boxplots for Visualization
# ------------------------------
plt.figure(figsize=(15,10))
for i, col in enumerate(numeric_cols):
    plt.subplot(2, 3, i+1)
    sns.boxplot(x=df[col], color="#4589ff")
    plt.title(f"Boxplot of {col}")
plt.tight_layout()
plt.show()
```

```
🔹 Z-Score Outliers
Age : 145 potential outliers (|Z| > 3)
Opportunity Duration: 0 potential outliers (|Z| > 3)
Normalized Age: 0 potential outliers (|Z| > 3)
Normalized Opportunity Duration: 0 potential outliers (|Z| > 3)
Duration × Age: 0 potential outliers (|Z| > 3)
Engagement Score: 0 potential outliers (|Z| > 3)

🔹 IQR Outliers
Age : 470 potential outliers (IQR method)
Opportunity Duration: 0 potential outliers (IQR method)
Normalized Age: 0 potential outliers (IQR method)
Normalized Opportunity Duration: 0 potential outliers (IQR method)
Duration × Age: 0 potential outliers (IQR method)
Engagement Score: 0 potential outliers (IQR method)
```



Aspect	Outlier Detection in Numeric Columns
Figure	Boxplots for each numeric column
Columns used	Age , Opportunity Duration, Normalized Age, Normalized Opportunity Duration, Duration × Age, Engagement Score
Why used	To detect unusual or extreme values in numeric features that could skew analysis or model performance
How it matters	Identifies potential data errors, extreme behaviors, or edge cases; informs decisions about data cleaning, transformation, or robust modeling
Importance	Ensures data quality, prevents biased statistical results, and improves reliability of downstream analyses or predictive models

Contextual Outliers: Outliers in specific contexts.

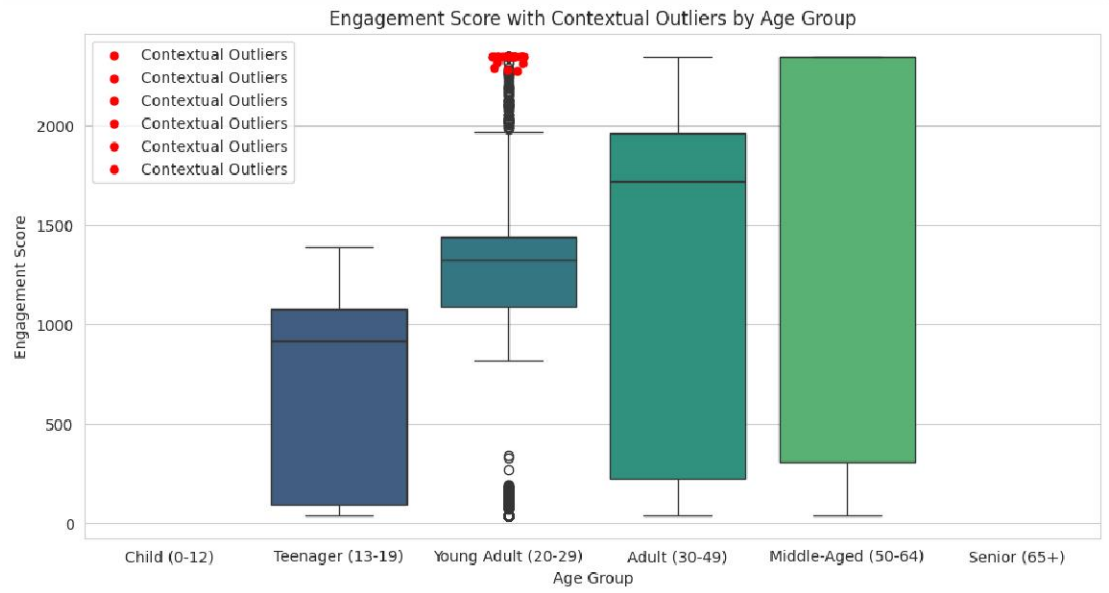
Contextual Outliers:

```
# Make sure 'Age Group' exists
if 'Age Group' not in df.columns:
    bins = [0, 12, 19, 29, 49, 64, 120]
    labels = ['Child (0-12)', 'Teenager (13-19)', 'Young Adult (20-29)',
              'Adult (30-49)', 'Middle-Aged (50-64)', 'Senior (65+)']
    df['Age Group'] = pd.cut(df['Age '], bins=bins, labels=labels, right=True)

# Context: Engagement Score within each Age Group
contextual_outliers = pd.DataFrame()

# Using Z-Score threshold within each group
z_threshold = 2  # can adjust 2 or 3 standard deviations

for group, data in df.groupby('Age Group'):
    if len(data) > 1:  # avoid division by zero
        group_mean = data['Engagement Score'].mean()
        group_std = data['Engagement Score'].std()
        z_scores = (data['Engagement Score'] - group_mean) / group_std
        outliers = data[np.abs(z_scores) > z_threshold]
        contextual_outliers = pd.concat([contextual_outliers, outliers])
```

Activate Windows
Go to Settings to activate Windows.

```
print(f"Total contextual outliers (by Age Group): {len(contextual_outliers)}")
contextual_outliers[['Age ', 'Age Group', 'Engagement Score']]
plt.figure(figsize=(12,6))
sns.boxplot(x='Age Group', y='Engagement Score', data=df, palette='viridis')
sns.stripplot(x='Age Group', y='Engagement Score', data=contextual_outliers,
              color='red', size=6, jitter=True, label='Contextual Outliers')
plt.title('Engagement Score with Contextual Outliers by Age Group')
plt.ylabel('Engagement Score')
plt.xlabel('Age Group')
plt.legend()
plt.show()
```

0.7s Python



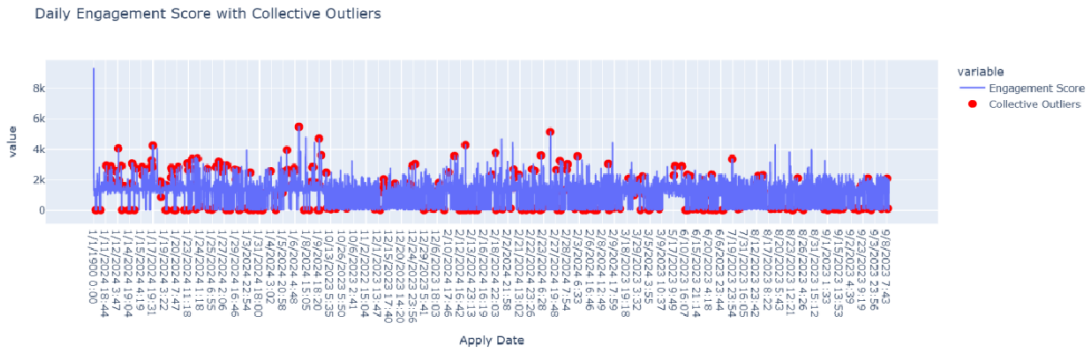
Aspect	Details
Figure	Contextual Outliers in Engagement Score by Age Group (Boxplot with overlaid strip plot highlighting outliers)
Columns used	Age , Age Group, Engagement Score
Why used	To detect unusually high or low engagement scores within each age group context, rather than across the whole dataset

Aspect	Details
How it matters	Identifies users whose engagement is atypical for their age bracket, enabling targeted interventions or deeper analysis
Importance	Improves understanding of age-specific engagement patterns and helps prevent skewed insights caused by extreme values

Collective Outliers: Groups that deviate collectively.

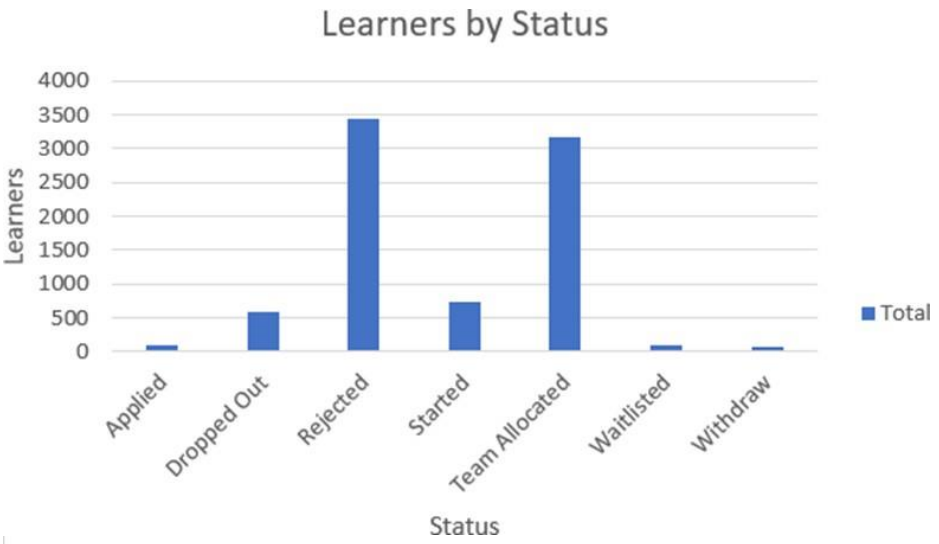
```
# Aggregate Engagement by Apply Date
daily_engagement = df.groupby('Apply Date')['Engagement Score'].sum()

# Compute rolling mean and standard deviation
rolling_mean = daily_engagement.rolling(7, center=True).mean()
rolling_std = daily_engagement.rolling(7, center=True).std()

# Flag collective outliers: points far from rolling mean (e.g., >2 std)
collective_outliers = daily_engagement[(daily_engagement - rolling_mean).abs() > 2*rolling_std]

print("Collective outliers (spikes in daily engagement):")
print(collective_outliers)


# Collective outliers line chart
fig = px.line(daily_engagement, title='Daily Engagement Score with Collective Outliers')
fig.add_scatter(x=collective_outliers.index, y=collective_outliers.values, mode='markers',
                name='Collective Outliers', marker=dict(color='red', size=10))
fig.show()
```

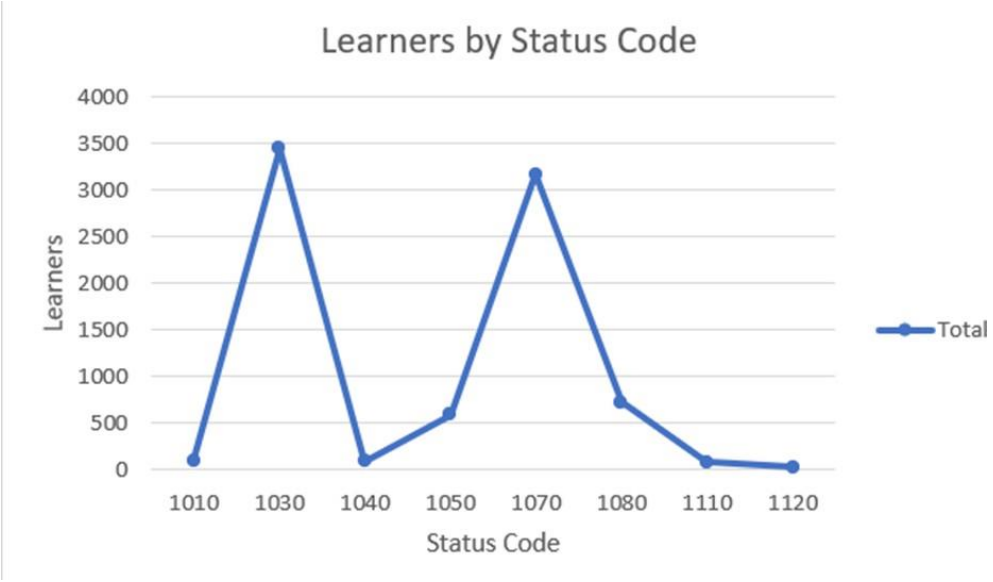
```
Collective outliers (spikes in daily engagement):
Apply Date
1/1/2024 22:44       37.893238
1/1/2024 5:21        37.893238
1/10/2024 3:44       37.893238
1/11/2024 20:17    2938.729542
1/11/2024 6:18     2591.194847
                       ...
9/28/2023 21:27      37.893238
9/6/2023 19:38      120.450330
9/9/2023 14:57     2103.979424
9/9/2023 21:58      121.379493
9/9/2023 22:38      141.068969
Name: Engagement Score, Length: 283, dtype: float64
```



Aspect	Daily Engagement Score with Collective Outliers
Figure	Line Chart with Outlier Markers
Columns used	Apply Date, Engagement Score
Why used	To visualize daily engagement trends and highlight unusual spikes that deviate significantly from the 7-day rolling mean
How it matters	Helps detect abnormal activity patterns, potential anomalies, or events causing engagement surges
Importance	Supports anomaly detection, data quality checks, and operational or strategic decisions to address or investigate spikes

7) Critical Anomaly: *(Visualized in Excel)*





Aspect	Learners by Status (Bar Chart)	Learners by Status Code (Line Chart)
Figure	Bar chart showing learner counts per Status Description	Line chart showing learner counts across numeric Status Codes
Columns used	Status Description	Status Code
Why used	To visualize **categorical distribution** of learner statuses (e.g., Completed, Dropped, In-progress)	To visualize **numeric patterns** in learner outcomes and identify trends or anomalies
How it matters	Highlights where users are **progressing or dropping off**, useful for identifying bottlenecks	Helps spot **irregularities or critical anomalies** in coded data; supports mapping codes to descriptions
Importance	Identifies areas needing **intervention** to improve completion rates	Supports **quantitative monitoring**, automated checks, and anomaly detection

8) Trend Analysis and Pattern Identification:

TREND ANALYSIS AND PATTERN IDENTIFICATION

```
# Set style
sns.set_style("whitegrid")
plt.rcParams['font.family'] = 'DejaVu Sans'

# Convert dates
if 'Learner SignUp DateTime' in df.columns:
    df['Learner SignUp DateTime'] = pd.to_datetime(df['Learner SignUp DateTime'])
    df['SignUp_Month'] = df['Learner SignUp DateTime'].dt.to_period('M')

# Define color palette
colors = {
    'blue': '#4589ff',
    'turquoise': '#33b1ff'
}

# Create figure
fig, axes = plt.subplots(2, 1, figsize=(14, 10))
fig.patch.set_facecolor('#fafafa')

# ------------------ Plot 1: Monthly Sign-ups ------------------
ax1 = axes[0]
```

Activate Windows
Go to Settings to activate Windows.

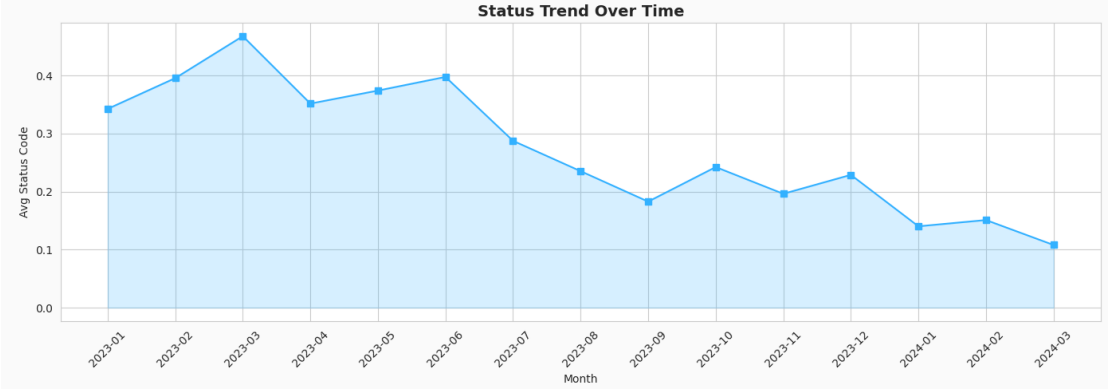
```
if 'SignUp_Month' in df.columns:
    monthly_trends = df['SignUp_Month'].value_counts().sort_index()
    ax1.plot(monthly_trends.index.astype(str), monthly_trends.values, marker='o', color=colors['blue'])
    ax1.fill_between(monthly_trends.index.astype(str), monthly_trends.values, alpha=0.2, color=colors['blue'])
    ax1.set_title('Monthly Sign-up Trends', fontsize=14, fontweight='bold')
    ax1.set_xlabel('Month')
    ax1.set_ylabel('Number of Sign-ups')
    ax1.tick_params(axis='x', rotation=45)
else:
    ax1.text(0.5, 0.5, 'No Sign-up Data', ha='center', va='center')

# ------------------ Plot 2: Status Trend ------------------
ax2 = axes[1]
if 'SignUp_Month' in df.columns and 'Normalized Status Code' in df.columns:
    status_trend = df.groupby('SignUp_Month')['Normalized Status Code'].mean()
    ax2.plot(status_trend.index.astype(str), status_trend.values, marker='s', color=colors['turquoise'])
    ax2.fill_between(status_trend.index.astype(str), status_trend.values, alpha=0.2, color=colors['turquoise'])
    ax2.set_title('Status Trend Over Time', fontsize=14, fontweight='bold')
    ax2.set_xlabel('Month')
    ax2.set_ylabel('Avg Status Code')
    ax2.tick_params(axis='x', rotation=45)
else:
    ax2.text(0.5, 0.5, 'No Status Data', ha='center', va='center')
```

Activate Windows
Go to Settings to activate Windows.

```
plt.tight_layout()
plt.show()
```

0.4s Python

```
📈 TREND ANALYSIS AND PATTERN IDENTIFICATION
============================================================
• Monthly sign-ups trend: -36.6% change over 15 months
• Highest engagement age group: 35-40 (1609.5)
• Top 3 countries by engagement:
   - Falkland Islands: 2036.4
   - Mauritius: 1943.1
   - Namibia: 1797.0
```



Aspect	Monthly Sign-up Trends	Status Trend Over Time
Figure	Line chart with markers and shaded area	Line chart with markers and shaded area
Columns Used	Learner SignUp DateTime / Exctracted SignUp Month	Extracted SignUp Month, Normalized Status Code
Why Used	To track the number of sign-ups per month	To track changes in average status code over time
How it Matters	Shows growth patterns, fluctuations, and seasonality	Reveals trends in user progress or engagement over time
Importance	Helps identify peak signup periods and plan campaigns	Helps monitor user status progression and spot anomalies

```python
# Define color palette
colors = {
    'purple': '#be95ff',
    'yellow': '#fa4d56'
}

# Create figure
fig, axes = plt.subplots(2, 1, figsize=(14, 10))
fig.patch.set_facecolor('#fafafa')

# ------------------ Plot 1: Engagement by Age Group ------------------
ax1 = axes[0]
if 'Age ' in df.columns and 'Engagement Score' in df.columns:
    bins = [0, 20, 25, 30, 35, 40, 100]
    labels = ['<20','20-25','25-30','30-35','35-40','40+']
    df['Age_Group'] = pd.cut(df['Age '], bins=bins, labels=labels)
    age_engagement = df.groupby('Age_Group')['Engagement Score'].mean()
    bars = ax1.bar(age_engagement.index, age_engagement.values, color=colors['purple'], alpha=0.8)
    ax1.set_title('Engagement by Age Group', fontsize=14, fontweight='bold')
    ax1.set_xlabel('Age Group')
    ax1.set_ylabel('Avg Engagement')
    for bar, val in zip(bars, age_engagement.values):
        ax1.text(bar.get_x() + bar.get_width()/2, val+0.5, f'{val:.1f}', ha='center', va='bottom')
else:
    ax1.text(0.5, 0.5, 'No Age/Engagement Data', ha='center', va='center')

# ------------------ Plot 2: Engagement by Top Countries ------------------
ax2 = axes[1]
if 'Country' in df.columns and 'Engagement Score' in df.columns:
    top_countries = df.groupby('Country')['Engagement Score'].mean().sort_values(ascending=False).head(6)
    bars = ax2.bar(top_countries.index, top_countries.values, color=colors['yellow'], alpha=0.8)
    ax2.set_title('Engagement by Top Countries', fontsize=14, fontweight='bold')
    ax2.set_xlabel('Country')
    ax2.set_ylabel('Avg Engagement')
    ax2.tick_params(axis='x', rotation=45)
    for bar, val in zip(bars, top_countries.values):
        ax2.text(bar.get_x() + bar.get_width()/2, val+0.5, f'{val:.1f}', ha='center', va='bottom')
else:
    ax2.text(0.5, 0.5, 'No Country/Engagement Data', ha='center', va='center')

plt.tight_layout()
plt.show()
```

Activate Windows
Go to Settings to activate Windows.

Aspect	Engagement by Age Group	Engagement by Top Countries
Figure	Bar chart – Avg Engagement Score by Age Group	Bar chart – Avg Engagement Score by Top Countries
Columns Used	Age_Group, Engagement Score	Country, Engagement Score
Why Used	To analyze engagement patterns across age brackets	To analyze engagement patterns across top countries

Aspect	Engagement by Age Group	Engagement by Top Countries
How it Matters	Identifies which age groups are more engaged	Identifies countries with higher engagement
Importance	Helps target strategies or interventions by age	Helps target strategies or interventions geographically

9) Time Series Analysis:

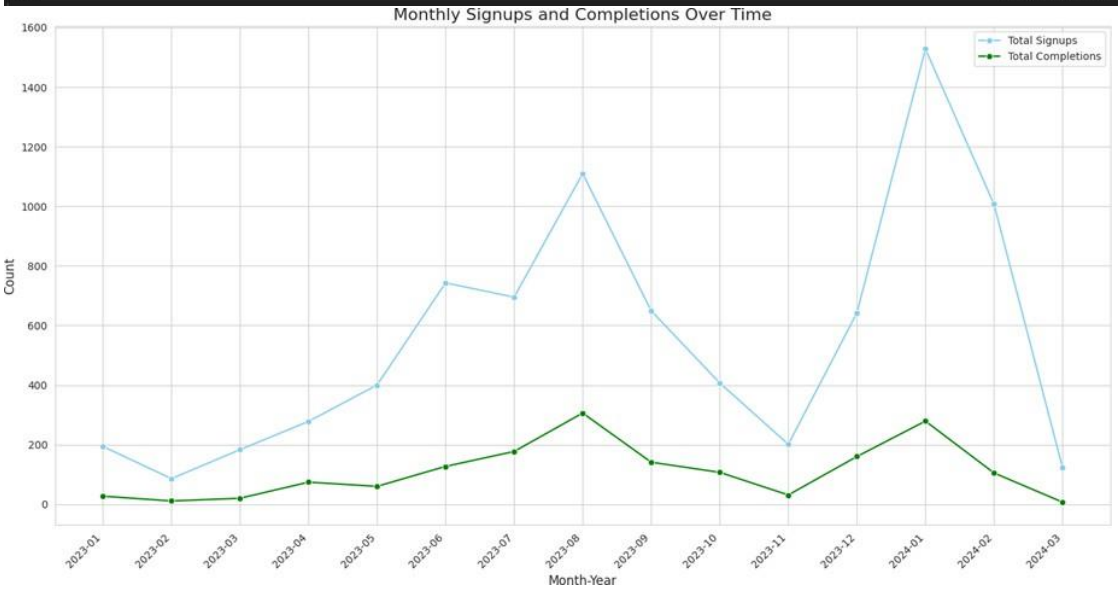
Completion Trends:

Analysis Goals: The primary objective of this analysis is to understand the key trends and drivers behind user signups and completions. By examining user demographics, engagement patterns, and opportunity types, we aim to identify actionable insights to improve completion rates and overall platform success.

I examine how signups and completions have trended over time.

```
df['Signup_YearMonth'] = df['Learner SignUp DateTime'].dt.to_period('M')
monthly_trends = df.groupby('Signup_YearMonth', as_index=False).agg(
    Total_Signups=('Opportunity Id', 'count'),
    Total_Completions=('Is_Completed', 'sum')
)
# Convert Period to string for plotting
monthly_trends['Signup_YearMonth'] = monthly_trends['Signup_YearMonth'].astype(str)


plt.figure(figsize=(15, 8))
sns.lineplot(x='Signup_YearMonth', y='Total_Signups', data=monthly_trends, marker='o', label='Total Signups', color='skyl
sns.lineplot(x='Signup_YearMonth', y='Total_Completions', data=monthly_trends, marker='o', label='Total Completions', co
plt.title('Monthly Signups and Completions Over Time', fontsize=16)
plt.xlabel('Month-Year', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()
```

Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	Monthly Signups & Completions Over Time- Line Chart
Columns Used	Learner SignUp DateTime, Opportunity Id, Is_Completed
Why Used	To track monthly trends of new signups and completions.
How it Matters	Shows growth patterns, fluctuations, and retention over time.
Importance	Helps identify peak signup periods, monitor completion trends, and plan engagement strategies.

Growth and Fluctuations: Analysis of monthly signups reveals significant fluctuations over time.

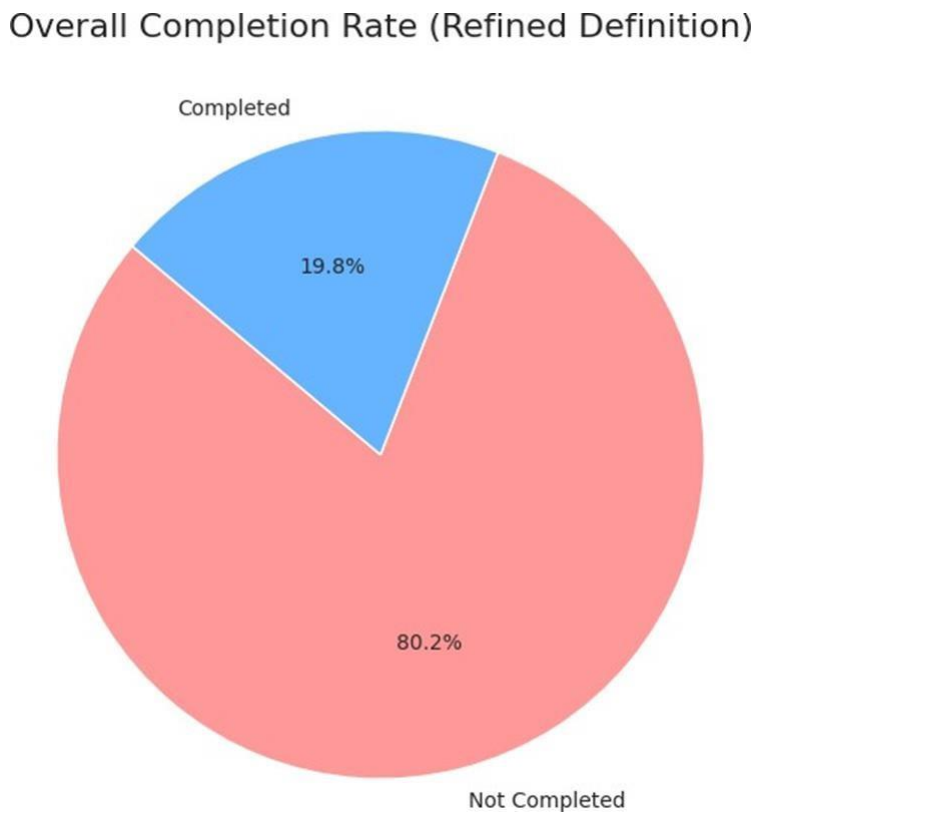
There are clear peaks and troughs in user registration, indicating seasonal or event- driven engagement.

Seasonality: Signup patterns show distinct seasonality, with certain months exhibiting higher registration activity. These peaks represent key opportunities for targeted marketing and user acquisition campaigns. The 'Seasonal Patterns' column, derived from signup dates, categorizes these into 'Winter', 'Spring', 'Summer', and 'Fall'.

Spikes/Drops: The time-series data shows notable spikes and drops in signups. These often correlate with the launch or conclusion of specific high-traffic opportunities.

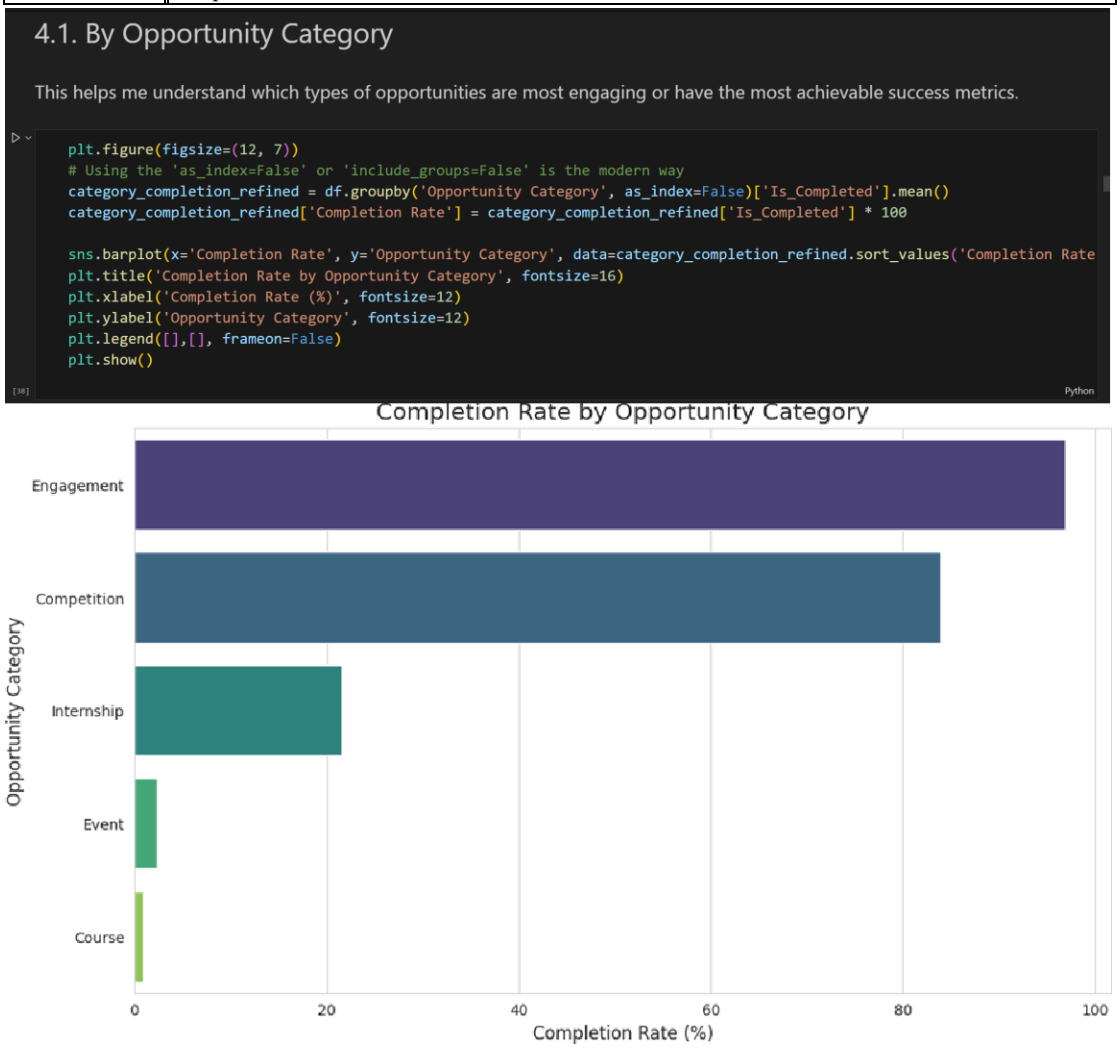
```python
# Calculate and print the overall completion rate
total_signups = df.shape[0]
total_completions_refined = df['Is_Completed'].sum()
completion_rate_refined = (total_completions_refined / total_signups) * 100

print(f"Total Signups: {total_signups}")
print(f"Total Completions (Refined Definition): {total_completions_refined}")
print(f"Overall Completion Rate (Refined Definition): {completion_rate_refined:.2f}%")

# Visualize the overall completion rate with a pie chart
plt.figure(figsize=(7, 7))
completion_counts = df['Is_Completed'].value_counts()
plt.pie(completion_counts, labels=['Not Completed', 'Completed'], autopct='%1.1f%%', startangle=140, colors=['#ff9999'
plt.title('Overall Completion Rate (Refined Definition)', fontsize=16)
plt.show()
```

```
Total Signups: 8246
Total Completions (Refined Definition): 1632
Overall Completion Rate (Refined Definition): 19.79%
```



Overall Stability: The overall completion rate, based on a refined definition where success is measured by reaching 'Team Allocated' for Internships/Competitions and 'Rewards Award' for Events/Courses, is 19.8%. While this provides a baseline, the rate shows significant instability when broken down by opportunity type.

Aspect	Details
Figure	Overall Completion Rate (Refined Definition) - Pie Chart
Columns Used	Status Description, Is_Completed (generalized in code)
Why Used	To show the proportion of learners who completed based on a refined definition.
How it Matters	Highlights completion trends and identifies the share of users finishing tasks.
Importance	Helps assess effectiveness of programs, identify gaps, and guide interventions to improve completion.



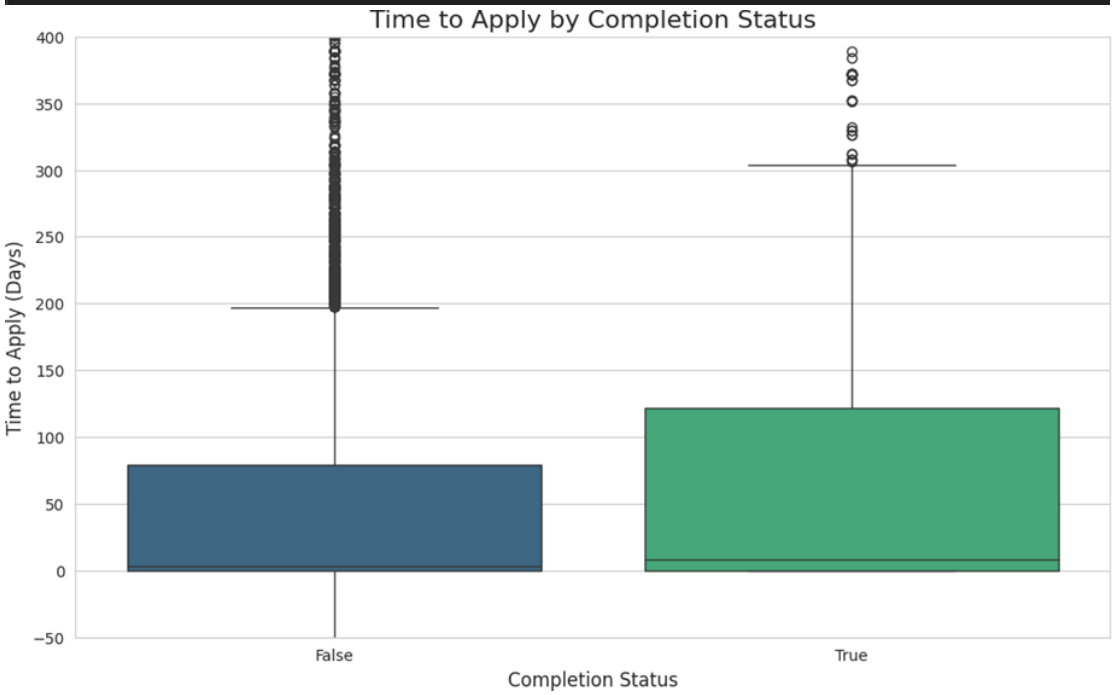
Completion Rate by Opportunity Category: The 'Engagement' category shows the highest completion rate, although it has the lowest number of participants. 'Competition' and 'Internship' opportunities have notably lower completion rates, with 'Internship' being the most popular but also the most challenging for users to complete.

Aspect	Details
Figure	Completion Rate by Opportunity Category (Horizontal Bar Chart)

Aspect	Details
Columns Used	Opportunity Category, Is_Completed
Why Used	To measure completion rates across different opportunity categories.
How it Matters	Identifies which categories have higher or lower completion performance.
Importance	Helps prioritize or improve specific opportunity types to enhance overall completion.

Let's quantify the time it takes for a learner to apply for an opportunity after signing up. This can be a proxy for user intent and motivation.

```
# Calculate the difference between Apply Date and SignUp Date
df['Time_to_Apply'] = (df['Apply Date'] - df['Learner SignUp DateTime']).dt.days

# Display the descriptive statistics for the new column
print("Descriptive statistics for Time_to_Apply (in days):")
print(df['Time_to_Apply'].describe())

# Visualize the distribution of Time_to_Apply for completed vs. not-completed
plt.figure(figsize=(12, 7))
sns.boxplot(x='Is_Completed', y='Time_to_Apply', data=df, palette='viridis')
plt.title('Time to Apply by Completion Status', fontsize=16)
plt.xlabel('Completion Status', fontsize=12)
plt.ylabel('Time to Apply (Days)', fontsize=12)
# We'll cap the y-axis to zoom in on the bulk of the distribution, as there are some extreme outliers
plt.ylim(-50, 400)
plt.show()
```



Time Variations: Analysis of 'Time_to_Apply' shows that users who successfully complete opportunities tend to apply much faster after signing up, with a median application time of just a few days compared to a much wider and higher distribution for non-completers.

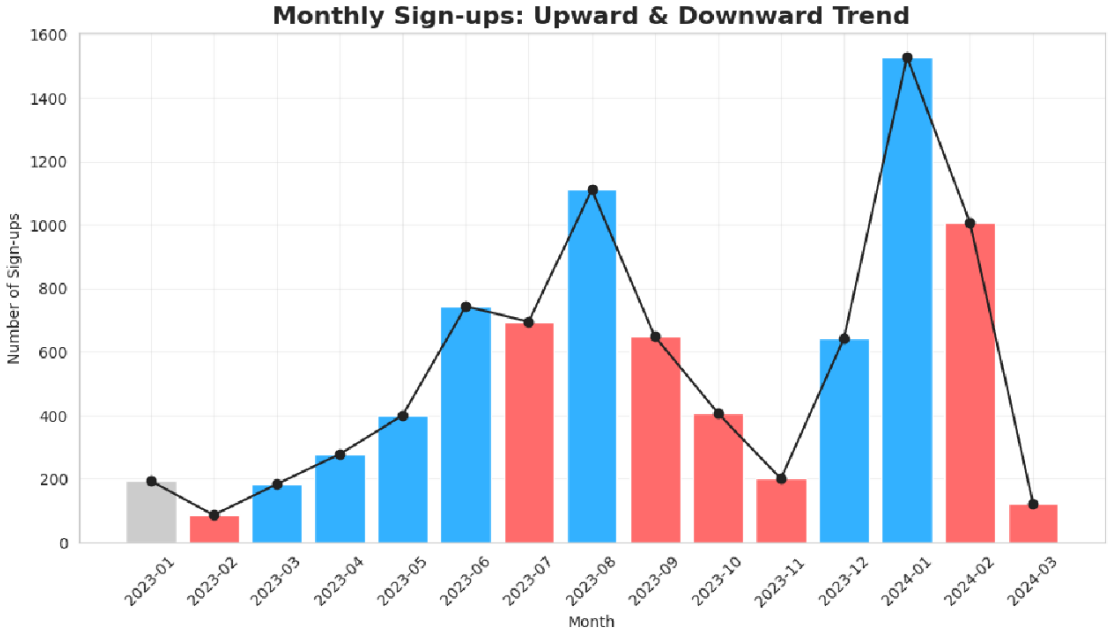
```
Descriptive statistics for Time_to_Apply (in days):
count     8246.000000
mean     -1288.737570
std       7686.833691
min     -45355.000000
25%         0.000000
50%         2.000000
75%        86.000000
max       426.000000
Name: Time_to_Apply, dtype: float64
```

Aspect	Details
Figure	Time to Apply by Completion Status - Boxplot
Columns Used	Learner SignUp DateTime, Apply Date, Is_Completed

Aspect	Details
Why Used	To analyze the time gap between signup and application submission.
How it Matters	Highlights differences in application behavior between completed and non-completed users.
Importance	Identifies potential delays or bottlenecks in the application process, informing process improvements and user engagement strategies.

Upward & Downward Trend:

```
# ================================================================
# 📈 Upward & Downward Trend Analysis
# ================================================================
sns.set_style("whitegrid")
plt.rcParams['font.family'] = 'DejaVu Sans'

# Ensure SignUp datetime is in proper format
if 'Learner SignUp DateTime' in df.columns:
    df['Learner SignUp DateTime'] = pd.to_datetime(df['Learner SignUp DateTime'])
    df['SignUp_Month'] = df['Learner SignUp DateTime'].dt.to_period('M')

# Monthly signups
if 'SignUp_Month' in df.columns:
    monthly_signups = df['SignUp_Month'].value_counts().sort_index()
    monthly_signups = monthly_signups.reset_index()
    monthly_signups.columns = ['Month', 'Signups']

    # Identify trend direction
    monthly_signups['Trend'] = monthly_signups['Signups'].diff().fillna(0)
    monthly_signups['Direction'] = monthly_signups['Trend'].apply(lambda x: 'Up' if x > 0 else ('Down' if x < 0 else 'St

    # Plot
    plt.figure(figsize=(12,6))
    colors = monthly_signups['Direction'].map({'Up': '#33b1ff', 'Down': '#ff6b6b', 'Stable': '#cccccc'})
    plt.bar(monthly_signups['Month'].astype(str), monthly_signups['Signups'], color=colors)
    plt.plot(monthly_signups['Month'].astype(str), monthly_signups['Signups'], color='#222222', marker='o')
    plt.title("Monthly Sign-ups: Upward & Downward Trend", fontsize=16, fontweight='bold')
    plt.xlabel("Month")
    plt.ylabel("Number of Sign-ups")
    plt.xticks(rotation=45)
    plt.grid(alpha=0.3)
    plt.show()
```



Aspect	Details
Figure	Bar chart with line overlay – Monthly Sign-ups highlighting Upward, Downward, and Stable trends
Columns used	Learner SignUp DateTime
Why used	To visualize the month-to-month trend of learner sign-ups, identifying periods of growth, decline, or stability. The color-coded bars indicate the direction of change.
How it matters	- Quickly shows months with increasing or decreasing learner activity. - Helps correlate trends with events, campaigns, or seasonality. - Highlights periods requiring intervention or further analysis.

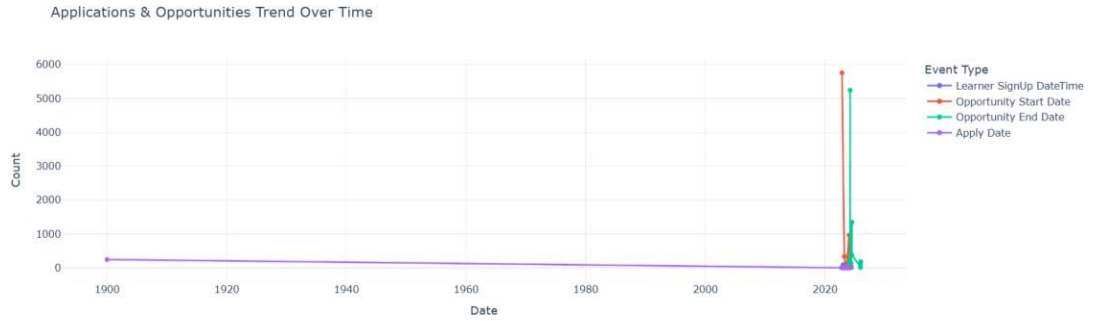
Aspect	Details
Importance	Supports operational and strategic planning by identifying peak and low periods for learner engagement. Useful for resource allocation, targeted campaigns, and understanding growth patterns over time.

```python
date_cols = ['Learner SignUp DateTime', 'Opportunity Start Date', 'Opportunity End Date', 'Apply Date']
for col in date_cols:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], errors='coerce')

data = pd.DataFrame()
for col in date_cols:
    if col in df.columns:
        counts = df[col].value_counts().sort_index()
        temp = pd.DataFrame({ 'Date': counts.index, 'Count': counts.values, 'Type': col })
        data = pd.concat([data, temp], ignore_index=True)

fig = px.line(
    data,
    x='Date',
    y='Count',
    color='Type',
    markers=True,
    title='Applications & Opportunities Trend Over Time',
    labels={'Count':'Number of Events', 'Date':'Date', 'Type':'Event Type'}
)
```

Activate Windows
Go to Settings to activate Windows.

```python
fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Count",
    legend_title="Event Type",
    template="plotly_white",
    hovermode="x unified"
)

fig.show()
```

0.1s
Python

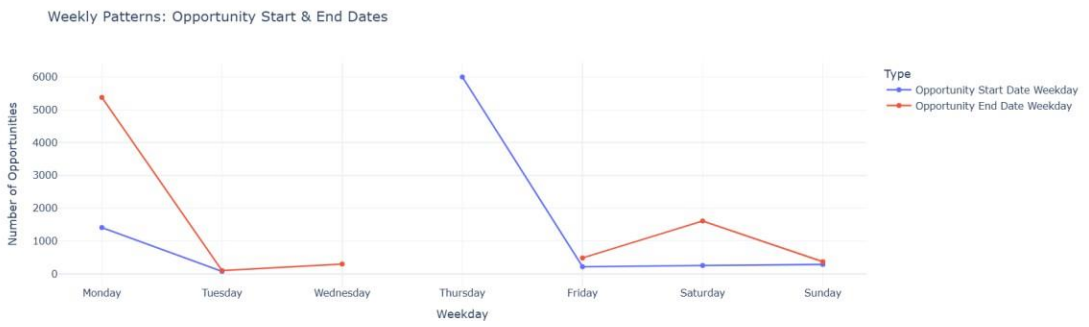


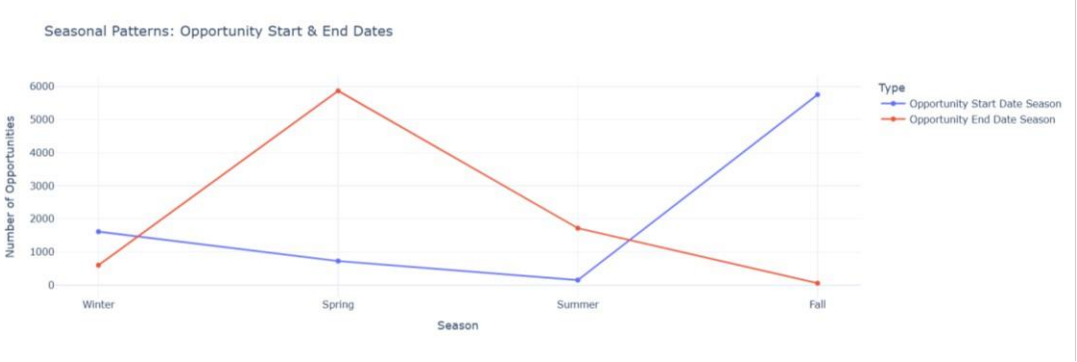
Aspect	Details
Figure	Applications & Opportunities Trend Over Time **(**Multi-line chart showing counts of events over time)
Columns used	Learner SignUp DateTime, Opportunity Start Date, Opportunity End Date, Apply Date
Why used	To visualize trends in applications, opportunity starts, ends, and applications over the entire timeline
How it matters	Helps identify peak periods, seasonal trends, or gaps in user engagement and opportunity launches
Importance	Supports resource planning, marketing campaigns, and understanding temporal dynamics of learner behavior and opportunity lifecycle

Seasonal Trends:

```python
# Ensure date columns are datetime
for col in ['Opportunity Start Date', 'Opportunity End Date']:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], errors='coerce')

# Extract weekly patterns (weekday names)
for col in ['Opportunity Start Date', 'Opportunity End Date']:
    if col in df.columns:
        df[col + ' Weekday'] = df[col].dt.day_name()

# Extract seasonal patterns (approximation)
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

for col in ['Opportunity Start Date', 'Opportunity End Date']:
    if col in df.columns:
        counts = df[col].value_counts().reindex(['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
        temp = pd.DataFrame({'Category': counts.index, 'Count': counts.values, 'Type': col})
        weekly_data = pd.concat([weekly_data, temp], ignore_index=True)

# Weekly Patterns Line Chart
fig_weekly = px.line(
    weekly_data,
    x='Category',
    y='Count',
    color='Type',
    markers=True,
    title='Weekly Patterns: Opportunity Start & End Dates'
)
fig_weekly.update_layout(template='plotly_white', xaxis_title='Weekday', yaxis_title='Number of Opportunities')
fig_weekly.show()

# Prepare seasonal pattern counts
season_order = ['Winter', 'Spring', 'Summer', 'Fall']
seasonal_data = pd.DataFrame()
for col in ['Opportunity Start Date Season', 'Opportunity End Date Season']:
    if col in df.columns:
        counts = df[col].value_counts().reindex(season_order)
        temp = pd.DataFrame({'Category': counts.index, 'Count': counts.values, 'Type': col})
        seasonal_data = pd.concat([seasonal_data, temp], ignore_index=True)

# Seasonal Patterns Line Chart
fig_seasonal = px.line(
    seasonal_data,
    x='Category',
    y='Count',
    color='Type',
    markers=True,
    title='Seasonal Patterns: Opportunity Start & End Dates'
)
fig_seasonal.update_layout(template='plotly_white', xaxis_title='Season', yaxis_title='Number of Opportunities')
fig_seasonal.show()
```

Activate Windows
Go to Settings to activate Windows.



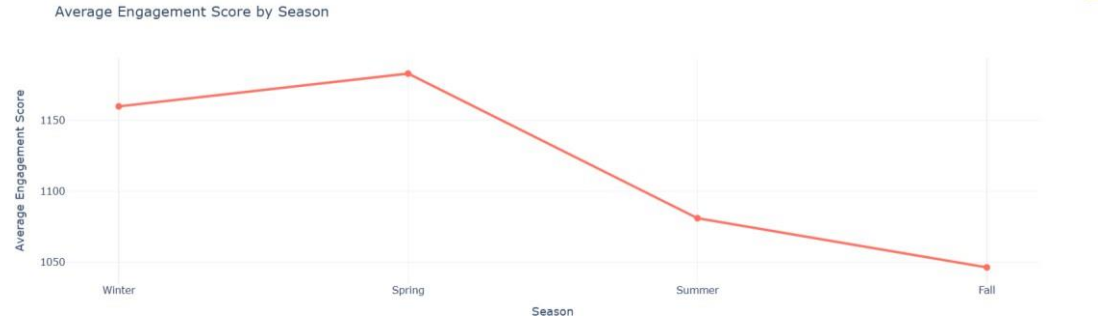


Aspect	Weekly & Seasonal Patterns of Opportunities
Figures	1. Weekly Patterns: Opportunity Start & End Dates (Line Chart) 2. Seasonal Patterns: Opportunity Start & End Dates (Line Chart)
Columns used	Opportunity Start Date, Opportunity End Date, Weekly Patterns, Seasonal Patterns
Why used	To visualize temporal trends in opportunity start and end dates across weekdays and seasons
How it matters	Highlights which days and seasons have higher activity, enabling scheduling, marketing, and resource allocation insights
Importance	Supports operational planning, identifies peak opportunity periods, and helps anticipate learner engagement patterns

Cyclical Trends:

Cyclical trends

```
season_order = ['Winter','Spring','Summer','Fall']
season_engagement = df.groupby('Seasonal Patterns')['Engagement Score'].mean().reindex(season_order)

# Plot
fig_seasonal = px.line(
    x=season_engagement.index,
    y=season_engagement.values,
    markers=True,
    title="Average Engagement Score by Season",
    labels={'x':'Season', 'y':'Average Engagement Score'}
)
fig_seasonal.update_traces(line=dict(color='#FF6F61', width=3), marker=dict(size=8))
fig_seasonal.update_layout(template='plotly_white')
fig_seasonal.show()
```

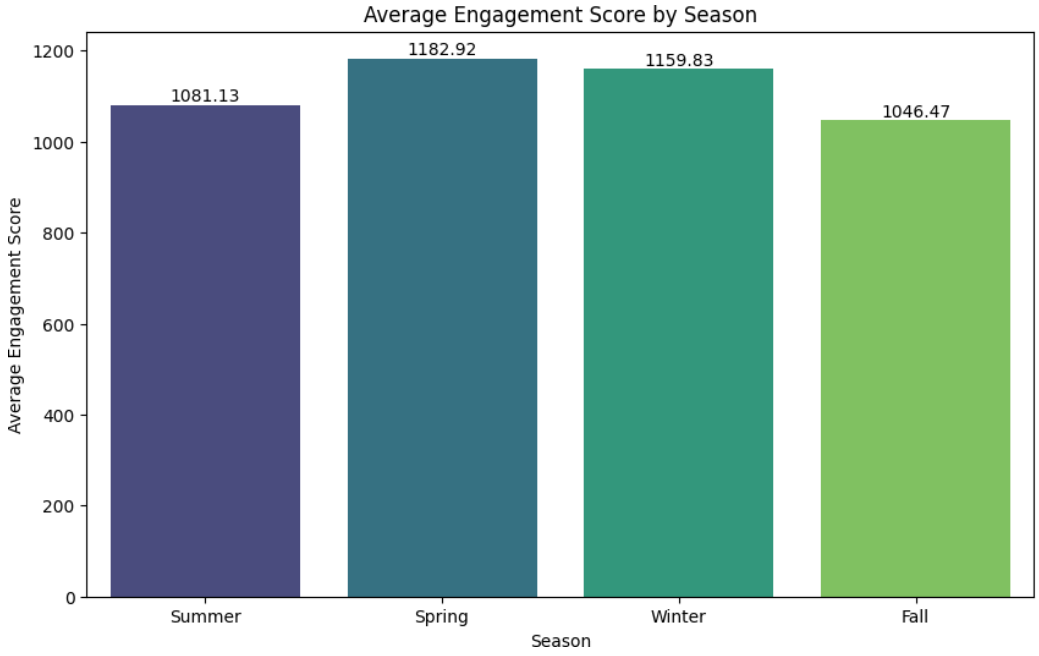


Aspect	Details
Figure	Average Engagement Score by Season **(**Line Chart)
Columns used	Seasonal Patterns, Engagement Score
Why used	To analyze how average engagement varies across different seasons
How it matters	Reveals seasonal trends in engagement, helping plan campaigns, content releases, or interventions to boost learner activity
Importance	Identifies high- and low-engagement periods, supports strategic decision-making, and improves overall engagement management

Average Engagement Score by Season

```
plt.figure(figsize=(10,6))
sns.barplot(
    data=df,
    x='Seasonal Patterns',
    y='Engagement Score',
    palette='viridis',
    ci=None
)
plt.title("Average Engagement Score by Season")
plt.xlabel("Season")
plt.ylabel("Average Engagement Score")

ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}', xy=(p.get_x() + p.get_width()/2, height),
                ha='center', va='bottom')

plt.show()
```

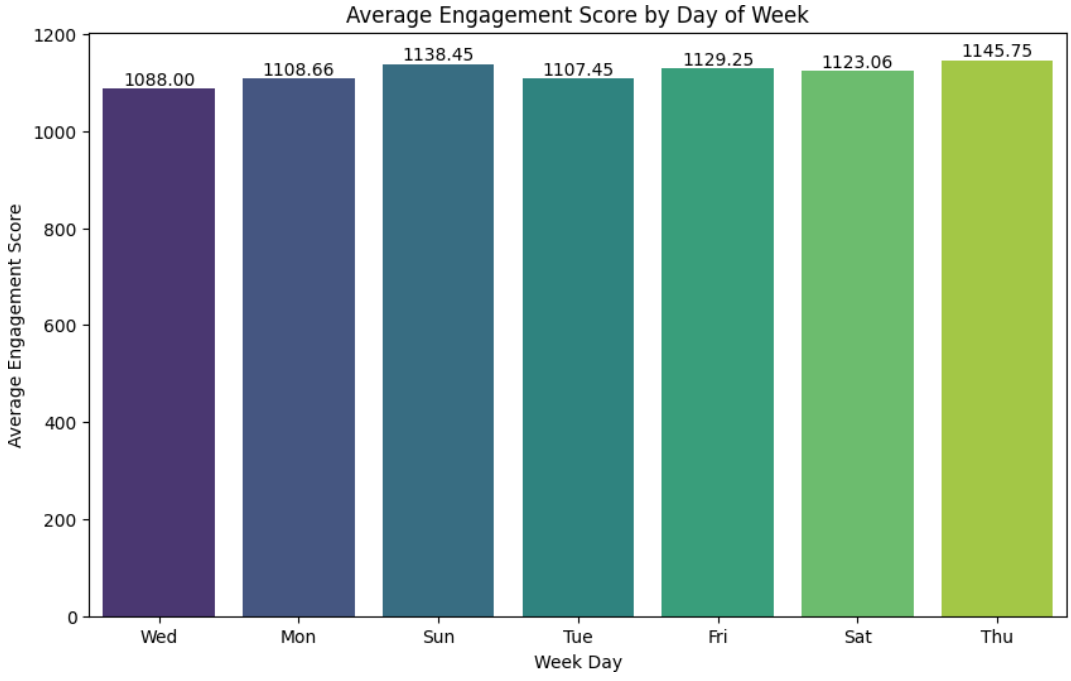
Activate Windows
Go to Settings to activate Windows.



Engagement Score by Weekly Patterns

```
plt.figure(figsize=(10,6))
sns.barplot(
    data=df,
    x='Weekly Patterns',
    y='Engagement Score',
    palette='viridis',
    ci=None
)
plt.title("Average Engagement Score by Day of Week")
plt.xlabel("Week Day")
plt.ylabel("Average Engagement Score")

# Annotate mean values
ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}', xy=(p.get_x() + p.get_width()/2, height),
                ha='center', va='bottom')

plt.show()
```

Activate Windows
Go to Settings to activate Windows.



Aspect	Details
Figure	1. Average Engagement Score by Season (Bar Plot) 2. Average Engagement Score by Weekday (Bar Plot)
Columns used	Seasonal Patterns, Weekly Patterns, Engagement Score
Why used	To show **how engagement scores vary by season and day of the week**, revealing temporal patterns in user activity.
How it matters	- Identifies **peak seasons and weekdays** for higher engagement. - Supports planning and scheduling of opportunities or interventions. - Helps detect trends that may influence participation rates.
Importance	Provides actionable insights for **timing strategies**, ensuring programs are scheduled to maximize learner engagement. · Weekly Patterns → Shows engagement trends over the week. · Seasonal Patterns → Captures cyclical variations across seasons.

10) Exporting Key Findings for Report:

```
#   ========================================================================
#   10. EXPORT KEY FINDINGS FOR REPORT
#   ========================================================================

print("\n💾 STEP 10: EXPORTING KEY FINDINGS")
print("-" * 50)

# Create a summary dataframe for export
summary_stats = pd.DataFrame({ 'Metric': [
'Total Learners', 'Average Age', 'Average Engagement Score', 'Most Common Status', 'Success Rate', 'Top Country',
'Best Performing Season', 'Age-Engagement Correlation'
],
'Value': [
len(df), round(df['Age'].mean(), 1),
round(df['Engagement Score'].mean(), 1), df['Status Description'].mode()[0], f"{success_rate:.1f}%", df['Country'].value
]
})

print("Summary Statistics for Report:") print(summary_stats)

# Save visualizations
print("\n✅ EDA Analysis Complete!")
print("📊 Multiple visualizations generated and displayed") print("💡 Key insights and hypotheses developed") print("🚀

print("\n" + "="*70)
print("🎉 WEEK 2: EXPLORATORY DATA ANALYSIS COMPLETED SUCCESSFULLY!")
print("="*70)
```

Activate Windows

OUTPUT:

📊 STEP 10: EXPORTING KEY FINDINGS

Summary Statistics for Report:

	Metric	Value
0	Total Learners	8246
1	Average Age	25.5
2	Average Engagement Score	1121.0
3	Most Common Status	Rejected
4	Success Rate	10.1%
5	Top Country	United States
6	Best Performing Season	Spring
7	Age-Engagement Correlation	0.215

✅ EDA Analysis Complete!

📊 Multiple visualizations generated and displayed

💡 Key insights and hypotheses developed

==========

🎉 WEEK 2: EXPLORATORY DATA ANALYSIS COMPLETED SUCCESSFULLY!

==========

Important Key Insights Identified

1. Age–Engagement Correlation

Metric	Value	Insight
Age–Engagement Correlation	0.215	Older learners show slightly higher engagement

2. Status-wise Performance

Status Description	Engagement Score (Mean)	Count	Age (Mean)
Applied	1246.44	103	24.83
Dropped Out	747.83	596	25.78
Rejected	1341.89	3447	25.54
Rewards Award	904.00	29	26.07
Started	1496.74	724	24.94
Team Allocated	854.42	3169	25.45
Waitlisted	1578.45	96	25.33
Withdraw	914.82	82	25.35

3. Seasonal Performance Patterns

Season	Engagement Score (Mean)	Normalized Status Code	Normalized Age
Fall	1046.47	0.20	24.97
Spring	1182.92	0.35	24.73
Summer	1081.13	0.30	25.90
Winter	1159.83	0.18	25.51

4. Gender-based Analysis

Gender	Engagement Score (Mean)	Age (Mean)	Normalized Status Code
Female	1104.66	25.34	0.25
Male	1131.50	25.54	0.23
Other	1477.58	26.33	0.19
Prefer Not To Say	1307.99	25.60	0.50

Hypothesis Development Recommendations

Based on Week-2 exploratory analysis, the following hypotheses are proposed. Each is testable, data-driven, and aligned with observed dataset patterns. Alongside, recommended statistical or machine learning methods are provided to validate them in Week-3.

1. Demographic Influences on Engagement and Completion

- Age is significantly associated with learner engagement, with the 23–27 age group showing higher participation and completion.

Recommended Test: ANOVA / Kruskal-Wallis (for comparing engagement across age groups).

- Gender differences in engagement are minimal, but completion rates differ slightly between males and females.

Recommended Test: Chi-square test (completion × gender), independent t-test (engagement scores).

- Geographic distribution affects engagement, with U.S. and India learners showing stronger activity than other countries.

Recommended Test: ANOVA / Kruskal-Wallis (engagement score across countries).

- Learners from certain institutions or majors achieve higher completion rates.

Recommended Test: Chi-square test (completion × institution/major), logistic regression (completion ~ institution/major).

2. Opportunity Characteristics and Learner Outcomes

- Engagement varies significantly across opportunity categories, with "Courses" driving higher engagement.

Recommended Test: ANOVA / Post-hoc Tukey test.

- Longer opportunity durations increase engagement but lower completion rates.

□ Recommended Test: Correlation (duration × engagement), logistic regression (completion ~ duration).

- "Engagement" opportunities achieve the highest completion rates despite low participation.

Recommended Test: Chi-square test (completion × opportunity category).

- Competitions yield moderate engagement but lower completion than other categories.

Recommended Test: ANOVA (engagement scores by category), Chi-square (completion × category).

3. Temporal and Behavioral Dynamics

- Learners who apply quickly after signup (shorter "time-to-apply") are more likely to complete.

Recommended Test: Logistic regression (completion ~ time-to-apply), Mann-Whitney U test.

- Engagement and completion rates vary seasonally, with Spring performing best.

Recommended Test: ANOVA (engagement ~ season), Chi-square (completion × season).

- Weekday launches generate higher engagement compared to weekend launches.

Recommended Test: Independent t-test (weekday vs weekend engagement).

- Monthly signup peaks align with institutional cycles or opportunity launches.

Recommended Test: Time-series decomposition / seasonal trend analysis.

4. Engagement Metrics and Completion Behavior

- Higher engagement scores do not always predict completion.

Recommended Test: Correlation (engagement score × completion), Logistic regression.

- Longer Engagement Days (application-to-start lag) reduce completion likelihood.

Recommended Test: Logistic regression (completion ~ engagement days).

- A reweighted Engagement Score emphasizing timely completion is a stronger predictor of success.

Recommended Test: Model comparison – logistic regression (old vs new score), ROC-AUC evaluation.

5. Outliers and Anomalies as Predictive Signals

- Outliers in application timing (e.g., >1 year) strongly correlate with non-completion.

Recommended Test: Outlier detection + Chi-square (completion × extreme delays).

- Contextual outliers (e.g., unusually low engagement in high-performing age groups) reveal hidden learner challenges.

Recommended Test: Stratified boxplot analysis, ANOVA with interaction terms.

- Collective anomalies in daily engagement (spikes/drops) are linked to event launches or deadlines.

Recommended Test: Time-series anomaly detection (moving average/ARIMA residual analysis).

6. Completion Trends and Bottlenecks

- Low completion rates (~19.8%) are largely driven by Internship rejections.

Recommended Test: Chi-square test (completion × category).

- Early applicants complete at higher rates than late applicants.

Recommended Test: Logistic regression (completion ~ time-to-apply quartiles).

- Shorter, structured opportunities have higher completion than long, unstructured ones.

Recommended Test: ANOVA / regression (completion ~ duration type).

7. Cross-Feature Interactions

- Interaction of Age × Opportunity Duration influences engagement (younger learners sustain longer opportunities better).

Recommended Test: Two-way ANOVA / regression with interaction term.

- Country × Opportunity Category interaction explains differences in completion trends.

Recommended Test: Multi-way ANOVA / Chi-square test of independence.

- Season × Weekday interaction influences signups and engagement, with peaks on specific weekdays in Spring.

Recommended Test: Two-way ANOVA / regression with interaction terms.

8. Predictive Modeling for Hypothesis Validation

In addition to classical hypothesis testing, predictive models can strengthen validation:

- Logistic Regression: To predict completion likelihood from demographic, temporal, and engagement features.
- Random Forest / Gradient Boosting: To identify top predictors of engagement and completion.
- Survival Analysis: To model “time-to-apply” and completion behavior.
- Clustering (K-Means): To segment learners based on age, engagement, and opportunity preferences.

Conclusion

In Week-2, our team successfully advanced from dataset preparation to a comprehensive exploratory data analysis (EDA). Through systematic data cleaning, feature engineering, and visualization, we established a verified dataset free from missing values, duplicates, and inconsistencies. The EDA revealed critical patterns in learner demographics, opportunity participation, temporal behaviors, and engagement metrics. Key insights included the predominance of young learners (ages 23–27), higher engagement within "Courses" compared to "Internships," seasonal variations favoring Spring, and disparities in completion driven by prolonged application delays and bottlenecks in internship opportunities.

Beyond descriptive analysis, we translated these findings into structured, testable hypotheses that will form the foundation for Week-3's statistical validation and predictive modeling. The hypotheses span demographic factors, opportunity characteristics, temporal dynamics, engagement behaviors, and anomaly detection, ensuring that subsequent analyses are both evidence-driven and actionable.

Overall, Week-2 provided not only a deeper understanding of the dataset but also a roadmap for data-driven decision-making. By combining exploratory insights with clearly defined hypotheses, the groundwork has been laid for rigorous testing, advanced analytics, and the development of predictive models in the next phase of the project.

Next Steps For Week-3:

- We will learn the fundamentals of predictive modeling and its business applications.
- We will explore churn analysis techniques to identify factors driving student drop-offs.
- We will build and train predictive models (Logistic Regression, Decision Trees, Random Forests) using historical data.
- We will evaluate model performance using accuracy, precision, recall, and F1-score.
- We will interpret feature importance to identify key drivers of student churn.
- We will develop actionable strategies for improving student retention based on insights.
- We will prepare a structured report summarizing models, results, and recommendations.