Start coding or generate with AI.

```
from google.colab import files
uploaded = files.upload()
```

Choose Files course_enr...nt_data.csv

- **course_enrollment_data.csv**(text/csv) - 902 bytes, last modified: 8/3/2025 - 100% done

```
Saving course_enrollment_data.csv to course_enrollment_data.csv
```

```
import pandas as pd

# Replace 'your_file.csv' with your actual filename
df = pd.read_csv('course_enrollment_data.csv')

# Preview first few rows
df.head()
```

	enrollment_id	student_name	course_name	instructor_name	enrollment_date	completion_status	course_fee	grade
0	101	Ananya Sharma	Data Science Basics	Dr. Meera Rao	2023-01-15	Completed	1200.0	A
1	102	Ravi Verma	Python for Beginners	Prof. Raj Malhotra	2023-02-10	Ongoing	800.0	NaN
2	103	Priya Singh	Machine Learning	Dr. Tanvi Desai	2023-03-05	Completed	1500.0	B

Next steps: Generate code with df | View recommended plots | New interactive sheet

Start coding or generate with AI.

Task

Analyze the data in the uploaded CSV file "data.csv" using Python. The analysis should include data loading and preview, cleaning (handling missing values), descriptive statistics, various visualizations (bar charts, pie charts, line charts, correlation heatmap, histograms), analysis of patterns, trends, and outliers, and a summary of 5 key insights. The final output should be visually appealing and well-structured.

Load and preview data

Subtask:

Load the dataset into a pandas DataFrame and display the first few rows to understand its structure.

Data cleaning

Subtask:

Check for missing values and handle them appropriately (e.g., imputation, dropping rows/columns).

Reasoning: Check for missing values in each column of the DataFrame.

```
print("Missing values before handling:")
print(df.isnull().sum())
```

```
Missing values before handling:
enrollment_id        0
student_name         0
course_name          0
instructor_name      0
enrollment_date      0
```

Gemini

Generating...

Preparing

Load and

```
completion_status    0
course_fee           0
grade                5
dtype: int64
```

Reasoning: Handle the missing values in the 'grade' column by filling them with a placeholder value.

```
df['grade'] = df['grade'].fillna('Not Available')
print("\nMissing values after handling:")
print(df.isnull().sum())
display(df.head())
```

```
Missing values after handling:
enrollment_id        0
student_name         0
course_name          0
instructor_name      0
enrollment_date      0
completion_status    0
course_fee           0
grade                0
dtype: int64
```

	enrollment_id	student_name	course_name	instructor_name	enrollment_date	completion_status	course_fee	grac
0	101	Ananya Sharma	Data Science Basics	Dr. Meera Rao	2023-01-15	Completed	1200.0	
1	102	Ravi Verma	Python for Beginners	Prof. Raj Malhotra	2023-02-10	Ongoing	800.0	N Availab
2	103	Priya Singh	Machine Learning	Dr. Tanvi Desai	2023-03-05	Completed	1500.0	

Descriptive statistics

Subtask:

Generate summary statistics for numerical and categorical columns to understand the data distribution.

Reasoning: Generate descriptive statistics for both numerical and categorical columns to understand the data distribution and characteristics.

```
print("Descriptive statistics for numerical columns:")
display(df.describe())

print("\nDescriptive statistics for categorical columns:")
display(df.describe(include='all'))
```

```
Descriptive statistics for numerical columns:
```

	enrollment_id	course_fee
count	10.00000	10.000000
mean	105.50000	1085.000000
std	3.02765	262.519841
min	101.00000	700.000000
25%	103.25000	912.500000
50%	105.50000	1050.000000
75%	107.75000	1275.000000
max	110.00000	1500.000000

```
Descriptive statistics for categorical columns:
```

	enrollment_id	student_name	course_name	instructor_name	enrollment_date	completion_status	course_fee
count	10.00000	10	10	10	10	10	10.000000
unique	NaN	10	10	10	10	3	NaN
top	NaN	Ananya Sharma	Data Science Basics	Dr. Meera Rao	2023-01-15	Completed	NaN
freq	NaN	1	1	1	1	5	NaN
mean	105.50000	NaN	NaN	NaN	NaN	NaN	1085.000000
std	3.02765	NaN	NaN	NaN	NaN	NaN	262.519841
min	101.00000	NaN	NaN	NaN	NaN	NaN	700.000000
25%	103.25000	NaN	NaN	NaN	NaN	NaN	912.500000
50%	105.50000	NaN	NaN	NaN	NaN	NaN	1050.000000
75%	107.75000	NaN	NaN	NaN	NaN	NaN	1275.000000

Data visualization

Subtask:

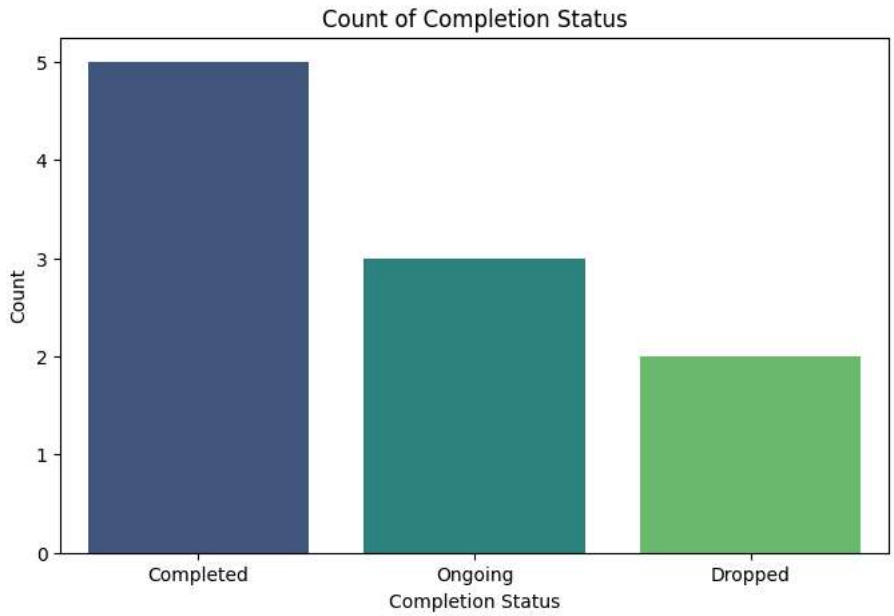
Create various plots (bar charts, pie charts, line charts, histograms) to visualize different aspects of the data.

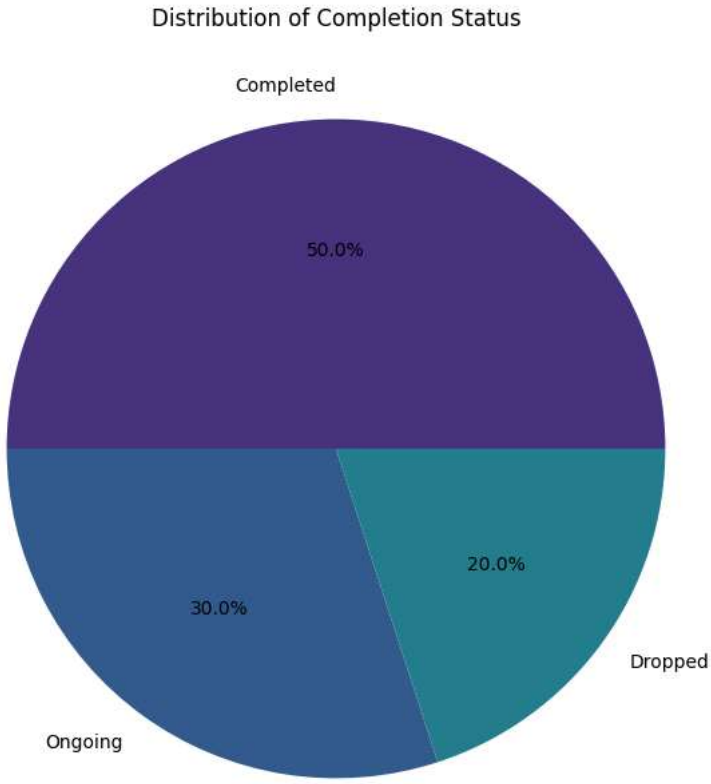
Reasoning: Generate the specified plots to visualize different aspects of the data.

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

# 1. Bar chart of completion_status counts
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='completion_status', palette='viridis')
plt.title('Count of Completion Status')
plt.xlabel('Completion Status')
plt.ylabel('Count')
plt.show()

# 2. Pie chart of completion_status distribution
completion_counts = df['completion_status'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(completion_counts, labels=completion_counts.index, autopct='%1.1f%%', colors=sns.color_palette('viridis'))
plt.title('Distribution of Completion Status')
plt.show()

# 3. Bar chart of grade counts (excluding 'Not Available')
grade_counts = df[df['grade'] != 'Not Available']['grade'].value_counts()
plt.figure(figsize=(8, 5))
sns.countplot(data=df[df['grade'] != 'Not Available'], x='grade', order=grade_counts.index, palette='viridis')
plt.title('Count of Grades (Excluding Not Available)')
plt.xlabel('Grade')
plt.ylabel('Count')
plt.show()
```

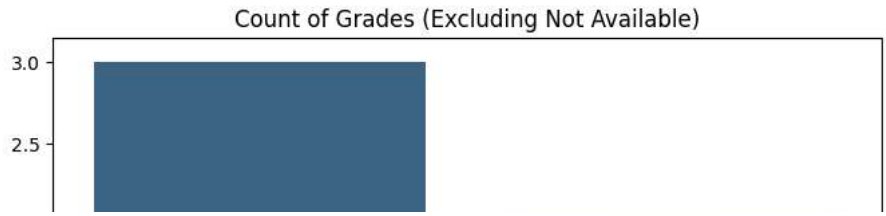
```
# 4. Histogram of course_fee
plt.figure(figsize=(8, 5))
sns.histplot(data=df, x='course_fee', kde=True, bins=10, color='skyblue')
plt.title('Distribution of Course Fee')
plt.xlabel('Course Fee')
plt.ylabel('Frequency')
plt.show()

# 5. Bar chart of average course_fee for each completion_status
avg_fee_by_completion = df.groupby('completion_status')['course_fee'].mean().reset_index()
plt.figure(figsize=(8, 5))
sns.barplot(data=avg_fee_by_completion, x='completion_status', y='course_fee', palette='viridis')
plt.title('Average Course Fee by Completion Status')
plt.xlabel('Completion Status')
plt.ylabel('Average Course Fee')
plt.show()

# 6. Bar chart of enrollment counts per course_name
course_counts = df['course_name'].value_counts()
plt.figure(figsize=(10, 6))
sns.countplot(data=df, y='course_name', order=course_counts.index, palette='viridis')
plt.title('Enrollment Count per Course Name')
plt.xlabel('Count')
plt.ylabel('Course Name')
plt.show()

# 7. Bar chart of enrollment counts per instructor_name
instructor_counts = df['instructor_name'].value_counts()
plt.figure(figsize=(10, 6))
sns.countplot(data=df, y='instructor_name', order=instructor_counts.index, palette='viridis')
plt.title('Enrollment Count per Instructor Name')
plt.xlabel('Count')
plt.ylabel('Instructor Name')
plt.show()
```

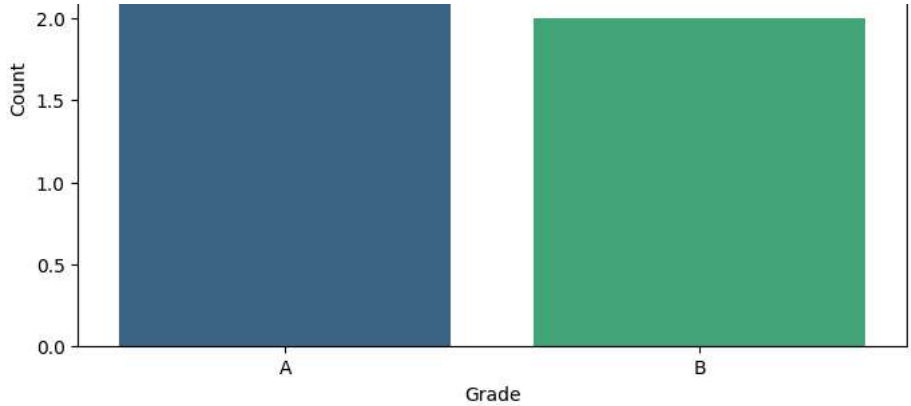
```
/tmp/ipython-input-3789579997.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to

  sns.countplot(data=df, x='completion_status', palette='viridis')
```

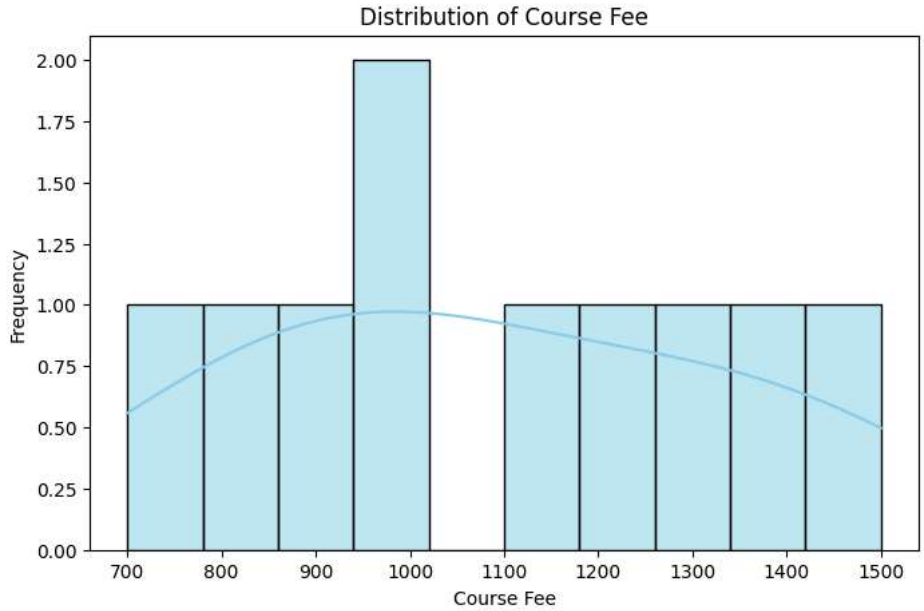




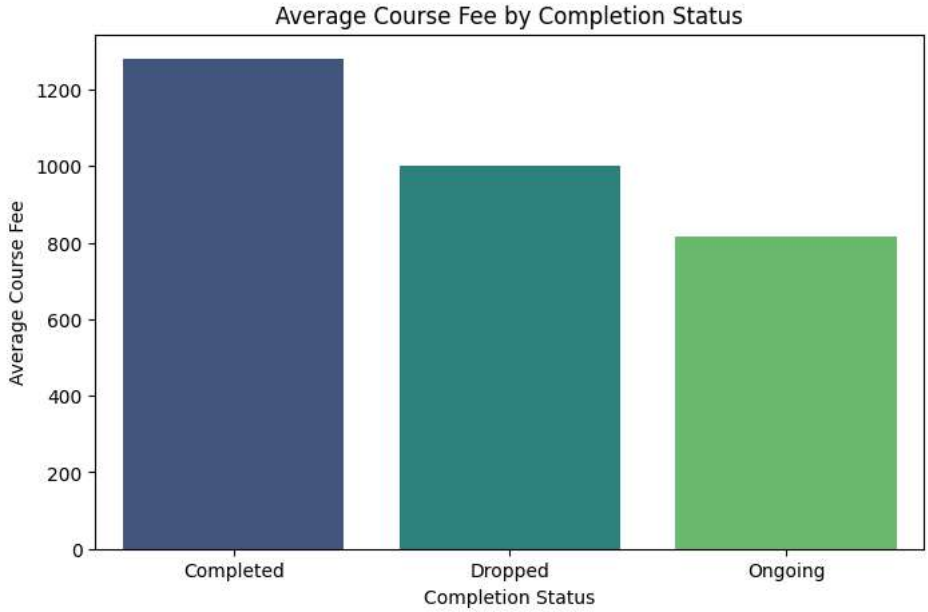

```
/tmp/ipython-input-3789579997.py:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to

  sns.countplot(data=df[df['grade'] != 'Not Available'], x='grade', order=grade_counts.index, palette='viridis')
```



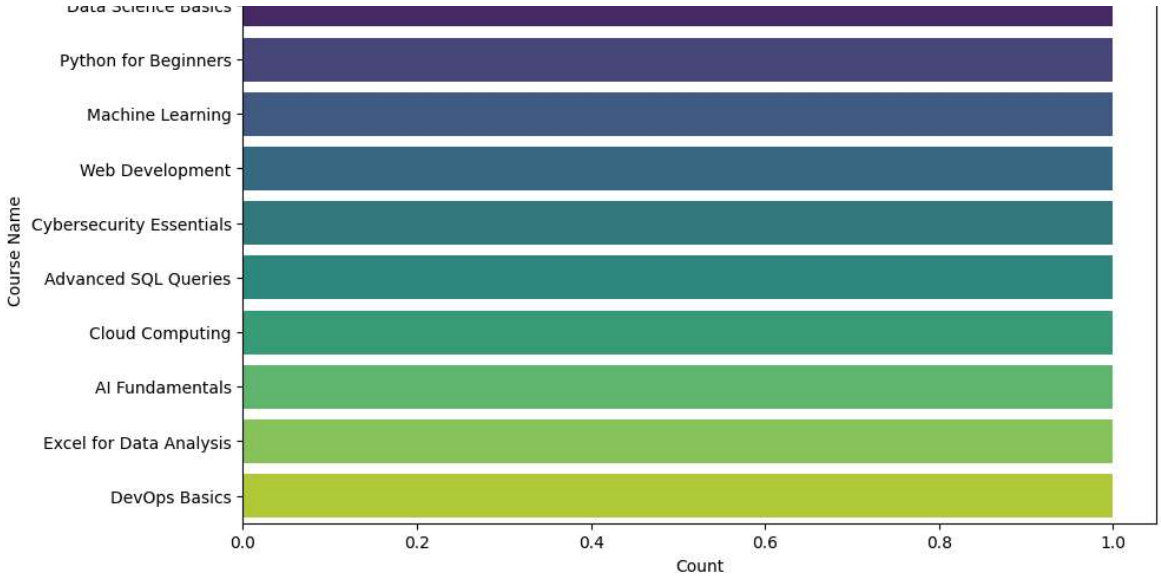





```
/tmp/ipython-input-3789579997.py:39: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to

  sns.barplot(data=avg_fee_by_completion, x='completion_status', y='course_fee', palette='viridis')
```




```
/tmp/ipython-input-3789579997.py:48: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to

  sns.countplot(data=df, y='course_name', order=course_counts.index, palette='viridis')
```




```
/tmp/ipython-input-3789579997.py:57: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to

  sns.countplot(data=df, y='instructor_name', order=instructor_counts.index, palette='viridis')
```