Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

PRACTICAL – 7

Aim: Perform the following Data Preparation task on any of the data
- **Check the correlation between various columns**
- **Check the skewness and kurtosis of data**

```
[1]: import pandas as pd
```

```
[2]: sample=pd.read_csv("Sample - Superstore - Sample - Superstore.csv")
```

```
[3]: sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Order ID       9994 non-null   object
 1   Order Date     9994 non-null   object
 2   Ship Date      9994 non-null   object
 3   Ship Mode      9994 non-null   object
 4   Customer ID    9994 non-null   object
 5   Customer Name  9994 non-null   object
 6   Segment        9994 non-null   object
 7   Country        9994 non-null   object
 8   City           9994 non-null   object
 9   State          9994 non-null   object
 10  Postal Code    9994 non-null   int64
 11  Region         9994 non-null   object
 12  Product ID     9994 non-null   object
 13  Category       9994 non-null   object
 14  Sub-Category   9994 non-null   object
 15  Product Name   9994 non-null   object
 16  Sales          9994 non-null   float64
 17  Quantity       9994 non-null   int64
 18  Discount       9994 non-null   float64
 19  Profit         9994 non-null   float64
dtypes: float64(3), int64(2), object(15)
memory usage: 1.5+ MB
```

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

```
[4]: sample.columns
```

```
[4]: Index(['Order ID', 'Order Date', 'Ship Date', 'Ship Mode', 'Customer ID',
            'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code',
            'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name',
            'Sales', 'Quantity', 'Discount', 'Profit'],
           dtype='object')
```

ANALYTICAL STATEMENT:

```
[5]: sample['Sales'].corr(sample['Quantity'])
```

```
[5]: np.float64(0.2007947713738976)
```

ANALYTICAL STATEMENT:

```
[6]: sample['Quantity'].corr(sample['Profit'])
```

```
[6]: np.float64(0.06625318912428485)
```

ANALYTICAL STATEMENT:

```
[7]: sample[['Profit','Discount']].kurt()
```

```
[7]: Profit      397.188515
     Discount      2.409546
     dtype: float64
```

ANALYTICAL STATEMENT:

```
[8]: sample['Discount'].kurt()
```

```
[8]: np.float64(2.4095461225966774)
```

ANALYTICAL STATEMENT:

```
[9]: sample['Sales'].kurt()
```

```
[9]: np.float64(305.311753246823)
```

ANALYTICAL STATEMENT:

```
[10]: sample['Profit'].skew()
```

```
[10]: np.float64(7.561431562468343)
```

ANALYTICAL STATEMENT:

```
[11]: sample['Sales'].skew()
```

```
[11]: np.float64(12.97275234181623)
```

ANALYTICAL STATEMENT:

```
[12]: sample['Discount'].skew()
```

```
[12]: np.float64(1.6842947474238648)
```

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

PRACTICAL – 8

Aim: Perform the Data Transformation on date time and zip code feature.

```
import pandas as pd
data  = pd.read_csv("Loan.csv")
```

```
[3]: data.head()
```

[3]:

	customer_id	disbursed_amount	interest	market	employment	time_employed	householder	income	date_issued	target	loan_purpose	number_open_accounts	d
0	0	23201.5	15.4840	C	Teacher	<=5 years	RENT	84600.0	2013-06-11	0	Debt consolidation	4	
1	1	7425.0	11.2032	B	Accountant	<=5 years	OWNER	102000.0	2014-05-08	0	Car purchase	13	
2	2	11150.0	8.5100	A	Statistician	<=5 years	RENT	69840.0	2013-10-26	0	Debt consolidation	8	
3	3	7600.0	5.8656	A	Other	<=5 years	RENT	100386.0	2015-08-20	0	Debt consolidation	20	
4	4	31960.0	18.7392	E	Bus driver	>5 years	RENT	95040.0	2014-07-22	0	Debt consolidation	14	

ANALYTICAL STATEMENT:

```
[4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   customer_id            10000 non-null  int64  
 1   disbursed_amount       10000 non-null  float64
 2   interest               10000 non-null  float64
 3   market                 10000 non-null  object 
 4   employment             9389 non-null   object 
 5   time_employed          9471 non-null   object 
 6   householder            10000 non-null  object 
 7   income                 10000 non-null  float64
 8   date_issued            10000 non-null  object 
 9   target                 10000 non-null  int64  
 10  loan_purpose           10000 non-null  object 
 11  number_open_accounts   10000 non-null  int64  
 12  date_last_payment      10000 non-null  object 
 13  number_credit_lines_12  238 non-null    float64
dtypes: float64(4), int64(3), object(7)
memory usage: 1.1+ MB
```

ANALYTICAL STATEMENT:

```
data['Date'] = pd.to_datetime(data['date_issued'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   customer_id            10000 non-null  int64
 1   disbursed_amount       10000 non-null  float64
 2   interest               10000 non-null  float64
 3   market                 10000 non-null  object
 4   employment             9389 non-null   object
 5   time_employed          9471 non-null   object
 6   householder            10000 non-null  object
 7   income                 10000 non-null  float64
 8   date_issued            10000 non-null  object
 9   target                 10000 non-null  int64
 10  loan_purpose           10000 non-null  object
 11  number_open_accounts   10000 non-null  int64
 12  date_last_payment      10000 non-null  object
 13  number_credit_lines_12 238 non-null    float64
 14  Date                   10000 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(4), int64(3), object(7)
memory usage: 1.1+ MB
```

ANALYTICAL STATEMENT:

[6]:
```
data['Month'] = data['Date'].dt.month
data['Day'] = data['Date'].dt.day
data['Year'] = data['Date'].dt.year
data[['Month','Day','Year']].head()
```

[6]:

	Month	Day	Year
0	6	11	2013
1	5	8	2014
2	10	26	2013
3	8	20	2015
4	7	22	2014

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[7]:
```
data['day_of_week'] = data['Date'].dt.day_of_week
data['day_of_year'] = data['Date'].dt.day_of_year
```

[8]:
```
data[['Year','day_of_week','day_of_year']].head()
```

[8]:

	Year	day_of_week	day_of_year
0	2013	1	162
1	2014	3	128
2	2013	5	299
3	2015	3	232
4	2014	1	203

ANALYTICAL STATEMENT:

[9]:
```
def week_part(day):
    if day in [1,2,3,4,5,6,7]:
        return "week 1"
    elif day in [8,9,10,11,12,13,14]:
        return "week 2"
    elif day in [15,16,17,18,19,20,21]:
        return "week 3"
    elif day in [22,23,24,25,26,27,28]:
        return "week 4"
    elif day in [29,30,31]:
        return "week 5"
```

ANALYTICAL STATEMENT:

[10]:
```
data['Week_No'] = data['Day'].apply(week_part)
data[['Day','Week_No']]
```

[10]:

	Day	Week_No
0	11	week 2
1	8	week 2
2	26	week 4
3	20	week 3
4	22	week 4
...	...	...
9995	14	week 2
9996	20	week 3
9997	3	week 1
9998	23	week 4
9999	19	week 3

10000 rows × 2 columns

ANALYTICAL STATEMENT:

[11]:
```
data['Week_No'].value_counts()
```

[11]:
```
week 1    2652
week 3    2591
week 2    2545
week 4    2212
Name: Week_No, dtype: int64
```

ANALYTICAL STATEMENT:

[12]:
```
import numpy as np
data["date issued:is_weekend"] = np.where(data["day_of_week"].isin([5,6]),1,0)
data[['Date','day_of_week','date issued:is_weekend']].head()
```

[12]:

	Date	day_of_week	date issued:is_weekend
0	2013-06-11	1	0
1	2014-05-08	3	0
2	2013-10-26	5	1
3	2015-08-20	3	0
4	2014-07-22	1	0

ANALYTICAL STATEMENT:

[13]:
```
data['is_leap_year'] = data['Date'].dt.is_leap_year
data[['Date','is_leap_year']]
```

[13]:

	Date	is_leap_year
0	2013-06-11	False
1	2014-05-08	False
2	2013-10-26	False
3	2015-08-20	False
4	2014-07-22	False
...	...	...
9995	2010-01-14	False
9996	2015-03-20	False
9997	2015-04-03	False
9998	2014-11-23	False
9999	2015-01-19	False

10000 rows × 2 columns

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[14]:
```
data['is_leap_year'].value_counts()
```

[14]:
```
False    9385
True      615
Name: is_leap_year, dtype: int64
```

ANALYTICAL STATEMENT:

[15]:
```
data['Date'].min(), data['Date'].max()
```

[15]:
```
(Timestamp('2007-07-10 00:00:00'), Timestamp('2015-12-27 00:00:00'))
```

ANALYTICAL STATEMENT:

[16]:
```
data['Date'].max() - data['Date'].min()
```

[16]:
```
Timedelta('3092 days 00:00:00')
```

ANALYTICAL STATEMENT:

[17]:
```
data['dt_period'] = data['Date'].dt.to_period('Y')
data.head()
```

[17]:

ne_employed	householder	income	date_issued	target	...	Date	Month	Day	Year	day_of_week	day_of_year	Week_No	date issued:is_weekend	is_leap_year	dt_period
<=5 years	RENT	84600.0	2013-06-11	0	...	2013-06-11	6	11	2013	1	162	week 2	0	False	2013
<=5 years	OWNER	102000.0	2014-05-08	0	...	2014-05-08	5	8	2014	3	128	week 2	0	False	2014
<=5 years	RENT	69840.0	2013-10-26	0	...	2013-10-26	10	26	2013	5	299	week 4	1	False	2013
<=5 years	RENT	100386.0	2015-08-20	0	...	2015-08-20	8	20	2015	3	232	week 3	0	False	2015
>5 years	RENT	95040.0	2014-07-22	0	...	2014-07-22	7	22	2014	1	203	week 4	0	False	2014

ANALYTICAL STATEMENT:

[18]:
```
data['next_15_days'] = data['Date'] + pd.Timedelta(days=15)
data[['Date','next_15_days']].head()
```

[18]:

	Date	next_15_days
0	2013-06-11	2013-06-26
1	2014-05-08	2014-05-23
2	2013-10-26	2013-11-10
3	2015-08-20	2015-09-04
4	2014-07-22	2014-08-06

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[19]:
```
data['date_issued:is_year_start'] = data['Date'].dt.is_year_start
data['date_issued:is_quarter_start'] = data['Date'].dt.is_quarter_start
data['date_issued:is_month_start'] = data['Date'].dt.is_month_start
data['date_issued:is_year_end'] = data['Date'].dt.is_month_end
data[['date_issued','date_issued:is_year_start','date_issued:is_quarter_start',
    'date_issued:is_month_start','date_issued:is_year_end']].head(15)
```

[19]:

	date_issued	date_issued:is_year_start	date_issued:is_quarter_start	date_issued:is_month_start	date_issued:is_year_end
0	2013-06-11	False	False	False	False
1	2014-05-08	False	False	False	False
2	2013-10-26	False	False	False	False
3	2015-08-20	False	False	False	False
4	2014-07-22	False	False	False	False
5	2013-08-21	False	False	False	False
6	2015-09-27	False	False	False	False
7	2015-03-20	False	False	False	False
8	2014-02-14	False	False	False	False
9	2013-12-25	False	False	False	False
10	2015-11-22	False	False	False	False
11	2014-04-04	False	False	False	False
12	2015-10-26	False	False	False	False
13	2015-11-13	False	False	False	False
14	2015-04-23	False	False	False	False

ANALYTICAL STATEMENT:

[21]:
```
data['date_issued:is_year_start'].value_counts()
```

[21]:
```
False    9973
True       27
Name: date_issued:is_year_start, dtype: int64
```

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

PRACTICAL – 9

Aim: Perform Logistics Regression on Diabetic dataset and evaluate the model performance

[13]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
```

[14]:
```
# load dataset from url
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
column_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
                'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

df = pd.read_csv(url, names=column_names)
df.head()
```

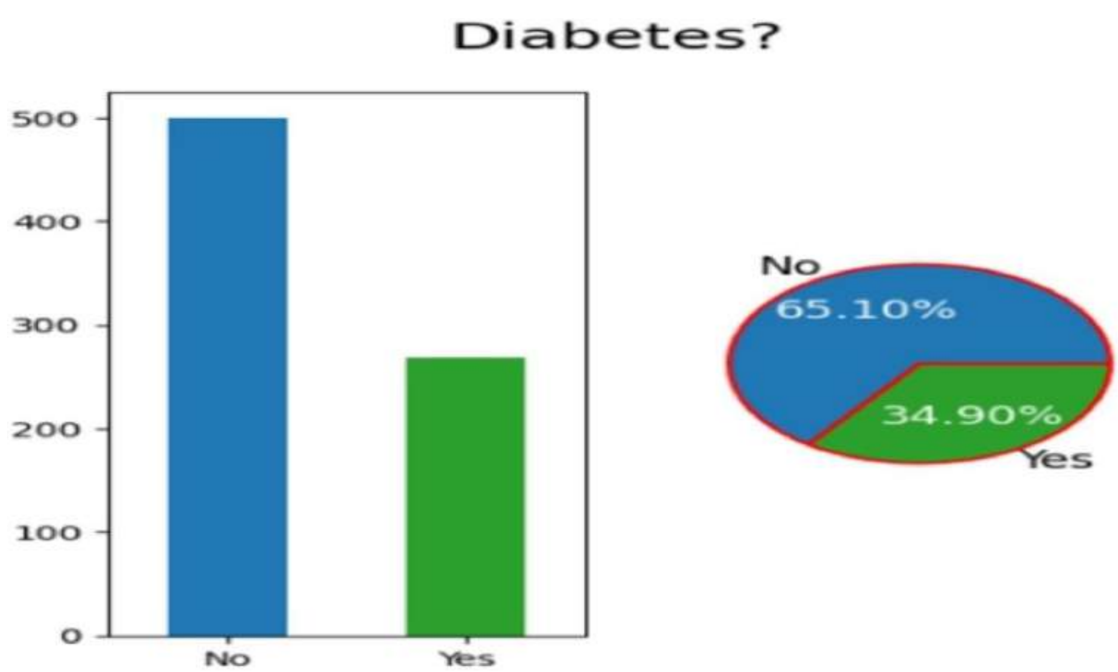
[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

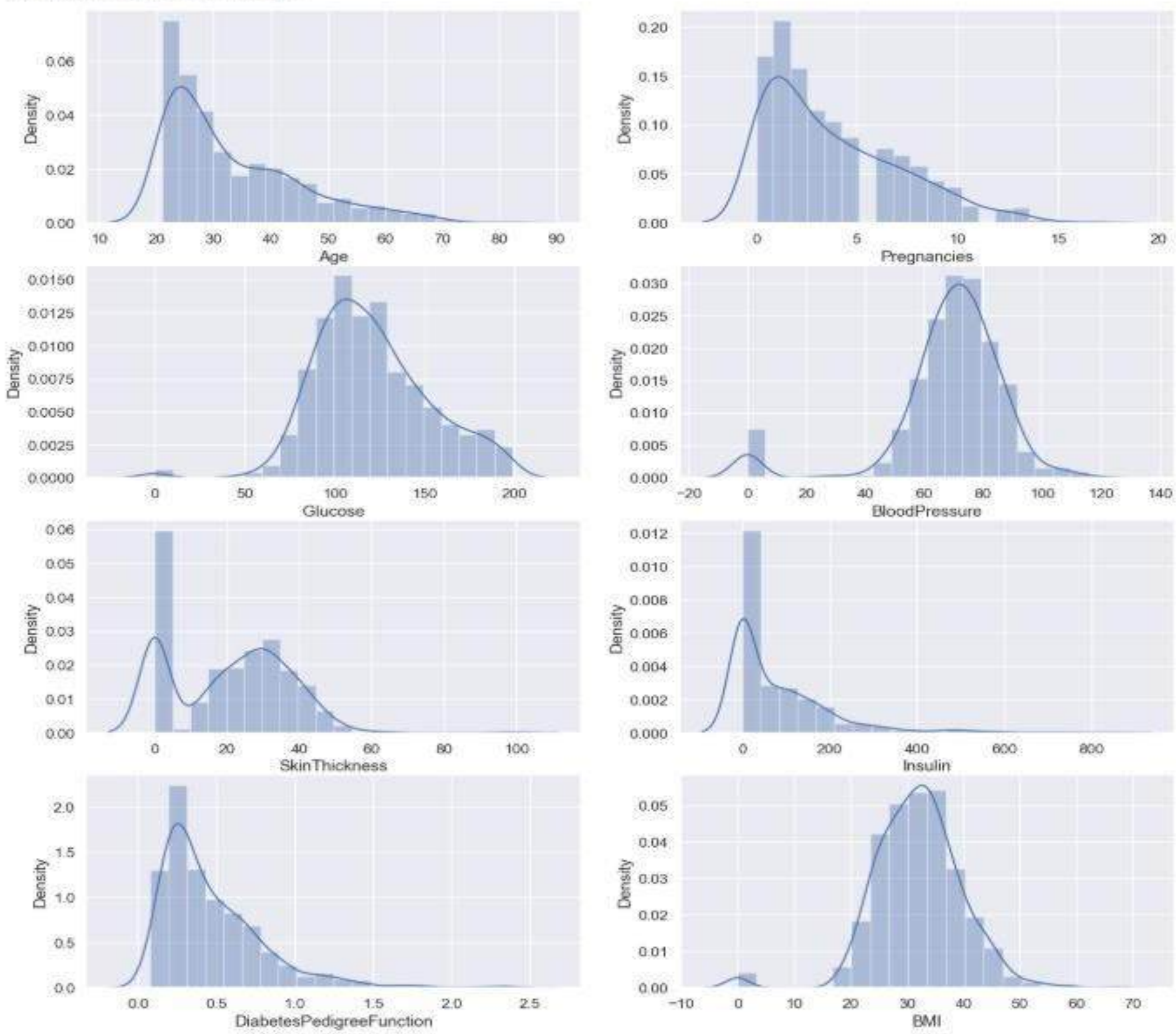
```python
import seaborn as sns
import matplotlib.pyplot as plt

f, ax = plt.subplots(1, 2, figsize=(5, 5))
f.suptitle('Diabetes?', fontsize=18)

# Bar plot
_ = df.Outcome.value_counts().plot.bar(
    ax=ax,
    rot=0,
    color=(sns.color_palette(), sns.color_palette()[2])
)
ax.set_xticklabels(["No", "Yes"])

# Pie chart
_ = df.Outcome.value_counts().plot.pie(
    labels=("No", "Yes"),
    autopct="%.2f%%",
    label="",
    fontsize=13,
    ax=ax[1],
    colors=(sns.color_palette(), sns.color_palette()[2]),
    wedgeprops={"linewidth": 1.5, "edgecolor": "r"}
)

# Set pie chart text color for percentage values
ax[1].texts[1].set_color("#F7F7F7")
ax[1].texts[3].set_color("#F7F7F7")

plt.tight_layout(rect=[0, 0.03, 1, 0.93])
plt.show()
```



ANALYTICAL STATEMENT:

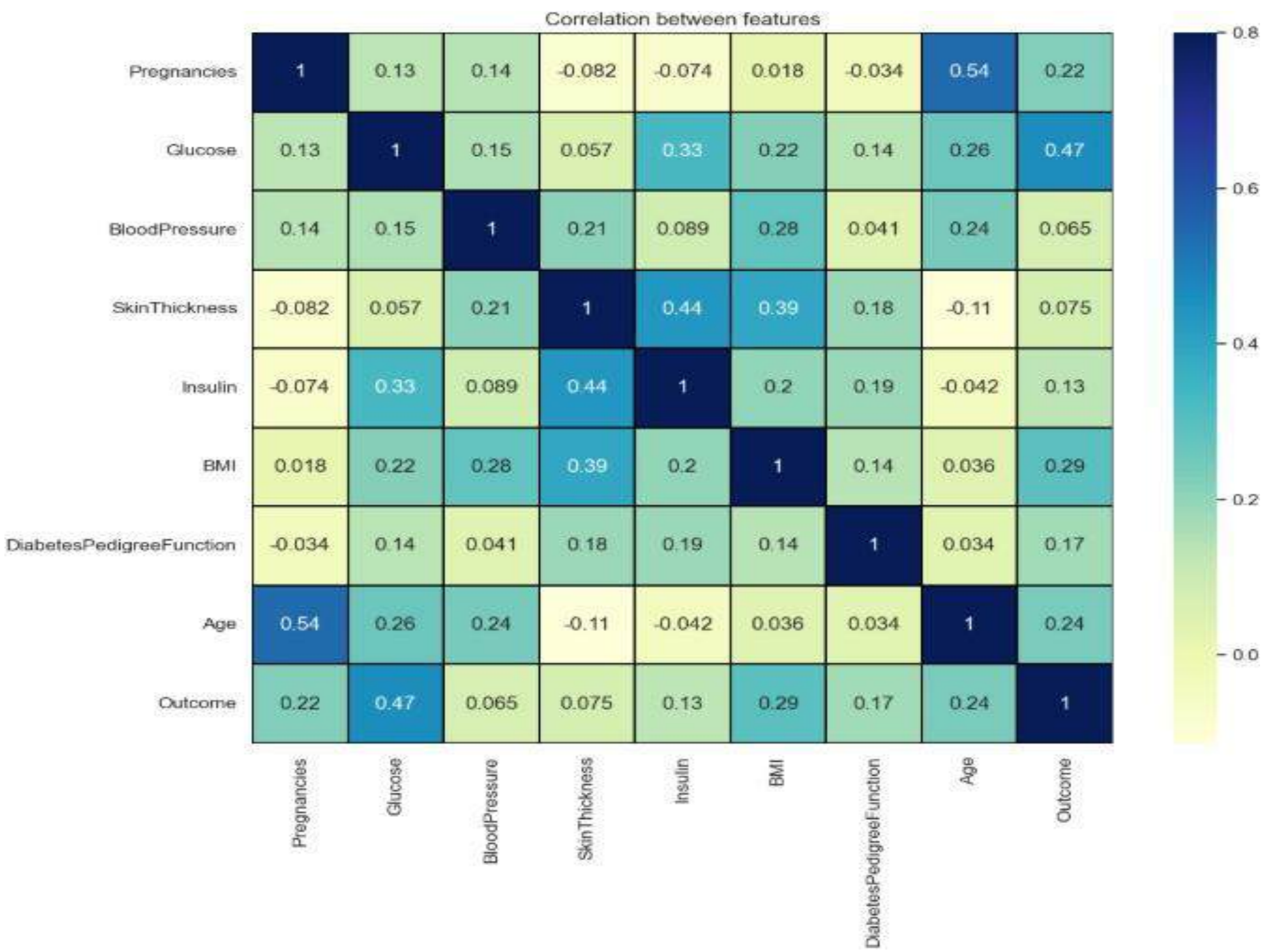
[24]:
```
fig, ax = plt.subplots(4, 2, figsize=(16, 16))

# Each line plots the distribution of a DataFrame column
sns.distplot(df.Age, bins=20, ax=ax[0, 0])
sns.distplot(df.Pregnancies, bins=20, ax=ax[0, 1])
sns.distplot(df.Glucose, bins=20, ax=ax[1, 0])
sns.distplot(df.BloodPressure, bins=20, ax=ax[1, 1])
sns.distplot(df.SkinThickness, bins=20, ax=ax[2, 0])
sns.distplot(df.Insulin, bins=20, ax=ax[2, 1])
sns.distplot(df.DiabetesPedigreeFunction, bins=20, ax=ax[3, 0])
sns.distplot(df.BMI, bins=20, ax=ax[3, 1])
```

[24]: <Axes: xlabel='BMI', ylabel='Density'>



ANALYTICAL STATEMENT:

```
[23]: corr = df.corr()  # Compute correlation matrix

sns.set(font_scale=1.15)
plt.figure(figsize=(14, 10))

sns.heatmap(
    corr,
    vmax=0.8,
    linewidths=0.01,
    square=True,
    annot=True,
    cmap='YlGnBu',
    linecolor="black"
)
plt.title('Correlation between features');
```



ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[25]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Prepare feature matrix X and label vector y
X = df.iloc[:, :-1]  # all rows, all columns except the last (features)
y = df.iloc[:, -1]   # all rows, last column (target)

# Split into train and test sets (75% train, 25% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0
)

# Create and train Logistic Regression model
LR = LogisticRegression()
LR.fit(X_train, y_train)
```

[25]:
```
▸ LogisticRegression
LogisticRegression()
```

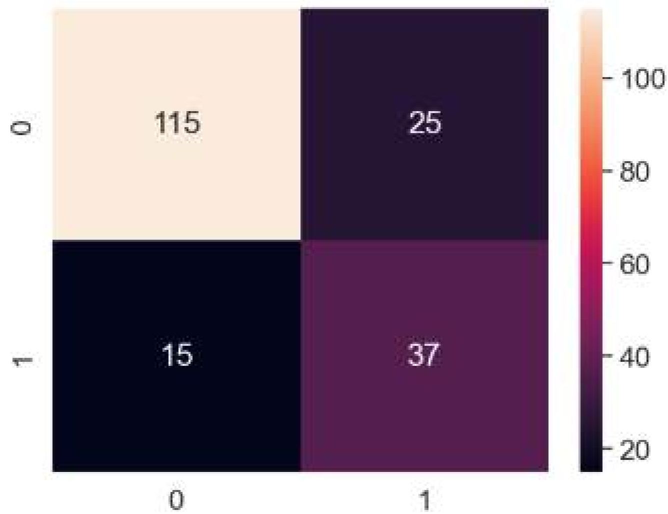
ANALYTICAL STATEMENT:

[26]:
```
# Prediction on test set
y_pred = LR.predict(X_test)

# Calculate and print accuracy (%)
print("Accuracy", LR.score(X_test, y_test) * 100)
```

```
Accuracy 79.16666666666666
```

ANALYTICAL STATEMENT:

[27]:
```
# Plot the confusion matrix
sns.set(font_scale=1.5)
cm = confusion_matrix(y_pred, y_test)
sns.heatmap(cm, annot=True, fmt="g")
plt.show()
```



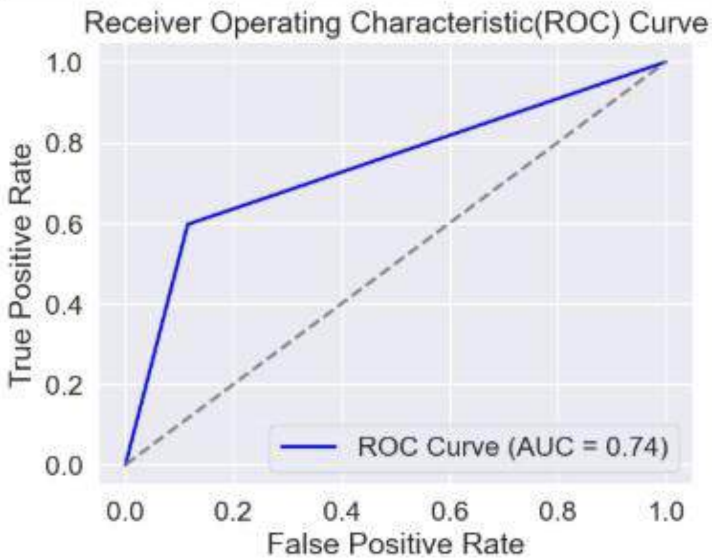
ANALYTICAL STATEMENT:

[28]:
```
print("Classification Report :\n", classification_report(y_test, y_pred))
```

```
Classification Report :
               precision    recall  f1-score   support

           0       0.82      0.88      0.85       130
           1       0.71      0.60      0.65        62

    accuracy                           0.79       192
   macro avg       0.77      0.74      0.75       192
weighted avg       0.79      0.79      0.79       192
```

ANALYTICAL STATEMENT:

```
[29]: fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic(ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

PRACTICAL – 10

Aim: Case Study: Amazon clothes sell clothes online. Customers come into the store, have meetings with a personal stylist, then they can go home and order either on a mobile app or website for the clothes they want. The company is trying to decide whether to focus their efforts on their mobile app experience or their website. Following is predictive analysis for this company

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('Ecommerce Customers.csv')
```

[3]:
```
df.head()
```

[3]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Email                 500 non-null    object
 1   Address               500 non-null    object
 2   Avatar                500 non-null    object
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

[5]:
```
df.describe()
```

[5]:

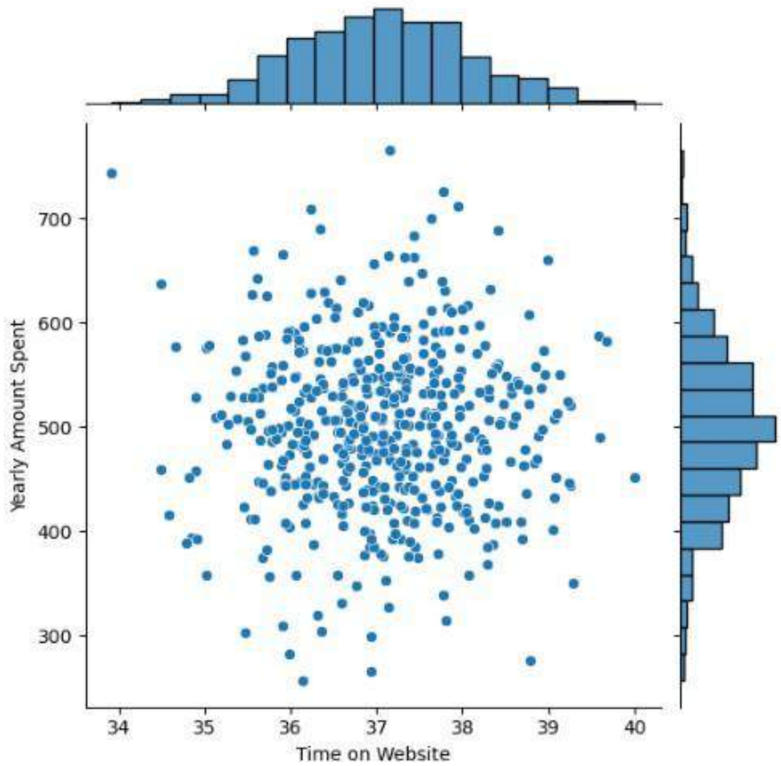
	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

```
[6]: sns.jointplot(x=df['Time on Website'], y=df['Yearly Amount Spent'])
```

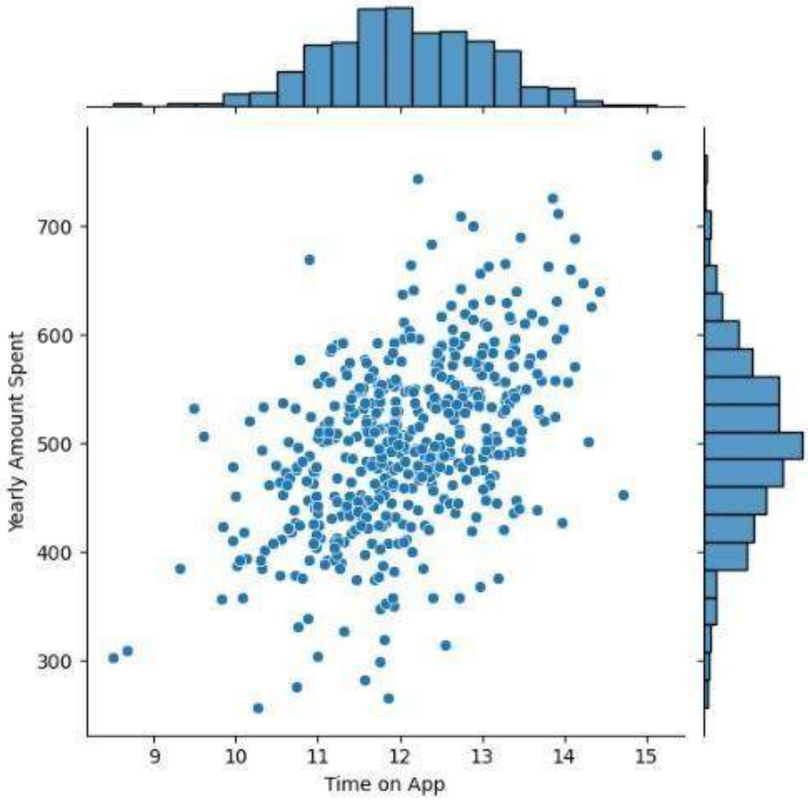
```
[6]: <seaborn.axisgrid.JointGrid at 0x2699b410ec0>
```



ANALYTICAL STATEMENT:

```
[7]: sns.jointplot(x=df['Time on App'], y=df['Yearly Amount Spent'])
```

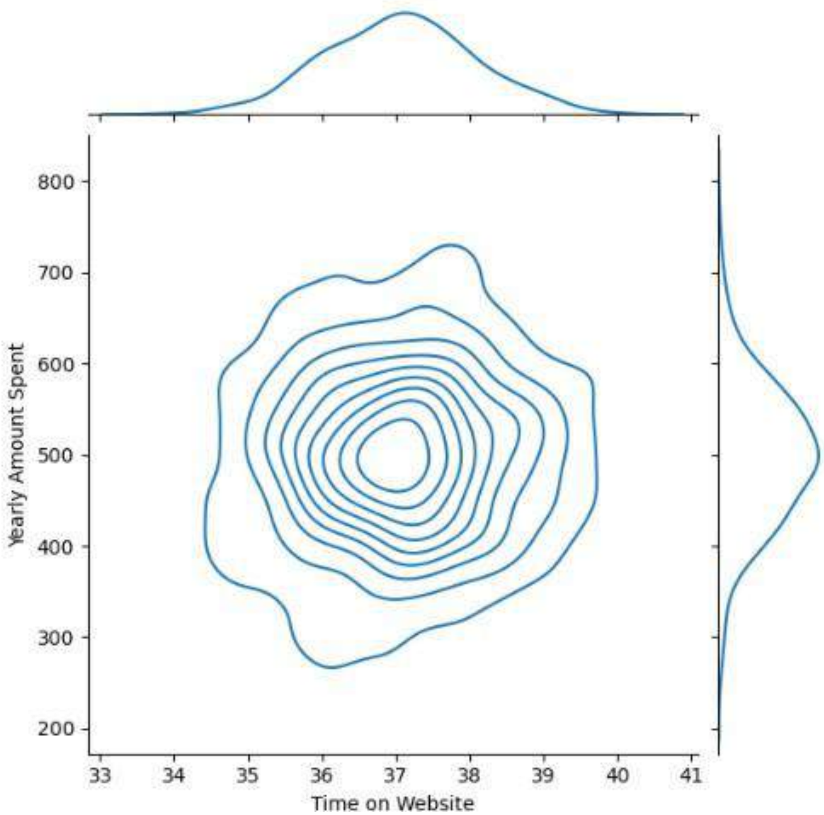
```
[7]: <seaborn.axisgrid.JointGrid at 0x2699b669f90>
```



ANALYTICAL STATEMENT:

[8]:
```
sns.jointplot(x=df['Time on Website'], y=df['Yearly Amount Spent'], kind='kde')
```

[8]: <seaborn.axisgrid.JointGrid at 0x2699c29a5d0>

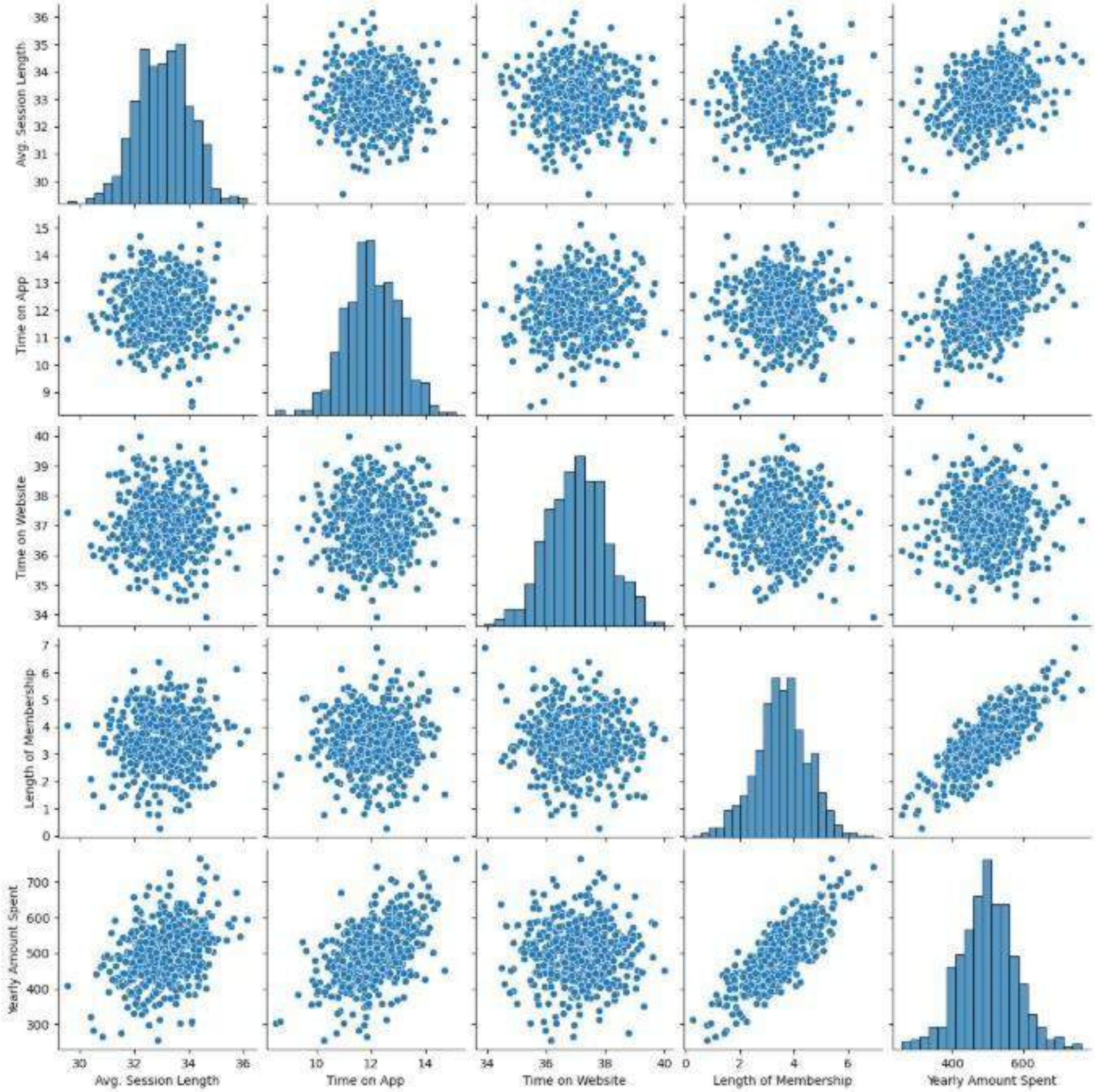


ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

```
[9]: sns.pairplot(df)
```

```
[9]: <seaborn.axisgrid.PairGrid at 0x2699b50a270>
```



ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[10]:
```
# Select target and features
y = df['Yearly Amount Spent']
X = df[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length of Membership']]

# Split into train and test sets (70% train, 30% test)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=466
)
```

[11]:
```
x_train
```

[11]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
110	31.853075	12.149375	37.325334	3.361815
100	32.498393	13.410759	35.990489	3.184619
189	32.200799	12.276982	38.232606	3.316465
328	33.369517	10.627949	38.040314	3.002957
122	33.268330	11.113330	37.387946	4.018727
...	...	...	...	...
95	32.461212	13.291143	38.633626	3.871003
495	33.237660	13.566160	36.417985	3.746573
164	33.154255	11.795887	37.658617	4.520353
369	34.357196	9.477778	37.906015	5.047023
330	30.574364	11.351049	37.088847	4.078308

350 rows × 4 columns

[12]:
```
x_test
```

[12]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
277	32.192499	13.325412	36.897295	5.049927
7	32.739143	12.351959	37.373359	4.434273
392	33.258238	11.514949	37.128039	4.662845
9	31.936549	11.814128	37.145168	3.202806
476	34.336677	11.246813	38.682584	2.094762
...	...	...	...	...
339	32.997459	12.589241	37.332241	2.804014
176	32.332637	11.548761	38.576516	4.773503
384	33.593964	11.520567	36.189132	3.561215
373	31.366212	11.163160	37.088319	3.620355
455	33.421212	10.706642	35.766154	3.393975

150 rows × 4 columns

ANALYTICAL STATEMENT:

```
[13]: y_train
```

```
[13]: 110    459.285123
      100    518.064558
      189    478.885391
      328    422.368737
      122    514.239521
                ...
      95     543.340166
      495    573.847438
      164    558.047581
      369    531.961551
      330    442.064414
      Name: Yearly Amount Spent, Length: 350, dtype: float64
```

```
[14]: y_test
```

```
[14]: 277    616.660286
      7      549.904146
      392    549.131573
      9      427.199385
      476    408.958336
                ...
      339    476.139247
      176    532.717486
      384    474.532329
      373    430.588883
      455    438.303788
      Name: Yearly Amount Spent, Length: 150, dtype: float64
```

ANALYTICAL STATEMENT:

```
[15]: from sklearn.linear_model import LinearRegression

      lm = LinearRegression()
      lm.fit(x_train, y_train)
```

```
[15]: ▾ LinearRegression
      LinearRegression()
```

ANALYTICAL STATEMENT:

```
[16]: # Print the coefficients for each feature
      print("coefficients : \n", lm.coef_)

      coefficients :
       [25.80332726 38.51709286  0.25895459 61.31108629]
```

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

[17]:
```
# Print the model intercept
print(lm.intercept_)
```

```
-1044.5036020011362
```

ANALYTICAL STATEMENT:

[18]:
```
# Make predictions on the test set
pred = lm.predict(x_test)
```

[19]:
```
pred
```

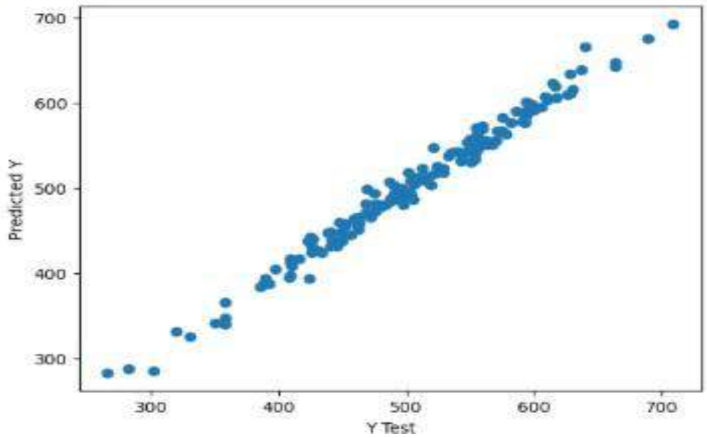
[19]:
```
array([618.59738219, 557.58489234, 552.69054252, 440.59792148,
       413.14058911, 340.28200231, 417.05076117, 567.14512987,
       486.39861505, 512.55353338, 605.29708886, 562.95628446,
       607.08443312, 460.13846164, 576.35649482, 444.91206259,
       424.23448484, 541.48186139, 553.75608221, 568.86829905,
       499.10455935, 456.93287659, 542.81573022, 417.49577921,
       366.04769894, 603.43045993, 590.70039625, 692.29665547,
       431.78681007, 500.34768221, 547.00973557, 517.81598481,
       566.28447227, 438.13950331, 491.37880007, 394.21075412,
       555.31484354, 331.52342984, 424.69870138, 285.04378446,
       325.61851113, 588.15883309, 638.46491303, 387.39050509,
       609.13717179, 450.60814251, 493.18836125, 386.62876645,
       587.74165284, 513.5439275 , 601.10754812, 341.94674184,
       404.59612411, 580.26830959, 547.50152258, 426.83924916,
       384.02457468, 445.40171027, 562.73546791, 523.01747242,
       480.87160424, 471.15971056, 518.07824072, 409.0043681 ,
       446.88904424, 484.40816565, 592.76916447, 465.9794765 ,
       455.72937994, 511.21739216, 572.54515976, 610.11018863,
       395.53832715, 487.05595184, 556.07747141, 488.59518707,
       394.50055902, 472.93228755, 480.77595416, 495.57527412,
       397.50941223, 531.5911328 , 348.06423309, 534.67898031,
       492.73232369, 550.64560032, 455.99795499, 481.04415785,
       598.69087795, 437.59311244, 552.22190717, 465.47724338,
       589.50022185, 622.24323315, 503.16112831, 589.0955597 ,
       478.07535762, 435.34662831, 442.52444787, 530.48062232,
       459.6123008 , 542.60349929, 486.07411577, 455.85960052,
       487.2535891 , 499.50230338, 516.43794831, 431.38320758,
       542.38014362, 432.01744695, 287.83181211, 571.01047393,
       457.12136611, 582.61188067, 508.95211349, 642.28039777,
       598.22751992, 647.68078398, 448.80808719, 533.45179723,
       283.19734675, 551.16954995, 525.20597843, 550.42009754,
       481.20571627, 552.55237583, 439.44568918, 504.61300804,
       490.54504007, 458.21853804, 675.01852482, 542.63614686,
       480.56915115, 577.2105439 , 594.81658673, 577.05241883,
       464.06446285, 633.59356482, 523.42199249, 615.41761635,
       508.06676416, 513.4365391 , 507.3196324 , 503.34480176,
       665.96149325, 473.42605979, 537.26895335, 493.78449287,
       426.39355141, 447.61368551])
```

ANALYTICAL STATEMENT:

Name: Kunal Manojkumar Mistri
Class: Msc.DA
Roll No: 3432

```
[20]: plt.scatter(y_test, pred)
      plt.xlabel('Y Test')
      plt.ylabel('Predicted Y')
```

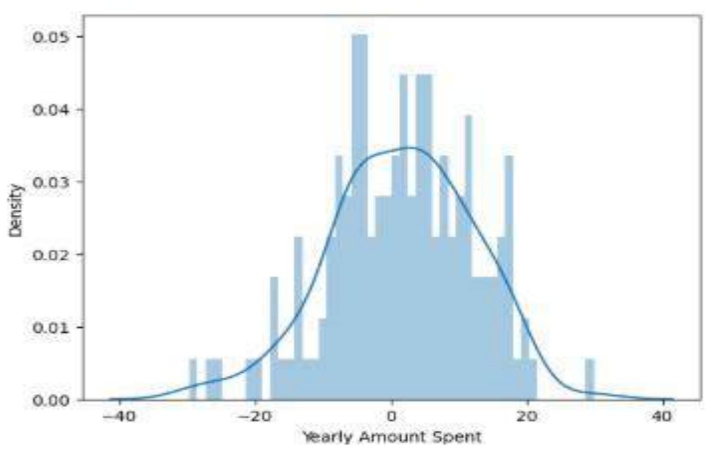
```
[20]: Text(0, 0.5, 'Predicted Y')
```



ANALYTICAL STATEMENT:

```
[21]: sns.distplot((y_test - pred), bins=50)
```

```
[21]: <Axes: xlabel='Yearly Amount Spent', ylabel='Density'>
```



ANALYTICAL STATEMENT:

```
[22]: from sklearn import metrics
      print('MAE:', metrics.mean_absolute_error(y_test, pred))              # Mean Absolute Error
      print('MSE:', metrics.mean_squared_error(y_test, pred))               # Mean Squared Error
      print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))     # Root Mean Squared Error

MAE: 8.515783908090931
MSE: 111.46467736774434
RMSE: 10.557683333371216
```

ANALYTICAL STATEMENT:

```
[25]: coefficients = pd.DataFrame(lm.coef_, X.columns)
      coefficients.columns = ['Coefficient']
      coefficients
```

[25]:

	Coefficient
Avg. Session Length	25.803327
Time on App	38.517093
Time on Website	0.258955
Length of Membership	61.311086

ANALYTICAL STATEMENT: