CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this Minor thesis titled **"MALL CUSTOMER REPORT "** in fulfillment of the requirement for the degree of Bachelor of Computer Application (specialization in IT Sector) and submitted to

"SATYUG DARSHAN INSTITUTE OF ENGINEERING AND TECHNOLOGY", is an authentic record of my own work carried out under the supervision of **Mr. Ankit mishra.**

The work contained in this thesis has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

Mohammad Jishan / Mohammad Arasalan
BCA(DS) 23/ 19 BCA(DS) 23/ 28

ACKNOWLEDGEMENT

It is with deep sense of gratitude and reverence that I express my sincere thanks to my supervisor Mr. **Ankit mishra** for her guidance, encouragement, help and useful suggestions throughout. Her untiring and painstaking efforts, methodical approach and individual help made it possible for me to complete this work in time. I consider myself very fortunate to have been associated with a scholar like her. Her affection, guidance and scientific approach served a veritable incentive for completion of this work. I shall ever remain indebted to the faculty members of Satyug Darshan Institute of Engineering and Technology, Faridabad, CAREERERA team and all my classmates for their cooperation, kindness and general help extended to me during the completion of this work. Although it is not possible to name individually, I cannot forget my well-wishers at Satyug Darshan Institute of Engineering and Technology, Faridabad and outsiders for their persistent support and cooperation. This acknowledgement will remain incomplete if I fail to express my deep sense of obligation to my parents and God for their consistent blessings and encouragement.

Mohammad Jishan / Mohammad Arasalan

ABOUT TRAINING

Studying Python and machine learning in college is like unlocking a world of endless possibilities. Python, known for its clear syntax and flexibility, is a perfect starting point. It's not just about learning a language but understanding how to wield it for data analysis, automation, and beyond.

Machine learning takes this a step further, teaching us to teach computers. From basic algorithms to complex neural networks, the journey is about turning data into insights. Hands-on projects play a key role, pushing us to apply theory to real-world challenges and collaborate with peers.

College offers more than just classes; it's a community where ideas flourish. Whether through research opportunities or industry partnerships, we gain the tools to innovate and solve problems creatively. In the end, learning Python and machine learning isn't just about skills—it's about shaping the future with every line of code.

CHAPTER -1

INTRODUCTION TO DATA SCIENCE

1.1 Introduction to topic

1. Exploring Data: Introduction to Data Science introduces students to the world of data exploration. It's about learning how to gather, clean, and organize data to extract meaningful insights. This process is crucial for making informed decisions in various fields, from business to healthcare.

2. Statistics and Analysis: Students delve into statistical methods and analysis techniques. This includes understanding probability, hypothesis testing, and regression analysis. These tools are fundamental for interpreting data patterns and drawing conclusions from datasets.

3. Programming Skills: A key aspect of Introduction to Data Science is learning programming languages like Python or R. These languages are used for data manipulation, visualization, and implementing machine learning algorithms. Hands-on coding exercises build proficiency in handling real-world data.

4. Data Visualization: Effective communication of insights is another focus. Students learn to create visual representations of data using tools like matplotlib . Visualizations help to convey complex information clearly to stakeholders and decision-makers.

5. Ethical Considerations: Lastly, Introduction to Data Science covers ethical considerations. This includes issues such as data privacy, bias in algorithms,

and responsible data handling practices. Understanding these ethical dimensions is crucial for using data science responsibly and ethically.

Introduction to Data Science lays the foundation for students to become proficient in handling data, applying statistical techniques, and using programming tools to derive insights. It prepares them for roles in data-driven decision-making and research across diverse industries.

1.2 Motivation

1. **Unveiling Insights**: Introduction to Data Science sparks excitement by revealing how data holds the keys to uncovering patterns and insights that drive decisions. It empowers learners to discover meaningful connections in vast datasets that impact real-world outcomes.

2. **Practical Application:** Students are motivated by the practical application of data science techniques. From predicting customer behavior to optimizing processes, they see how data-driven decisions can transform industries and improve everyday experiences.

3. **Versatility:** This course motivates learners with its versatility. Whether they're interested in healthcare, finance, marketing, or social sciences, data science skills are universally applicable. It opens doors to diverse career paths where understanding data is crucial.

4. **Innovation:** Introduction to Data Science inspires innovation. Students learn to harness the power of data to innovate and solve complex problems creatively. They explore cutting-edge technologies like machine learning and AI that shape the future.

5. **Impact**: Ultimately, this course motivates by emphasizing the impact of data science on society. From improving healthcare outcomes to driving sustainable practices, students understand how their skills can contribute positively to the world around them.

Introduction to Data Science motivates students by showing them the transformative potential of data analysis and the opportunities it brings to innovate and make a difference in the world**.**

1.3 Objective of training

1. **Data Proficiency:** The primary objective of training in Data Science is to develop proficiency in handling and manipulating large and complex datasets. This includes skills in data cleaning, integration, and transformation to ensure data quality for analysis.

2. **Statistical Analysis:** Training in Data Science aims to equip individuals with strong statistical knowledge and techniques. This includes understanding probability theory, hypothesis testing, and regression analysis to derive meaningful insights and make data-driven decisions.

3. **Machine Learning Skills:** An important objective is to teach machine learning algorithms and techniques. This includes supervised and unsupervised learning, classification, regression, clustering, and deep learning. These skills enable the development of predictive models and pattern recognition systems.

4. **Programming Competence:** Data Science training focuses on building proficiency in programming languages such as Python, R, and SQL. These languages are used for data manipulation, statistical analysis, visualization, and implementing machine learning algorithms.

5. **Business Acumen:** Lastly, training in Data Science aims to foster business acumen among practitioners. This includes understanding how to translate data insights into actionable business strategies, communicate findings to stakeholders, and drive organizational decision-making.

Training in Data Science prepares individuals for careers as data scientists, analysts, or machine learning engineers by providing them with the necessary technical skills, statistical knowledge, and business acumen to extract valuable insights from data and drive innovation in various industries.

CHAPTER -2

PYTHON FOR DATA SCIENCE

1.1. Introduction to Python

“Python is an interpreted, object-oriented, high-level programming language with dynamic semantics”. This language consists mainly of data structures which make it very easy for the data scientists to analyze the data very effectively. It does not only help in forecasting and analysis it also helps in connecting the two different languages. Two best features of this programming language is that it does not have any compilation step as compared to the other programming language in which compilation is done before the program is being executed and other one is the reuse of the code, it consist of modules and packages due to which we can use the previously written code anywhere in between the program whenever is required. There are multiple languages for example **R, Java, SQL,** available in the market which can be used to analyze and evaluate the data, but due to some outstanding features python is the most famous language used in the field of data science. Python is mostly used and easy among all other programming languages.

1. **Versatile and Beginner-Friendly**: Python is renowned for its versatility and beginner-friendly syntax. It serves as an excellent entry point into programming for those new to coding, while also offering powerful features for advanced users.

2. **Rich Ecosystem**: Python boasts a vast ecosystem of libraries and frameworks. From data analysis (e.g., pandas, NumPy) to web development (e.g., Django, Flask) and machine learning (e.g., TensorFlow, PyTorch), Python's rich library support facilitates diverse applications.

3. **Clear and Readable Code**: One of Python's strengths is its readability. Its elegant syntax, with minimalistic and straightforward code structures,

enhances code clarity and reduces the time spent on debugging, making it easier to maintain and collaborate on projects.

4. **Community and Support**: Python benefits from a vibrant community of developers worldwide. This community contributes to its continuous evolution and provides extensive documentation, tutorials, and forums for troubleshooting and sharing knowledge.

5. **Scalability and Industry Adoption**: Python's scalability makes it suitable for projects of varying scales, from small scripts to large-scale applications. Its popularity in industries such as finance, scientific research, and web development underscores its relevance and widespread adoption.

Python's combination of simplicity, power, and community support makes it an indispensable tool for beginners and professionals alike, enabling diverse applications across numerous fields and industries.

1.2 Operators, Conditional Statements …..

OPERATORS - Operators are the symbols in python that are used to perform Arithmetic or logical operations. Following are the different types of operators in python.

Arithmetic operators - Arithmetic operators carry out mathematical operations and they are mostly used with the numeric values.

Arithmetic operators		
Operator	Name	Example
+	Addition	A+B
-	Subtraction	A-B
*	Multiplication	A*B
/	Division	A/B
%	Modulus	A%B
**	Exponentiation	A**B
//	Quotient	A//B

Fig. 1.2.1: Arithmetic

operators A and B are the numeric value

Assignment operators - As the name decides this operators are used for assigning the values to the variables.

ASSIGNMENT OPERATORS		
Operator	Example	may also be written
=	a = 6	a = 6
+=	a += 3	a = a + 3
-=	a -= 4	a = a - 4
*=	a *= 5	a = a * 5
/=	a /= 6	a = a / 6
%=	a %= 7	a = a % 7
//=	a //= 8	a = a // 8
**=	a **= 9	a = a ** 9
&=	a &= 1	a = a & 1

Fig. 1.2.2: Assignment Operators

Here a is any value and number of operations are performed on this value.

Logical operators - These operators are used to join conditional statements

Logical Operators		
Operator	Description	Example
and	if both statements are true it returns true	x <5 **and** x <10
or	if any of the two statement is true it returns true	x <4 **or** x <8
not	if the result is true it reverses the result and gives false	**not** (x <4 **and** x <8)

Fig. 1.2.3: Logical Operators

Here a is any value provided by us and on which multiple operations can be performed.

Comparison operators - These operators are used to compare two different values.

Comparison operators		
Operator	Name	Example
==	Equal	a == b
!=	Not equal	a!=b
>	Greater than	a >b
<	less than	a <b
>=	Greater than equal to	a>= b
<=	less than equal to	a <=b

Fig. 1.2.4: Comparison operators

Here a and b are two different values and these values are compared.

Membership operators - These operators are used to check membership of a particular value. It is used to check whether a specific value is present in the object or not.

Membership operators		
Operator	Description	Example
in	it returns a True if the value is present inside the object	a **in** b
not in	it returns a True if the value is not present inside the object	a **not in** b

Fig. 1.2.5: Membership operators

Condition statements

import the Necessary Libraries :

Python is known for its extensive ecosystem of libraries that extend its capabilities for various tasks such as data analysis, machine learning, web development, and more. To leverage these libraries in your Python scripts or projects, you need to import them using the `import` statement. Here's how you can import necessary libraries:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from mpl_toolkits.mplot3d import Axes3D
```

In [2]:
```
data = pd.read_csv('../input/customer-segmentation-tutorial-in-python/Mall_Customers.csv')
data.head()
```

Out[2]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

1.3 Understanding Standard Libraries Pandas, Numpy…..

Libraries in Python

The Python library is vast. There are built-in functions in the library which are written in C language. This library provides access to system functionality such as file input output and that is not accessible to Python programmers. These modules and libraries provide solutions to the many problems in programming.

Following are some Python libraries.

- **Matplotlib**
- **Pandas**
- **Numpy**

Matplotlib

Matplotlib: A Powerful Plotting Library in Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots, histograms, power spectra, bar charts, error charts, scatterplots, and more, with publication-quality output.

Key Features:

1. **Versatility**: Matplotlib offers a wide variety of plots and customization options, making it suitable for both basic and advanced visualizations. It supports different plot types, including line plots, scatter plots, histograms, bar plots, and more.

2. **Integration**: Matplotlib seamlessly integrates with NumPy, Pandas, and other Python libraries, allowing easy plotting of data stored in arrays or data frames. It is a core component of the scientific Python ecosystem.

3. **Customization**: Users have fine-grained control over every aspect of a plot, including colors, labels, axes, grids, annotations, and more. Matplotlib's extensive customization capabilities enable users to create highly tailored visualizations.

4. **Multiple Interfaces**: Matplotlib provides two interfaces for plotting: a MATLAB-like state-based interface (`pyplot`) for quick and easy plot generation, and an object-oriented interface for more control and customization of plots.

5. **Community and Documentation**: With an active community and comprehensive documentation, Matplotlib is well-supported and continuously evolving. It offers numerous examples and tutorials to help users get started and advance their plotting skills

```python
import matplotlib.pyplot as plt
import numpy as np

# Data for mall customers
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
customers = [120, 140, 180, 160, 200]

# Plotting the data
plt.figure(figsize=(8, 5))
plt.bar(days, customers, color='skyblue')

# Adding labels and title
plt.xlabel('Days of the Week')
plt.ylabel('Number of Customers')
plt.title('Number of Mall Customers per Day')

# Adding values on top of bars
for i, customer in enumerate(customers):
    plt.text(i, customer + 5, str(customer), ha='center', va='bottom', fontsize=10)

# Display the plot
plt.ylim(0, 250)  # Adjusting y-axis limit for better visualization
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

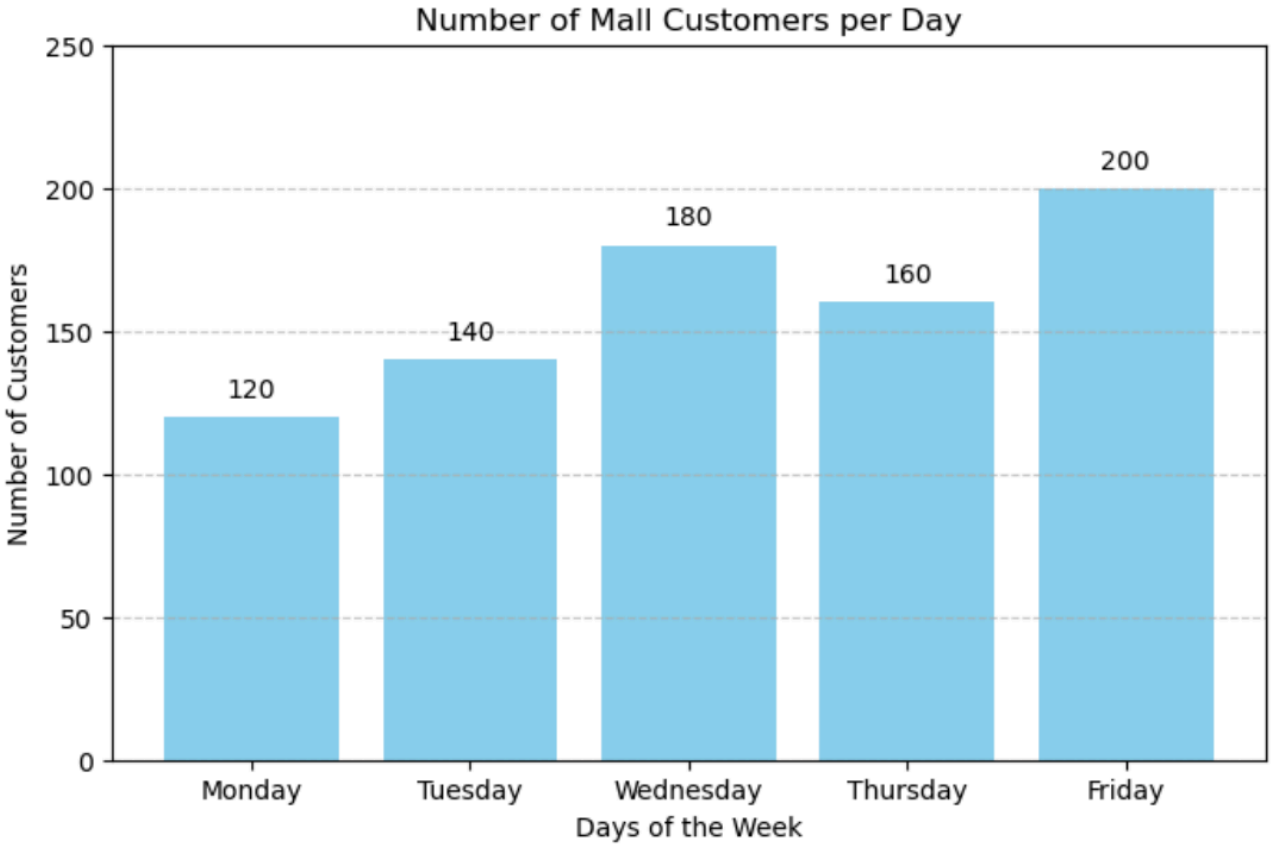


Figure 1.3.1: Matplotlib basic example

Conclusion:

Matplotlib is a powerful and flexible library that enables users to create professional-quality visualizations for data analysis, scientific research, presentations, and more. Its versatility, customization options, and extensive documentation make it an indispensable tool in the Python ecosystem.

Pandas

Pandas is a powerful Python library built on top of NumPy that provides easy-to-use data structures and data analysis tools. It is widely used for data manipulation, cleaning, analysis, and preparation tasks in Python.

Key Features:

1. **DataFrame**: Pandas introduces the `DataFrame` data structure, which is a two-dimensional tabular data structure with labeled axes (rows and columns). It allows for easy handling and manipulation of structured data.

2. **Series**: Alongside `DataFrame`, Pandas also provides the `Series` data structure, which represents a one-dimensional labeled array capable of holding any data type (integers, strings, floats, Python objects, etc.).

3. **Data Manipulation**: Pandas offers a rich set of functions and methods for data manipulation tasks such as merging and joining datasets, reshaping data, slicing and indexing, handling missing data (`NaN` values), and more.

4. **Data Cleaning**: It provides tools for cleaning messy data, including handling duplicates, outliers, and inconsistent data formats. Pandas also supports data transformation tasks like applying functions across data elements.

5. **Data Analysis**: Pandas facilitates exploratory data analysis (EDA) with built-in functions for descriptive statistics, grouping and aggregation, time series analysis, and plotting integration with Matplotlib.

```
2]: import pandas as pd

# Creating a DataFrame for mall customers
data = {'Day': ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
        'Customers': [120, 140, 180, 160, 200]}
df = pd.DataFrame(data)

# Displaying the DataFrame
print("DataFrame for Mall Customers:")
print(df)
print()

# Performing data manipulation
total_customers = df['Customers'].sum()
average_customers = df['Customers'].mean()

# Displaying summary statistics
print(f"Total number of customers: {total_customers}")
print(f"Average number of customers per day: {average_customers}")
```

OUTPUT:

```
DataFrame for Mall Customers:
         Day  Customers
0     Monday        120
1    Tuesday        140
2  Wednesday        180
3   Thursday        160
4     Friday        200

Total number of customers: 800
Average number of customers per day: 160.0
```

Figure 1.3.2: series and data frame in pandas

NumPy

”NumPy is a library for the Python programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays”. The previous similar programming of NumPy is Numeric, and this language was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. [12] It is an open source library and free of cost.

NumPy: Numerical Computing with Python

NumPy (Numerical Python) is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

Key Features:

1. **Array Object**: NumPy introduces the `ndarray` (N-dimensional array) data structure, which allows for efficient storage and manipulation of homogeneous data. Arrays can be one-dimensional, two-dimensional, or multi-dimensional.

2. **Mathematical Functions**: NumPy provides a wide range of mathematical functions that operate on arrays, making it easier to perform mathematical operations on entire datasets without writing explicit loops.

3. **Broadcasting**: NumPy's broadcasting capability allows operations between arrays of different shapes and sizes. This eliminates the need for explicit looping over arrays, making code more concise and efficient.

4. **Integration with Other Libraries**: NumPy seamlessly integrates with other Python libraries, such as Pandas, Matplotlib, and scikit-learn, enhancing their capabilities for data manipulation, visualization, and machine learning tasks.

5. **Performance**: NumPy operations are implemented in C and Fortran, ensuring high performance even with large datasets. This makes NumPy suitable for numerical computations, simulations, and data-intensive applications.

```
]: import numpy as np
   import time

   # Parameters
   num_customers = 10
   mall_opening_time = 10
   mall_closing_time = 21

   arrival_times = np.random.uniform(0, mall_closing_time - mall_opening_time, num_customers)

   shopping_durations = np.random.normal(2, 1, num_customers)


   print("Simulating customers in the mall...\n")
   for i in range(num_customers):
       arrival_time = mall_opening_time + arrival_times[i]
       duration = shopping_durations[i]
       print(f"Customer {i+1} arrives at {arrival_time:.2f} and spends {duration:.2f} hours shopping.")
       time.sleep(1)

   print("\nSimulation complete.")
```

OUTPUT

```
Simulating customers in the mall...

Customer 1 arrives at 19.75 and spends 2.49 hours shopping.
Customer 2 arrives at 16.57 and spends 1.11 hours shopping.
Customer 3 arrives at 16.72 and spends 0.21 hours shopping.
Customer 4 arrives at 16.23 and spends 2.91 hours shopping.
Customer 5 arrives at 11.60 and spends 2.23 hours shopping.
Customer 6 arrives at 20.95 and spends 2.46 hours shopping.
Customer 7 arrives at 14.00 and spends 0.67 hours shopping.
Customer 8 arrives at 19.36 and spends 3.25 hours shopping.
Customer 9 arrives at 13.89 and spends 1.11 hours shopping.
Customer 10 arrives at 12.17 and spends 1.27 hours shopping.

Simulation complete.
```

Figure 1.3.3: NumPy basic example

CHAPTER 4

APPROACH USED (REQUIRED TOOLS)

• **Decision Tree:** A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered.

• **KNN Algorithm:** K-NN algorithm stores all the available data and classifies a new data point based on the similarity.

• **Logistic Regression:** Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.

REQUIRED TOOLS:

For application development, the following Software Requirements are: Operating System: Windows 11

Language: python

Tools: JUPYTER notebook

CHAPTER 5:-RESULTS

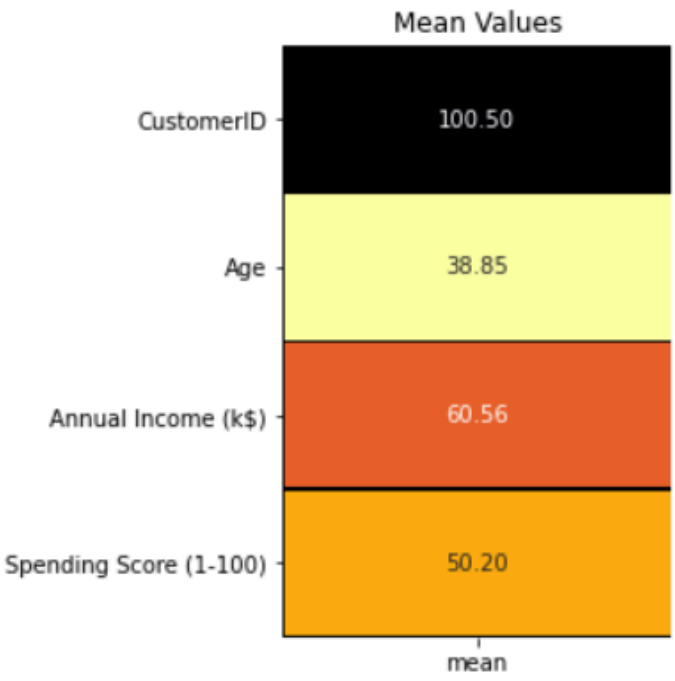
[3]:

```
fig,ax = plt.subplots(nrows = 1,ncols = 1,figsize = (5,5))

plt.subplot(1,1,1)
sns.heatmap(data.describe().T[['mean']],cmap = 'inferno_r',annot = True,fmt = '.2f',linecolor = 'black',linewidths = 0.4,cbar = False);
plt.title('Mean Values');

fig.tight_layout(pad = 3)
```

OUTPUT



Exploratory Data Analysis

Dividing features into Numerical and Categorical :

```
[ ]:
col = list(data.columns)
categorical_features = []
numerical_features = []
for i in col:
    if len(data[i].unique()) > 6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

print('Categorical Features :',*categorical_features)
print('Numerical Features :',*numerical_features)
```

```
Categorical Features : Gender
Numerical Features : CustomerID Age Annual Income (k$) Spending Score (1-100)
```

- Here, categorical features are defined if the attribute has less than 6 unique elements else it is a numerical feature.
- Typical approach for this division of features can also be based on the datatypes of the elements of the respective attribute.

Eg : datatype = integer, attribute = numerical feature ; datatype = string, attribute = categorical feature

- For this dataset, as the number of features are less, we can check the dataset manually as well.

```
[ ]: from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     df1 = data.copy(deep = True)

     df1['Gender'] = le.fit_transform(df1['Gender'])

     print('Label Encoder Transformation')
     print(df1['Gender'].unique(),' = ',le.inverse_transform(df1['Gender'].unique()))
```

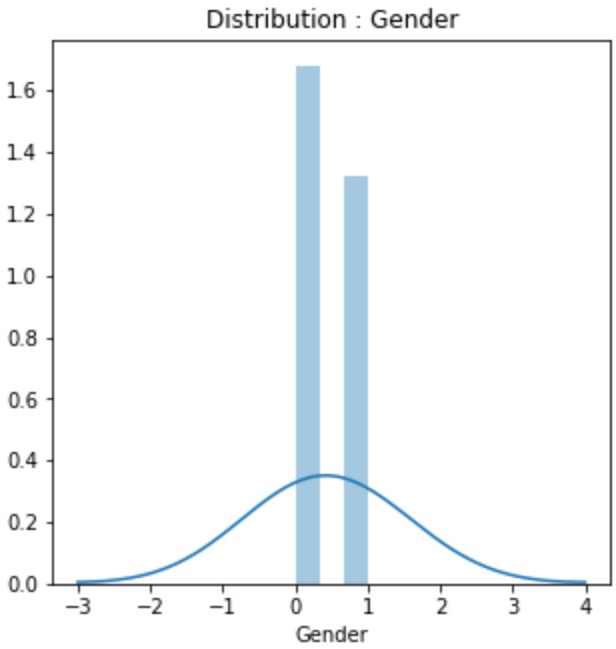
```
Label Encoder Transformation
[1 0]  =  ['Male' 'Female']
```

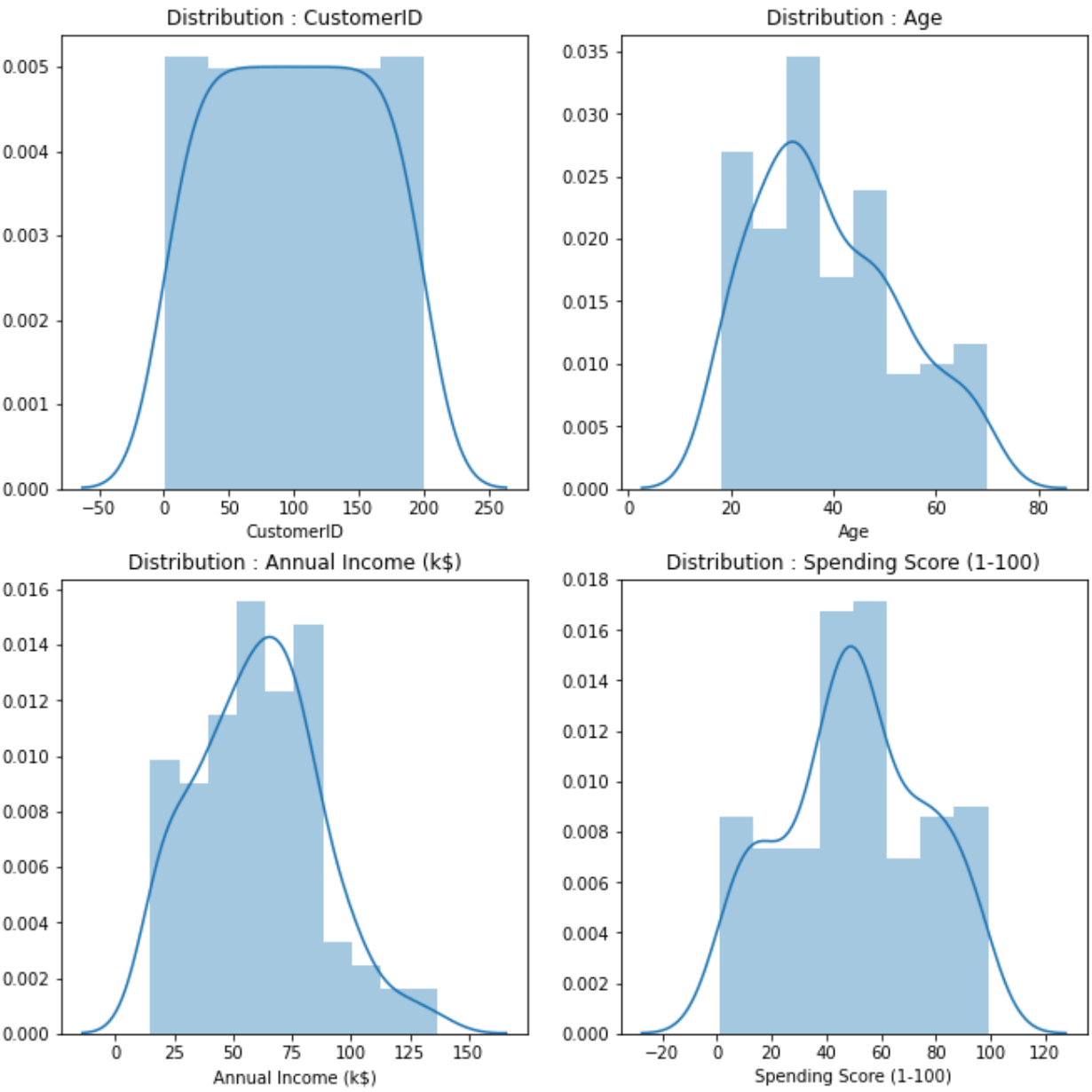
- Creating a deep copy of the original dataset and label encoding the text data of **Gender**.
- Modifications in the original dataset will not be highlighted in this deep copy.
- Hence, we use this deep copy of the dataset that has **Gender** converted into numerical values for visualization & modeling purposes.

Distribution of Categorical and Numerical Features :

```
[ ]: fig, ax = plt.subplots(nrows = 1,ncols = 1,figsize = (5,5))
     for i in range(len(categorical_features)):
         plt.subplot(1,1,i+1)
         sns.distplot(df1[categorical_features[i]],kde_kws = {'bw' : 1});
         title = 'Distribution : ' + categorical_features[i]
         plt.title(title)

     fig, ax = plt.subplots(nrows = 2,ncols = 2,figsize = (11,11))
     for i in range(len(numerical_features)):
         plt.subplot(2,2,i+1)
         sns.distplot(data[numerical_features[i]],)
         title = 'Distribution : ' + numerical_features[i]
         plt.title(title)
     plt.show()
```


Distribution : Gender
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3
4
Gender



- **Gender**, the only categorical feature, data displays a **normal distribution**.
- Distribution of **Age** and **Annual Income (k$)** is **positively or rightly skewed**.
- **Spending Score (1-100)** data distribution is similar to the **Head and Shoulder** pattern observed in stock charts.
- It displays a stock's price rising to a peak and then declines back to the base of the prior up-move. Something similar can be observed with the **2 shoulders** forming around the values **20 & 80** with the head being centered in between **40 - 60**.
- We will drop the **CustomerID** feature as it is just a number that is tagged to a customer.

In [2]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
# For interactive visualizations
import seaborn as sns
plt.style.use('fivethirtyeight')
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
from plotly import tools
init_notebook_mode(connected = True)
import plotly.figure_factory as ff
```

Load Dataset

In [4]:
```
# importing the dataset
data = pd.read_csv('../input/customer-segmentation-tutorial-in-python/Mall_Customers.csv')
data.shape
```

Out[4]:
```
(200, 5)
```

Review Dataset

In [5]:
```
# Let's see top 5 dataset
data.head()
```

Out[5]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [6]:
```
# Let's see last 5 datasets
data.tail()
```

Out[6]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID                200 non-null int64
Gender                    200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [8]:
```
data.describe()
```

Out[8]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Data Visualization

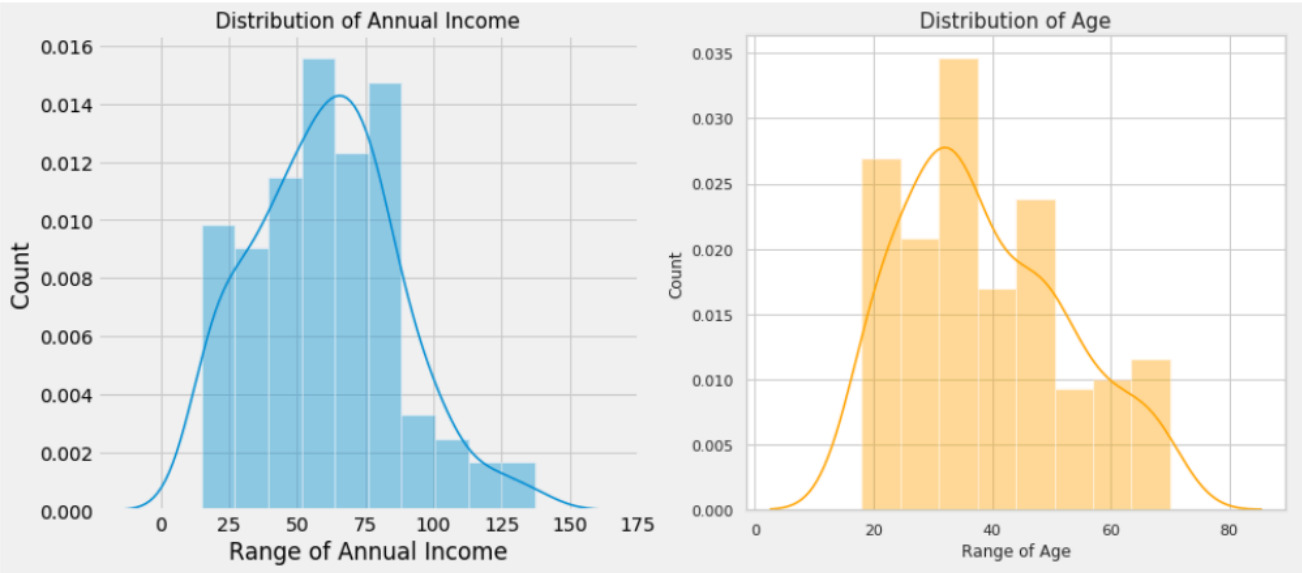
In [9]:

```
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (14, 6)

plt.subplot(1, 2, 1)
sns.set(style = 'whitegrid')
sns.distplot(data['Annual Income (k$)'])
plt.title('Distribution of Annual Income', fontsize = 15)
plt.xlabel('Range of Annual Income')
plt.ylabel('Count')


plt.subplot(1, 2, 2)
sns.set(style = 'whitegrid')
sns.distplot(data['Age'], color = 'orange')
plt.title('Distribution of Age', fontsize = 15)
plt.xlabel('Range of Age')
plt.ylabel('Count')
plt.show()
```



Here, In the above Plots we can see the Distribution pattern of Annual Income and Age, By looking at the plots,

We can infer one thing that There are few people who earn more than 100 US Dollars. Most of the people have an earning of around 50-75 US Dollars. Also, we can say that the least Income is around 20 US Dollars.

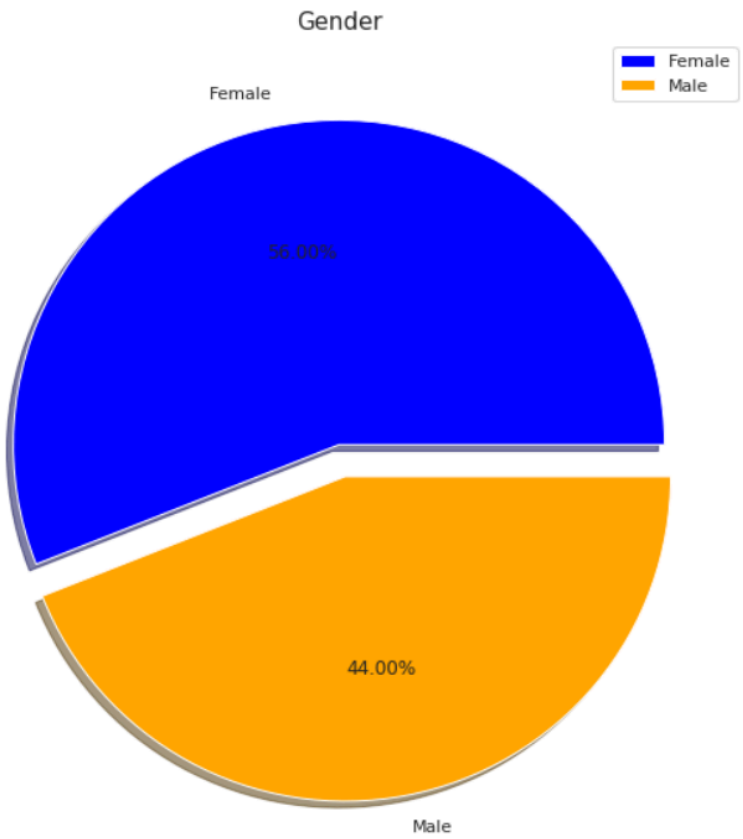
Taking inferences about the Customers.

The most regular customers for the Mall are around 30-35 years of age. Whereas the senior citizens age group is the least frequent visitor in the Mall. Youngsters are lesser in number as compared to the Middle aged people.

In [10]:

```
labels = ['Female', 'Male']
size = data['Gender'].value_counts()
colors = ['blue', 'orange']
explode = [0, 0.1]

plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(size, colors = colors, explode = explode, labels = labels, shadow = True, autopct =
'%.2f%%')
plt.title('Gender', fontsize = 15)
plt.axis('off')
plt.legend()
plt.show()
```

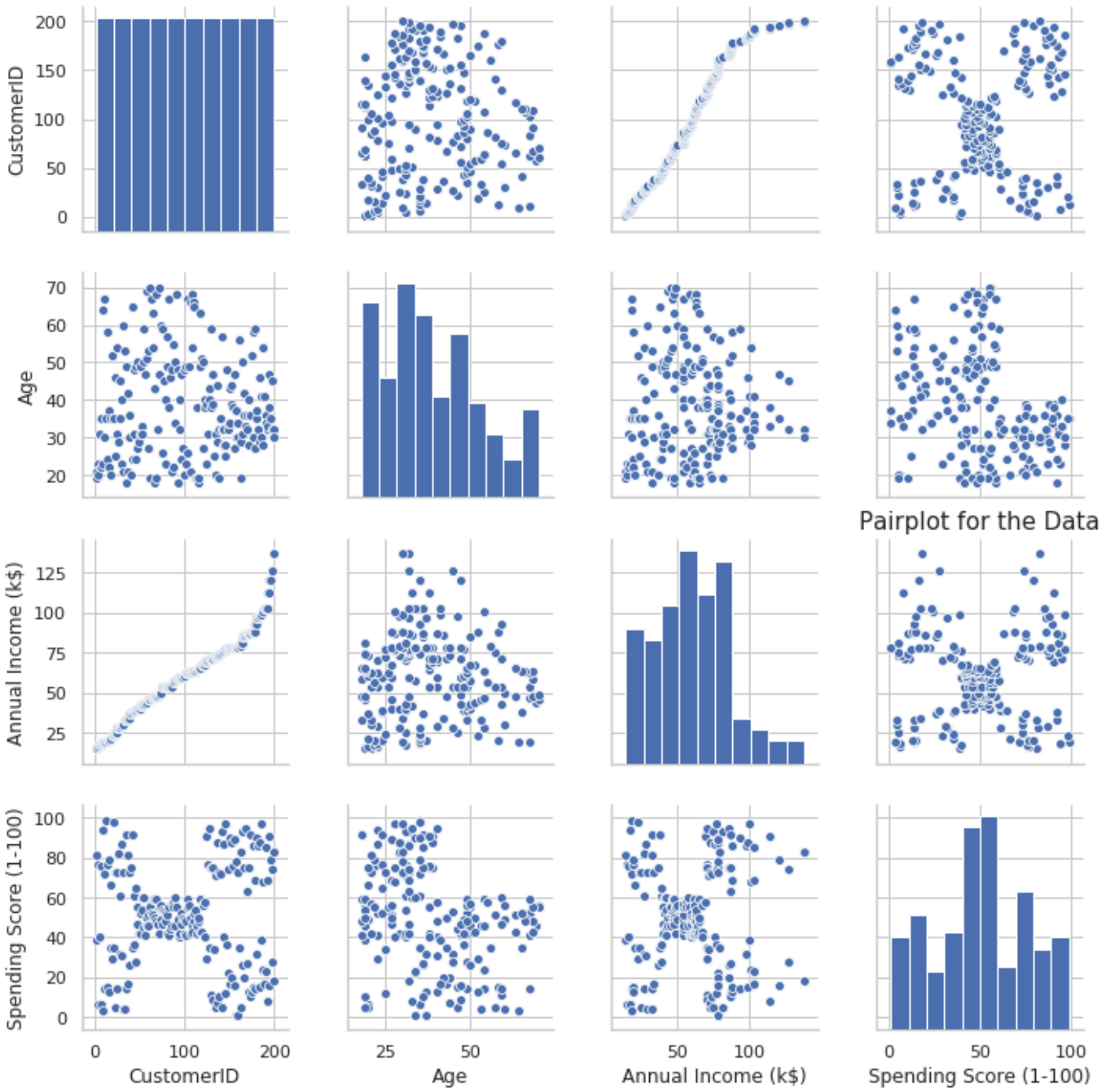


By looking at the above pie chart which explains about the distribution of Gender in the Mall

Interestingly, The Females are in the lead with a share of 56% whereas the Males have a share of 44%, that's a huge gap especially when the population of Males is comparatively higher than Females.

In [14]:

```
sns.pairplot(data)
plt.title('Pairplot for the Data', fontsize = 15)
plt.show()
```

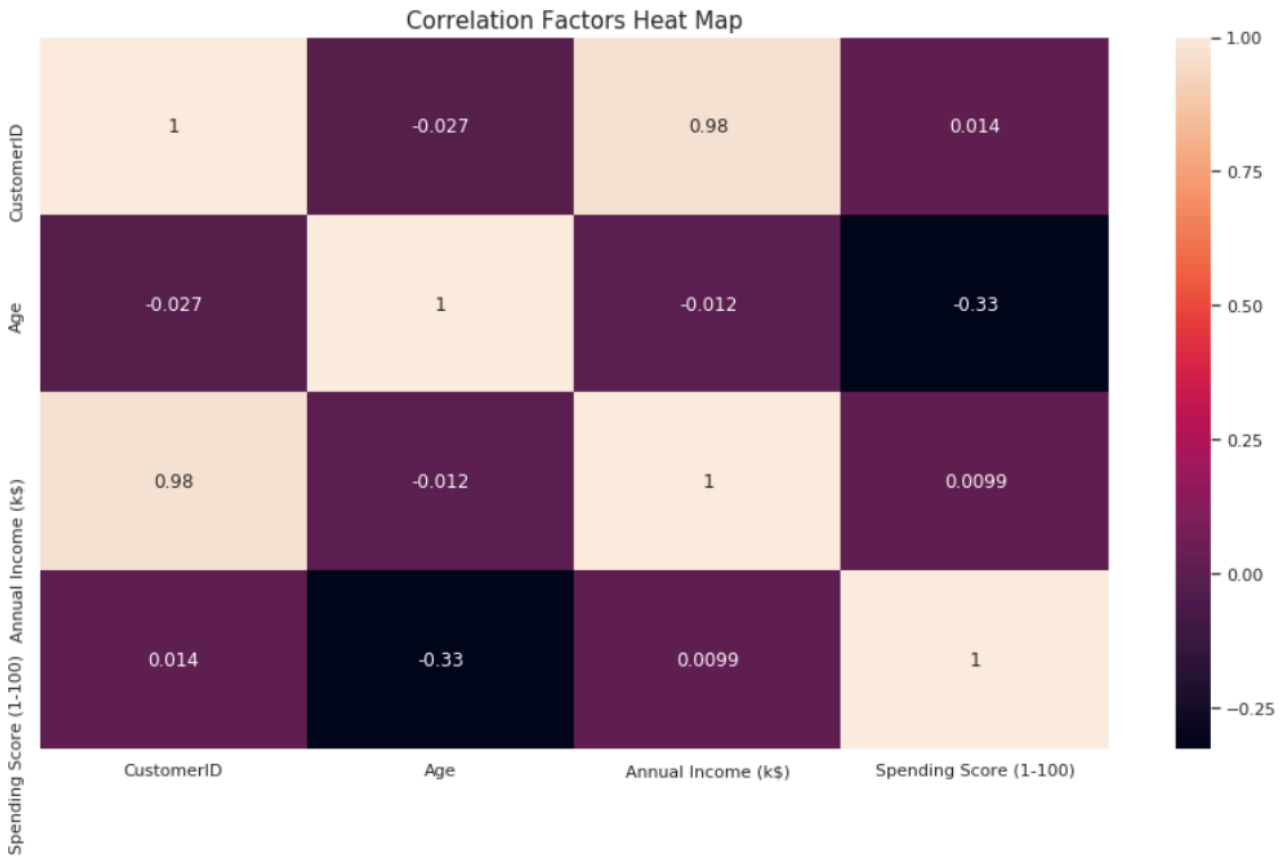


In [15]:

```
## Correlation coeffecients heatmap
sns.heatmap(data.corr(), annot=True).set_title('Correlation Factors Heat Map', size='15')
```

Out[15]:

```
Text(0.5, 1.0, 'Correlation Factors Heat Map')
```



The Above Graph for Showing the correlation between the different attributes of the Mall Customer Segmentation Dataset, This Heat map reflects the most correlated features with Orange Color and least correlated features with yellow color.

We can clearly see that these attributes do not have good correlation among them, that's why we will proceed with all of the features.

In [16]:

```
plt.rcParams['figure.figsize'] = (16, 8)
sns.stripplot(data['Gender'], data['Age'], palette = 'Purples', size = 10)
plt.title('Gender vs Spending Score', fontsize = 15)
plt.show()
```

Clustering Analysis

In [19]:

```
x = data.iloc[:, [3, 4]].values

# let's check the shape of x
print(x.shape)
```

```
(200, 2)
```

Conclusion

Analyzing mall customer behavior using Python offers valuable insights into consumer patterns and preferences. By leveraging Python's data analysis libraries such as NumPy, pandas, and matplotlib, businesses can delve deep into customer data to uncover trends, make informed decisions, and optimize their operations.

Python enables the processing of vast datasets efficiently, allowing businesses to segment customers based on demographics, shopping habits, and purchasing power. Techniques like data visualization with matplotlib help visualize these insights, making complex data more understandable and actionable.

Furthermore, Python's machine learning capabilities with libraries like scikit-learn enable businesses to predict customer behavior, recommend personalized offers, and optimize inventory management. This predictive capability enhances customer satisfaction and operational efficiency, ultimately driving profitability.

In conclusion, Python empowers businesses in the retail sector to harness the power of data for strategic decision-making and operational excellence, ultimately fostering long-term customer loyalty and business growth.

QUESTIONS/ANSWERED ABOUT PROJECTS

Q1: What is the objective of the mall customer report project?

- The objective is to analyze and understand customer behavior within a mall environment using data-driven insights.

Q2: What kind of data would be collected or analyzed in this project?

- Data such as customer demographics (age, gender), arrival times, shopping durations, spending habits, and possibly feedback or survey responses could be collected and analyzed.

Q3: How would Python be used in this project?

- Python would be used to manipulate and analyze data, generate statistical summaries, create visualizations (using libraries like matplotlib or seaborn), and potentially build predictive models to forecast customer behavior.

Q4: What are some key metrics or insights that could be derived from the analysis?

- Key metrics could include average time spent in the mall, popular times of day for shopping, average spending per customer segment, correlations between demographics and purchasing behavior, and customer retention rates.

Q5: How could the insights from this project benefit mall management or retailers?

- Insights could help optimize staffing schedules based on peak shopping times, tailor marketing strategies to different customer segments, improve store layouts or product placements, and enhance overall customer satisfaction and loyalty.

Q6: What are some challenges or considerations when conducting such an analysis?

- Challenges may include ensuring data privacy and security, cleaning and preprocessing large datasets, dealing with missing or inconsistent data, and interpreting results accurately to derive actionable insights.

Q7: Can machine learning be applied to this project? If so, how?

- Yes, machine learning techniques could be applied to predict customer preferences, segment customers into different groups based on behavior patterns, or forecast future trends in customer traffic or spending.

Q8: How would you present the findings and recommendations from this project?

- Findings could be presented through visual dashboards, reports summarizing key insights and recommendations, and interactive presentations to stakeholders.

Q9: What are potential future enhancements or extensions to this project?

- Future enhancements could include real-time data analysis for dynamic decision-making, integrating external data sources (e.g., weather, local events) for more contextual insights, and expanding the scope to include online and offline customer interactions.

Q10: What are some ethical considerations when analyzing customer data for this project?

- Ethical considerations include obtaining informed consent for data collection, ensuring data anonymization where necessary, and using data responsibly to avoid bias or discrimination.

Thank You!