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

```
Choose Files Breast_can...dataset.csv
• Breast_cancer_dataset.csv(text/csv) - 125204 bytes, last modified: 8/3/2025 - 100% done
Saving Breast_cancer_dataset.csv to Breast_cancer_dataset.csv
```

```
import pandas as pd

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

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

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.1

5 rows × 33 columns

```
Start coding or generate with AI.
```

Task

Analyze the data in the provided CSV file "data.csv" using Python. The analysis should include: importing necessary libraries (Pandas, NumPy, Matplotlib, Seaborn), loading and previewing the dataset, checking for and handling missing values, describing the data with summary statistics, creating various visualizations (bar charts, pie charts, line charts, correlation heatmap, histograms), analyzing patterns, trends, and outliers, and writing 5 key insights from the data. The output should be visually appealing and well-structured.

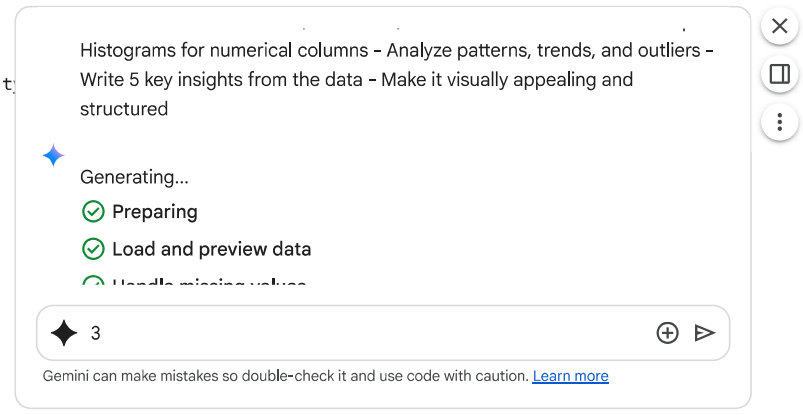
Load and preview data

Subtask:

Load the dataset using pandas and display the first few rows and column information.

Reasoning: Load the dataset, display the head and info of the dataframe to fulfill the subtask.

```
# Display the first 5 rows
display(df.head())

# Display column names and data t
df.info()
```



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	con points_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.1

5 rows × 33 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Handle missing values

Subtask:

Check for missing values and decide on an appropriate strategy to handle them (e.g., imputation, removal).

Reasoning: Check for missing values in each column and then drop the column with all missing values.

```
# Check for missing values in each column
missing_values = df.isnull().sum()

# Display the sum of missing values for each column
print("Missing values per column:")
print(missing_values)

# Drop the column 'Unnamed: 32' as it contains all missing values
df = df.drop('Unnamed: 32', axis=1)

# Verify that the column 'Unnamed: 32' has been removed
df.info()
```

```
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

Describe data

Subtask:

Generate summary statistics for numerical and categorical columns.

Reasoning: Generate descriptive statistics for both numerical and categorical columns and print the results.

```
# Generate descriptive statistics for numerical columns
numerical_summary = df.describe()
print("Descriptive statistics for numerical columns:")
display(numerical_summary)

# Generate descriptive statistics for categorical columns, including object type columns
categorical_summary = df.describe(include='object')
print("\nDescriptive statistics for categorical columns:")
display(categorical_summary)
```

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

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_me
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.0000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.0489
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.0388
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.0000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.0203
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.0335
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.0740
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.2012

8 rows × 31 columns

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

	diagnosis
count	569
unique	2
top	B
freq	357

Next steps: Generate code with `categorical_summary` View recommended plots New interactive sheet

Visualize data

Subtask:

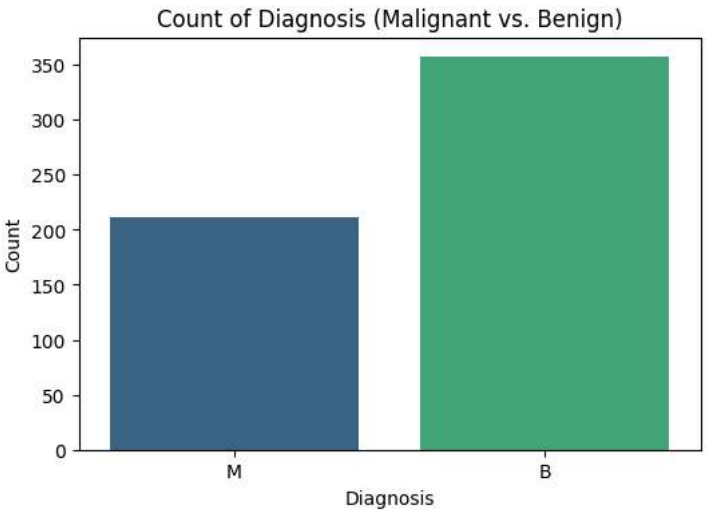
Create various plots including bar charts, pie charts, line charts, correlation heatmap, and histograms.

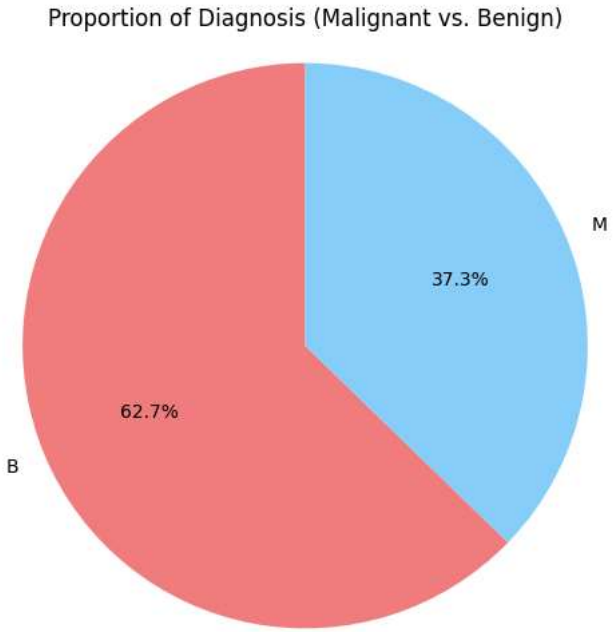
Reasoning: Create visualizations as requested, including bar chart, pie chart, histograms, and a correlation heatmap. Assess the suitability of line charts for this dataset.

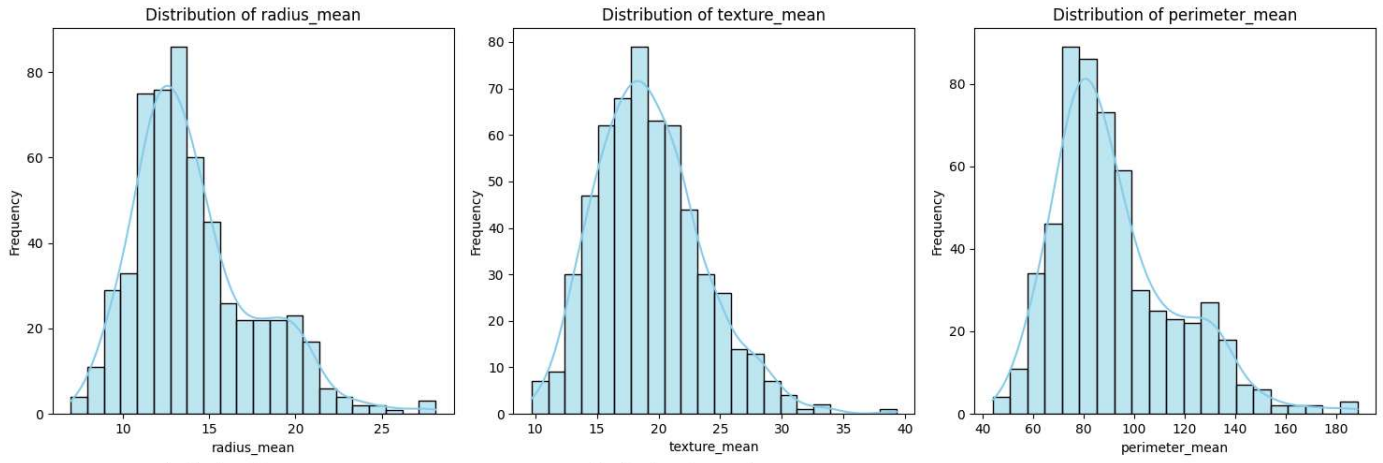
```
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Bar chart for diagnosis counts
plt.figure(figsize=(6, 4))
sns.countplot(x='diagnosis', data=df, palette='viridis')
plt.title('Count of Diagnosis (Malignant vs. Benign)')
plt.xlabel('Diagnosis')
plt.ylabel('Count')
plt.show()

# 2. Pie chart for diagnosis proportions
diagnosis_counts = df['diagnosis'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(diagnosis_counts, labels=diagnosis_counts.index, autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue'], startangle=90)
plt.title('Proportion of Diagnosis (Malignant vs. Benign)')
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()

# 3. Histograms for a selection of numerical features
numerical_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean']
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_features):
    plt.subplot(2, 3, i + 1)
    sns.histplot(df[col], kde=True, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

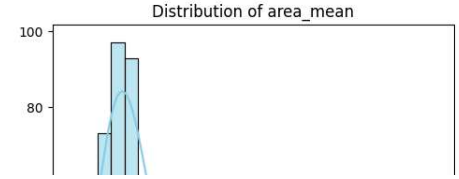
```
# 4. Correlation heatmap
# Exclude non-numerical columns like 'id' and 'diagnosis'
numerical_df = df.select_dtypes(include=['float64', 'int64']).drop(columns=['id'])
plt.figure(figsize=(12, 10))
sns.heatmap(numerical_df.corr(), annot=False, cmap='coolwarm')
plt.title('Correlation Heatmap of Numerical Features')
plt.show()

# 5. Assess suitability of line charts
print("\nAssessment of Line Charts:")
print("Line charts are typically used to show trends over time or sequential data.")
print("This dataset contains measurements taken at a single point in time for each patient, not time-series data.")
print("Therefore, line charts are not appropriate for visualizing the relationships or distributions in this dataset.")
```

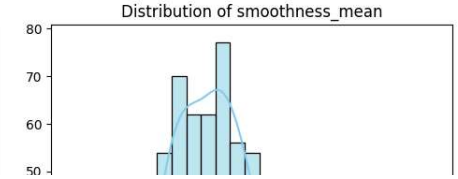
```
/tmp/ipython-input-3584067087.py:6: FutureWarning:

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

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

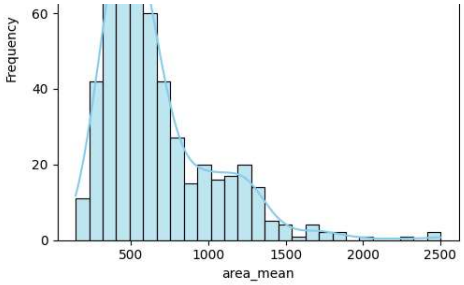


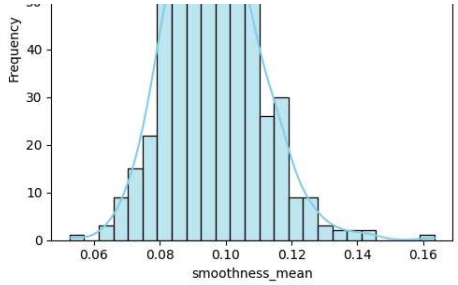


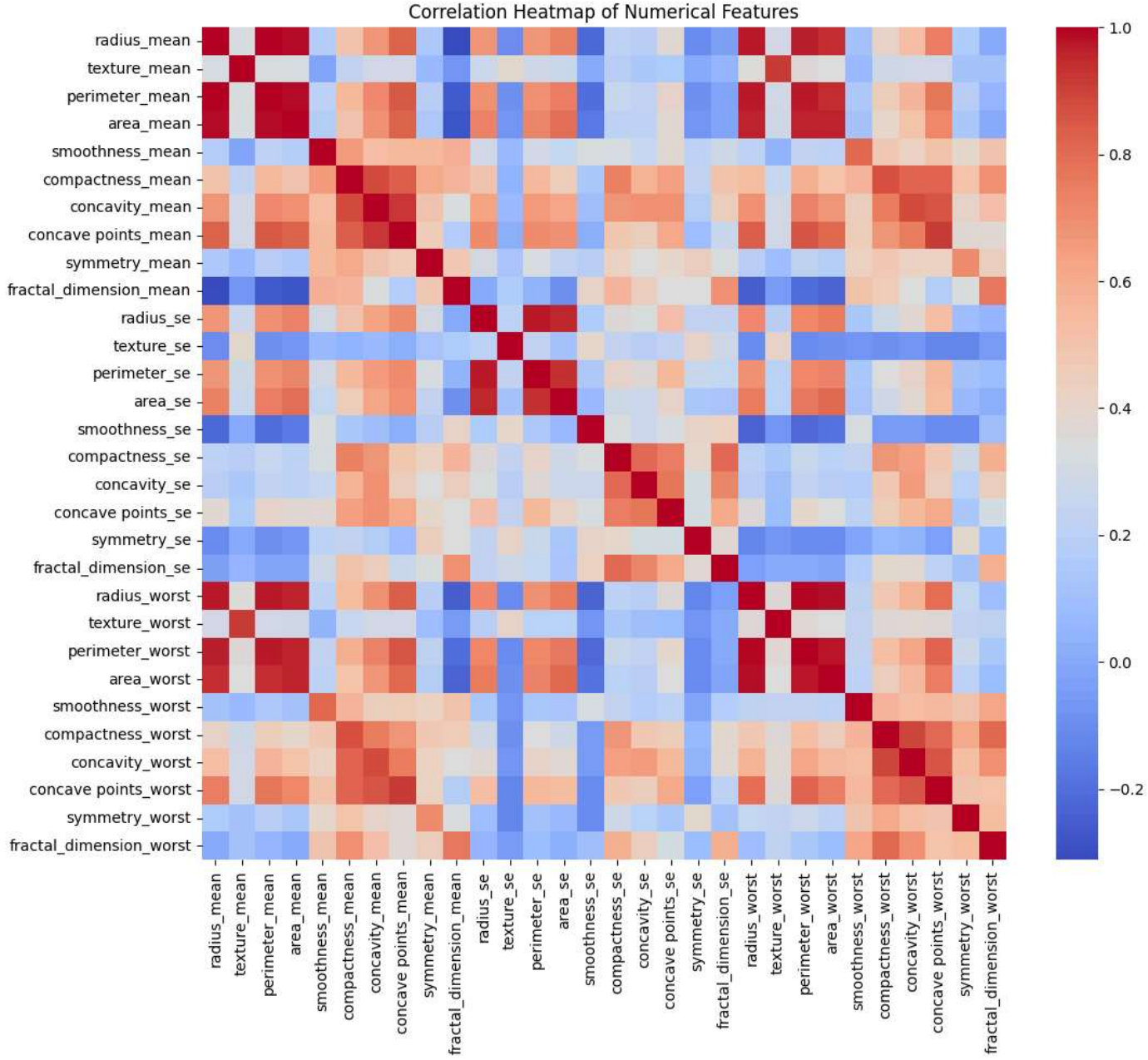













```
Assessment of Line Charts:
Line charts are typically used to show trends over time or sequential data.
This dataset contains measurements taken at a single point in time for each patient, not time-series data.
Therefore, line charts are not appropriate for visualizing the relationships or distributions in this dataset.
```