Notebook

March 15, 2025

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

[4]:
```
# Read data
data = pd.read_csv('Learning_Intern_Raw_Data.csv', sep=',')
data
```

[4]:
```
                                    user_id  gender age_range                 phone  \
0      6531ed09-304e-9573-a9c2-54e917  FEMALE     25-29  ["23409089964086"]
1      6531ed09-304e-9573-a9c2-54e917  FEMALE     25-29  ["23409089964086"]
2      6531ed09-304e-9573-a9c2-54e917  FEMALE     25-29  ["23409089964086"]
3      6531ed09-304e-9573-a9c2-54e917  FEMALE     25-29  ["23409089964086"]
4      6531ed09-304e-9573-a9c2-54e917  FEMALE     25-29  ["23409089964086"]
…                                   …       …         …                   …
65528  95539e11-a210-2b56-d5a4-a1b85f  FEMALE     30-34  ["23408067849026"]
65529  95539e11-a210-2b56-d5a4-a1b85f  FEMALE     30-34  ["23408067849026"]
65530  95539e11-a210-2b56-d5a4-a1b85f  FEMALE     30-34  ["23408067849026"]
65531  95539e11-a210-2b56-d5a4-a1b85f  FEMALE     30-34  ["23408067849026"]
65532  95539e11-a210-2b56-d5a4-a1b85f  FEMALE     30-34  ["23408067849026"]

       state               program               assignment_name  \
0      Lagos       Cloud Computing                  Milestone #1
1      Lagos       Cloud Computing                Weekly Test #1
2      Lagos       Cloud Computing                  Milestone #2
3      Lagos       Cloud Computing                Weekly Test #2
4      Lagos       Cloud Computing                  Milestone #3
…          …                     …                             …
65528  Lagos  AI Career Essentials                  Milestone #5
65529  Lagos  AI Career Essentials                Weekly Test #5
65530  Lagos  AI Career Essentials                  Milestone #6
65531  Lagos  AI Career Essentials                Weekly Test #6
65532  Lagos  AI Career Essentials  AiCE Final Grace Period Test

      assignment_type  assignment_score has_logged_into_lms  \
0           milestone            100.00                 Yes
```

```
1              test          95.23              Yes
2         milestone          94.55              Yes
3              test          90.90              Yes
4         milestone         100.00              Yes
…                 …              …                …
65528     milestone         100.00              Yes
65529          test         100.00              Yes
65530     milestone         100.00              Yes
65531          test         100.00              Yes
65532          test           0.00              Yes

      Is assignment resubmitted learner_deferred  learner_dropped_off  \
0                            No               No                  NaN
1                            No               No                  NaN
2                            No               No                  NaN
3                            No               No                  NaN
4                            No               No                  NaN
…                             …                …                    …
65528                        No               No                  NaN
65529                       Yes               No                  NaN
65530                        No               No                  NaN
65531                       Yes               No                  NaN
65532                        No               No                  NaN

       overall_score  graduated
0                NaN        NaN
1                NaN        NaN
2                NaN        NaN
3                NaN        NaN
4                NaN        NaN
…                  …          …
65528            NaN        NaN
65529            NaN        NaN
65530            NaN        NaN
65531            NaN        NaN
65532            NaN        NaN

[65533 rows x 15 columns]
```

[5]:
```
# Drop unnecessary columns
df = data[['user_id', 'program', 'assignment_name', 'assignment_type',␣
 ↪'assignment_score', 'has_logged_into_lms', 'Is assignment resubmitted',␣
 ↪'learner_deferred', 'learner_dropped_off', 'overall_score', 'graduated']]
```

[13]:
```
# Remove deferred users
df = df[df["learner_deferred"] == "No"]
```

```
print("Deferred users removed. Updated dataset saved.")
```

```
Deferred users removed. Updated dataset saved.
```

[6]:
```
# Filter only 'test' assignment types for overall_score
test_scores = df[df['assignment_type'] == 'test']
test_scores
```

[6]:
```
                                    user_id               program  \
1      6531ed09-304e-9573-a9c2-54e917       Cloud Computing
3      6531ed09-304e-9573-a9c2-54e917       Cloud Computing
5      6531ed09-304e-9573-a9c2-54e917       Cloud Computing
7      6531ed09-304e-9573-a9c2-54e917       Cloud Computing
9      6531ed09-304e-9573-a9c2-54e917       Cloud Computing
…                                      …                   …
65525  95539e11-a210-2b56-d5a4-a1b85f  AI Career Essentials
65527  95539e11-a210-2b56-d5a4-a1b85f  AI Career Essentials
65529  95539e11-a210-2b56-d5a4-a1b85f  AI Career Essentials
65531  95539e11-a210-2b56-d5a4-a1b85f  AI Career Essentials
65532  95539e11-a210-2b56-d5a4-a1b85f  AI Career Essentials

                   assignment_name assignment_type  assignment_score  \
1                   Weekly Test #1            test             95.23
3                   Weekly Test #2            test             90.90
5                   Weekly Test #3            test            100.00
7                   Weekly Test #4            test              0.00
9                   Weekly Test #5            test              0.00
…                                …               …                …
65525               Weekly Test #3            test            100.00
65527               Weekly Test #4            test             80.95
65529               Weekly Test #5            test            100.00
65531               Weekly Test #6            test            100.00
65532  AiCE Final Grace Period Test            test              0.00

      has_logged_into_lms Is assignment resubmitted learner_deferred  \
1                     Yes                        No               No
3                     Yes                        No               No
5                     Yes                        No               No
7                     Yes                        No               No
9                     Yes                        No               No
…                       …                         …                …
65525                 Yes                        No               No
65527                 Yes                       Yes               No
65529                 Yes                       Yes               No
65531                 Yes                       Yes               No
65532                 Yes                        No               No
```

```
       learner_dropped_off  overall_score  graduated
1                      NaN            NaN        NaN
3                      NaN            NaN        NaN
5                      NaN            NaN        NaN
7                      NaN            NaN        NaN
9                      NaN            NaN        NaN
…                    …            …        …
65525                  NaN            NaN        NaN
65527                  NaN            NaN        NaN
65529                  NaN            NaN        NaN
65531                  NaN            NaN        NaN
65532                  NaN            NaN        NaN

[35287 rows x 11 columns]
```

[7]:
```
# Compute average test score per user
overall_scores = test_scores.groupby('user_id')['assignment_score'].mean()

# Merge computed scores back into the original DataFrame
df.loc[:, 'overall_score'] = df['user_id'].map(overall_scores)

# Save the updated file
df.to_excel("Updated_Learning_Intern.xlsx", index=False)

print("Overall scores updated successfully!")
```

```
Overall scores updated successfully!
```

[18]:
```
# Compute average overall score per user
avg_scores = df.groupby('user_id')['overall_score'].mean().reset_index()

# Create an interactive histogram
fig = px.histogram(avg_scores, x="overall_score", nbins=20,
                   title="Interactive Histogram: Distribution of Overall␣
 ↪Scores",
                   labels={"overall_score": "Average Overall Score", "count":␣
 ↪"Number of Learners"},
                   opacity=0.75, color_discrete_sequence=["blue"])

# Customize hover info
fig.update_traces(marker_line_width=1, marker_line_color="black",␣
 ↪hoverinfo="x+y")

# Show interactive plot
fig.show();
```

Over 2,600 Learners (more than half of the total) had test scores below 25. 1,901 learners achieved test scores between 75 - 100

[24]:
```
# Group by two columns and compute the mean overall score
grouped_df = df.groupby('program')['overall_score'].mean().reset_index()

# Create a bar chart
fig = px.bar(grouped_df, x="program", y="overall_score", color="program",
             title="Overall Test Score by Program",
             labels={"overall_score": "Average Test Score", "program":␣
 ↪"Program"},
             barmode="group",
             color_discrete_sequence=px.colors.qualitative.Set1)

fig.show();
```

All programs have relatively the same average test scores.

[8]:
```
df['overall_score']
```

[8]:
```
0        40.875714
1        40.875714
2        40.875714
3        40.875714
4        40.875714
           …
65528    81.692857
65529    81.692857
65530    81.692857
65531    81.692857
65532    81.692857
Name: overall_score, Length: 65533, dtype: float64
```

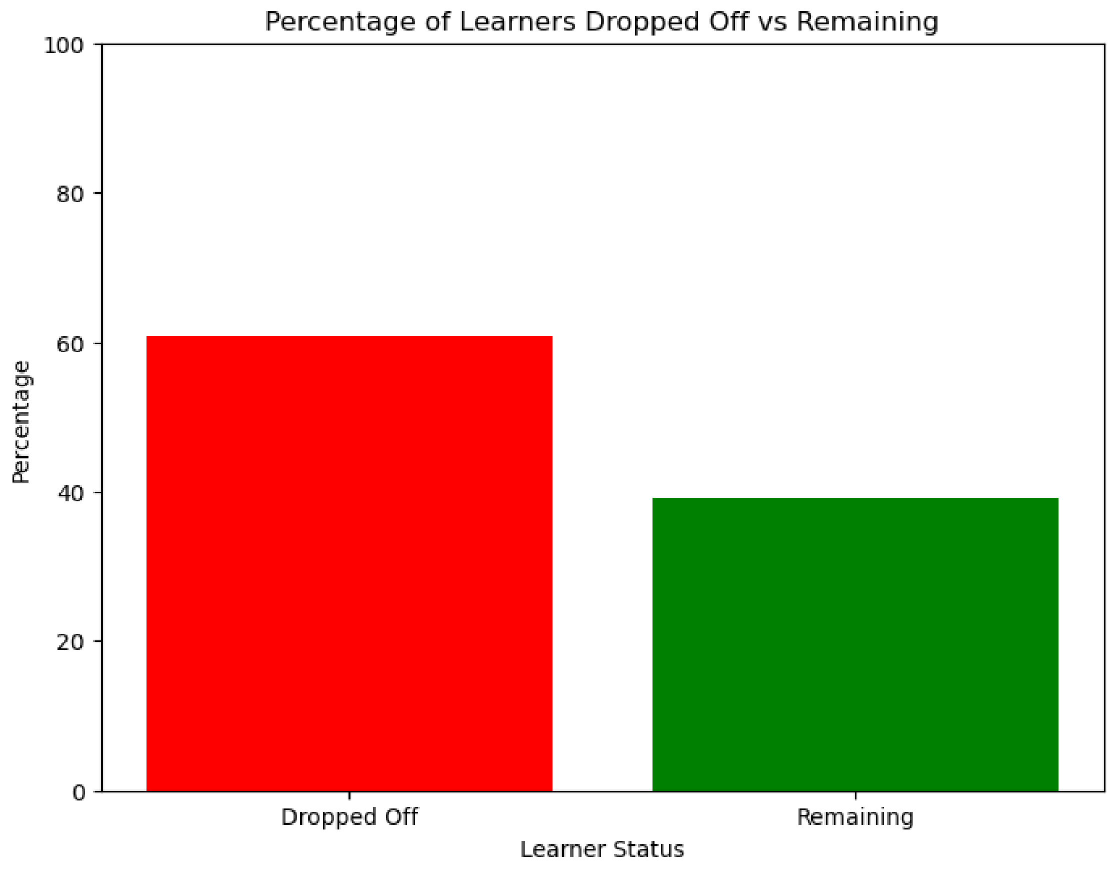
[25]:
```
# Define the required milestones for Learner_drop_off
required_milestones = ["Milestone #1", "Milestone #2", "Milestone #3"]

# Filter only required milestones
milestone_scores = df[df['assignment_name'].
 ↪isin(required_milestones)][['user_id', 'assignment_name',␣
 ↪'assignment_score']]

# Identify users who have **all three** milestones with a score greater than 0
valid_users = (
    milestone_scores.groupby('user_id')  # Exclude 'user_id' from groups
    .apply(lambda x: all(x.set_index('assignment_name').
 ↪loc[required_milestones, 'assignment_score'] > 0))
    .reset_index()
)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_14008\2721981694.py:10:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be
excluded from the operation. Either pass `include_groups=False` to exclude the
groupings or explicitly select the grouping columns after groupby to silence
this warning.
```

[27]:
```
# Get user IDs of those who submitted required milestones
submitted_all = valid_users[valid_users[0]]['user_id']

# Assign learner_dropped_off: 1 (Dropped Off), 0 (Stayed)
df.loc[:, 'learner_dropped_off'] = df['user_id'].apply(lambda x: 0 if x in␣
  ↪submitted_all.values else 1)

# Validation Step: Count dropped users
dropped_count = int(df['learner_dropped_off'].sum()/13)
total_users = df['user_id'].nunique()
print(f"Total Learners: {total_users}")
print(f"Learners Dropped Off: {dropped_count}")
print(f"Learners Remaining: {total_users - dropped_count}")

# Save the updated DataFrame
df.to_csv("updated_data.csv", index=False)

print("Users who didn't submit valid milestone scores have been marked as␣
  ↪dropped off successfully!")
```

```
Total Learners: 5002
Learners Dropped Off: 3040
Learners Remaining: 1962
Users who didn't submit valid milestone scores have been marked as dropped off
successfully!
```

[28]:
```
# Calculate percentages
dropped_off_percentage = (dropped_count / total_users) * 100
remaining_percentage = ((total_users - dropped_count) / total_users) * 100

# Data for plotting
labels = ['Dropped Off', 'Remaining']
percentages = [dropped_off_percentage, remaining_percentage]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(labels, percentages, color=['red', 'green'])
plt.xlabel('Learner Status')
plt.ylabel('Percentage')
plt.title('Percentage of Learners Dropped Off vs Remaining')
plt.ylim(0, 100)  # Set y-axis limit to 100%
plt.show();
```



[29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 65026 entries, 0 to 65532
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   user_id                     65026 non-null  object
 1   program                     65026 non-null  object
 2   assignment_name             65026 non-null  object
 3   assignment_type             65026 non-null  object
 4   assignment_score            65026 non-null  float64
 5   has_logged_into_lms         65026 non-null  object
 6   Is assignment resubmitted   65026 non-null  object
 7   learner_deferred           65026 non-null  object
 8   learner_dropped_off        65026 non-null  float64
 9   overall_score              65026 non-null  float64
 10  graduated                  0 non-null      float64
dtypes: float64(4), object(7)
memory usage: 6.0+ MB
```

[39]:
```
# Define the required milestones to graduate
reqd_milestones = ["Milestone #1", "Milestone #2", "Milestone #3", "Milestone␣
 ↪#4", "Milestone #5"]

# Filter only required milestones
mile_scores = df[df['assignment_name'].isin(reqd_milestones)][['user_id',␣
 ↪'assignment_name', 'assignment_score']]

# Drop rows where assignment_score is 0.00
mile_scores = mile_scores[mile_scores["assignment_score"] > 0]
```

[40]:
```
# Users who submitted all 5 milestones
f_mile_scores = mile_scores.groupby('user_id')['assignment_name'].count() > 4

# Filter the dataset to retain only 'f_mile_scores'
filtered_mile_scores = mile_scores[mile_scores['user_id'].
 ↪isin(f_mile_scores[f_mile_scores].index)]

# Display result
filtered_mile_scores
```

[40]:
```
                                  user_id assignment_name  assignment_score
39     02f0e023-398a-93e5-217e-5448f0    Milestone #1             96.55
41     02f0e023-398a-93e5-217e-5448f0    Milestone #2            100.00
43     02f0e023-398a-93e5-217e-5448f0    Milestone #3            100.00
45     02f0e023-398a-93e5-217e-5448f0    Milestone #4             99.68
47     02f0e023-398a-93e5-217e-5448f0    Milestone #5             96.67
…                                   …               …                 …
65520  95539e11-a210-2b56-d5a4-a1b85f    Milestone #1            100.00
65522  95539e11-a210-2b56-d5a4-a1b85f    Milestone #2             99.32
65524  95539e11-a210-2b56-d5a4-a1b85f    Milestone #3             98.53
65526  95539e11-a210-2b56-d5a4-a1b85f    Milestone #4            100.00
65528  95539e11-a210-2b56-d5a4-a1b85f    Milestone #5            100.00

[8670 rows x 3 columns]
```

[ ]:
```
f_mile_scores.info()
```

```
<class 'pandas.core.series.Series'>
Index: 2540 entries, 0034a903-0652-77b3-9c70-82b91a to
```

```
fff94200-23ac-3b93-fca0-178543
Series name: assignment_name
Non-Null Count  Dtype
--------------  -----
2540 non-null   bool
dtypes: bool(1)
memory usage: 86.9+ KB
```

[41]:
```
# Method 2 to obtain graduation milestone requirement

# Pivot table to ensure each user has all 5 milestones
milestone_pivot = mile_scores.pivot(index="user_id", columns="assignment_name",␣
 ↪values="assignment_score")

# Check if each user has ALL Milestones (#1 to #5)
users_with_5_milestones = milestone_pivot[reqd_milestones].gt(0).all(axis=1)

# Output Results
users_with_5_milestones
```

[41]:
```
user_id
0034a903-0652-77b3-9c70-82b91a    False
005b9685-1114-e183-1b69-45964c    False
005d3e80-f153-c133-8dea-4e5b66     True
0065897d-5ddf-d164-00c2-550e2e    False
00a5a86d-343d-3a27-e9ab-be6f4f    False
                                  …
ffbaaeb2-ab3c-03d3-a765-b5d611     True
ffd40f70-6761-dabd-b0b7-2c486c    False
ffd68c7b-8edc-9e0b-45f6-168767     True
fff8b323-3e76-e67d-cf1f-8390fb    False
fff94200-23ac-3b93-fca0-178543     True
Length: 2540, dtype: bool
```

[ ]:
```
milestone_pivot.head()
```

[ ]:
```
assignment_name                 Milestone #1  Milestone #2  Milestone #3  \
user_id
0034a903-0652-77b3-9c70-82b91a        100.00           NaN           NaN
005b9685-1114-e183-1b69-45964c           NaN         87.11         27.27
005d3e80-f153-c133-8dea-4e5b66         88.97         98.64         91.18
0065897d-5ddf-d164-00c2-550e2e        100.00           NaN           NaN
00a5a86d-343d-3a27-e9ab-be6f4f           NaN           NaN         83.53
00b795d6-f7b8-7904-f685-bef7d7         89.66         91.59        100.00
00d56c28-1c61-e176-fbd4-6acbac         89.31         81.82         66.67
00f46499-ad02-18ae-e634-9b5613         87.93         94.77         87.94
00fd6e8f-8ace-3e32-1b47-d9d0ed        100.00         98.64        100.00
```

```
012a5010-4041-71ac-c5f3-ec0ced        100.00         53.64        100.00

assignment_name                  Milestone #4  Milestone #5
user_id
0034a903-0652-77b3-9c70-82b91a            NaN           NaN
005b9685-1114-e183-1b69-45964c            NaN           NaN
005d3e80-f153-c133-8dea-4e5b66         100.00         93.70
0065897d-5ddf-d164-00c2-550e2e            NaN           NaN
00a5a86d-343d-3a27-e9ab-be6f4f          70.97         61.85
00b795d6-f7b8-7904-f685-bef7d7          94.19        100.00
00d56c28-1c61-e176-fbd4-6acbac          77.42         98.89
00f46499-ad02-18ae-e634-9b5613          95.16         90.37
00fd6e8f-8ace-3e32-1b47-d9d0ed          96.77         96.30
012a5010-4041-71ac-c5f3-ec0ced          90.00         97.04
```

[42]:
```
# Extract test data required for graduation
tests = df[df['assignment_name'].str.contains("Weekly Test #", na=False)]

# Ensure users have an average test score   75%
test_scores = tests.pivot(index='user_id', columns='assignment_name',␣
 ↪values='assignment_score').fillna(0)
test_avg = test_scores.mean(axis=1)  # Compute average score
test_criteria = test_avg >= 75  # True if average test score is at least 75%
```

[ ]:
```
test_criteria
```

[ ]:
```
user_id
0034a903-0652-77b3-9c70-82b91a    False
005a2866-1063-8f09-bf83-1cb6f4    False
005b9685-1114-e183-1b69-45964c    False
005d3e80-f153-c133-8dea-4e5b66     True
005f8882-251f-a015-ff42-c8e80c    False
                                  …
ffc80062-9d1b-3d92-6647-d46d08    False
ffd40f70-6761-dabd-b0b7-2c486c    False
ffd68c7b-8edc-9e0b-45f6-168767     True
fff8b323-3e76-e67d-cf1f-8390fb    False
fff94200-23ac-3b93-fca0-178543     True
Length: 5041, dtype: bool
```

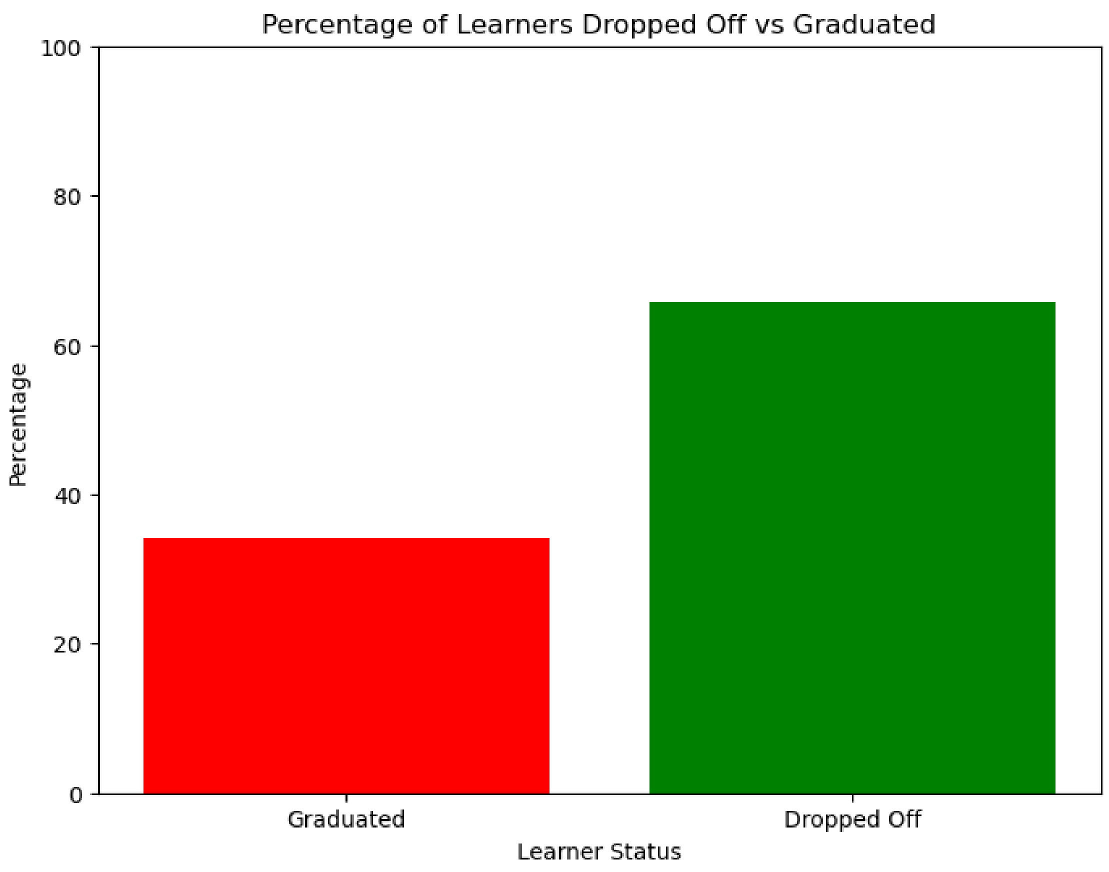
[43]:
```
# Apply both conditions to determine valid graduates
graduation_status = (users_with_5_milestones & test_criteria).astype(int)

# Assign graduation status to the original DataFrame
df['graduated'] = df['user_id'].map(graduation_status).fillna(0).astype(int)

print("Graduation status updated.")
```

```
Graduation status updated.
```

[44]:
```python
# Count unique graduates
num_graduated = df["user_id"][df["graduated"] == 1].nunique()
print(f"Total number of users who graduated: {num_graduated}")
```

```
Total number of users who graduated: 1712
```

[30]:
```python
# Calculate percentages
total_learners = 5002
dropped_off = 3290
graduated = 1712

dropped_off_percentage = (dropped_off / total_learners) * 100
graduated_percentage = (graduated / total_learners) * 100

# Data for plotting
labels = ['Graduated', 'Dropped Off']
percentages = [graduated_percentage, dropped_off_percentage]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(labels, percentages, color=['red', 'green'])
plt.xlabel('Learner Status')
plt.ylabel('Percentage')
plt.title('Percentage of Learners Dropped Off vs Graduated')
plt.ylim(0, 100)  # Set y-axis limit to 100%
plt.show();
```



Less than 40% of the total learners graduated

```
[ ]: # Method 2 to identify valid graduates
     test_criteria_df = test_criteria.to_frame(name='test_passed')  # Test criteria
     f_mile_scores_df = f_mile_scores.to_frame(name='milestone_completed')  #␣
      ↪Milestone criteria

     # Merge both graduation criteria
     final_df = pd.merge(test_criteria_df, f_mile_scores_df, left_index=True,␣
      ↪right_index=True, how='inner')
```

```
[ ]: final_df
```

```
[ ]:                                  test_passed  milestone_completed
     user_id
     0034a903-0652-77b3-9c70-82b91a        False                False
     005b9685-1114-e183-1b69-45964c        False                False
     005d3e80-f153-c133-8dea-4e5b66         True                 True
     0065897d-5ddf-d164-00c2-550e2e        False                False
     00a5a86d-343d-3a27-e9ab-be6f4f         True                False
```

```
…                                      …                    …
ffbaaeb2-ab3c-03d3-a765-b5d611         True                 True
ffd40f70-6761-dabd-b0b7-2c486c        False                False
ffd68c7b-8edc-9e0b-45f6-168767         True                 True
fff8b323-3e76-e67d-cf1f-8390fb        False                False
fff94200-23ac-3b93-fca0-178543         True                 True

[2540 rows x 2 columns]
```

```
[ ]: # Check for learners who meet both conditions
     final_df = final_df[(final_df['test_passed'] == True ) &␣
      ↪(final_df['milestone_completed'] == True) ]
```

```
[ ]: # Obtain user_id of graduating learners
     f_user_id = final_df.index
     f_user_id
```

```
[ ]: Index(['005d3e80-f153-c133-8dea-4e5b66', '00b795d6-f7b8-7904-f685-bef7d7',
            '00d56c28-1c61-e176-fbd4-6acbac', '00f46499-ad02-18ae-e634-9b5613',
            '00fd6e8f-8ace-3e32-1b47-d9d0ed', '012a5010-4041-71ac-c5f3-ec0ced',
            '014c1879-2025-1109-297f-57e7d8', '01695c10-c70c-9ad3-c211-573a88',
            '019fe0fc-c55c-1481-1170-ea7bb2', '01c6cef4-187b-9d11-efa8-d6fdbb',
            …
            'fed0ceb6-f5f7-273a-ba35-7fc573', 'fefa6261-1ce8-f1bb-71ef-ae8ca0',
            'ff2d5b94-dab8-0977-a4dc-c92ecd', 'ff470390-9fad-adbd-40b3-391bb3',
            'ff4e42e6-d026-25b6-cecf-5471b9', 'ff88bf17-7269-e3ad-cfd5-a4a622',
            'ffb0ecf0-eb0d-85c0-0d8b-aac797', 'ffbaaeb2-ab3c-03d3-a765-b5d611',
            'ffd68c7b-8edc-9e0b-45f6-168767', 'fff94200-23ac-3b93-fca0-178543'],
           dtype='object', name='user_id', length=1712)
```

```
[ ]: # Populate 'graduate' column based on user_id present in f_user_id
     df['graduate'] = df['user_id'].isin(f_user_id).astype(int)

     # Count unique graduates
     unique_graduates = df["user_id"][df["graduate"] == 1].nunique()
     print(f"Total number of unique graduates: {unique_graduates}")
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_11800\2048875875.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['graduate'] = df['user_id'].isin(f_user_id).astype(int)
```

[ ]:
```
# Validation output for both methods used
mismatch_count = (df['graduated'] != df['graduate']).sum()
print(mismatch_count)
```

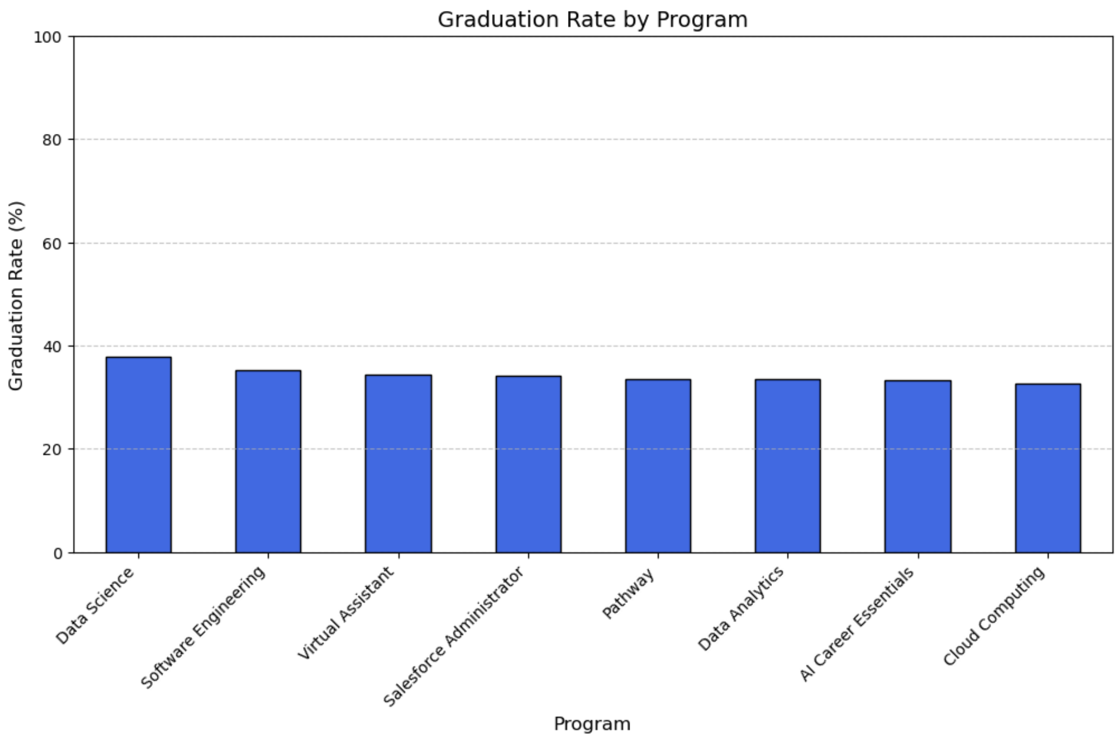
```
0
```

Both methods yielded the exact result for graduating learners.

[50]:
```
# Group by program and compute graduation rate
graduation_rates = df.groupby('program')['graduated'].mean() * 100  # Convert␣
↪to percentage

# Sort values for better visualization
graduation_rates = graduation_rates.sort_values(ascending=False)

print(graduation_rates)  # Check the computed graduation rates
```

```
program
Data Science                37.730871
Software Engineering        35.270806
Virtual Assistant           34.274953
Salesforce Administrator    34.239130
Pathway                     33.522727
Data Analytics              33.429672
AI Career Essentials        33.333333
Cloud Computing             32.647059
Name: graduated, dtype: float64
```

[51]:
```
plt.figure(figsize=(12, 6))  # Set figure size
graduation_rates.plot(kind='bar', color='royalblue', edgecolor='black')

plt.title("Graduation Rate by Program", fontsize=14)
plt.xlabel("Program", fontsize=12)
plt.ylabel("Graduation Rate (%)", fontsize=12)
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.ylim(0, 100)  # Ensure the y-axis shows percentages properly

plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add gridlines for better␣
↪readability
plt.show();
```



The graduation rate by program chart reveals that the program is not a