Importing libraries

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

```
C:\Users\Srushti\anaconda3\Lib\site-packages\pandas\core\arrays\ma
sked.py:60: UserWarning: Pandas requires version '1.3.6' or newer
of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

Reading file in python

In [2]:
```
data=pd.read_csv(r"C:\Users\Srushti\OneDrive\excel files\iphone dat
print(data)
```

```
     Gender  Age  Salary  Purchase Iphone
0      Male   19   19000                0
1      Male   35   20000                0
2    Female   26   43000                0
3    Female   27   57000                0
4      Male   19   76000                0
..      ...  ...     ...              ...
395  Female   46   41000                1
396    Male   51   23000                1
397  Female   50   20000                1
398    Male   36   33000                0
399  Female   49   36000                1

[400 rows x 4 columns]
```

Getting information about my data

Finding top rows

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

Out[3]:

	Gender	Age	Salary	Purchase Iphone
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0

Finding last rows

In [4]:
```
data.tail()
```

Out[4]:

	Gender	Age	Salary	Purchase Iphone
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

Finding total number of rows and columns

In [5]:
```
data.shape
```

Out[5]:
```
(400, 4)
```

Finding info of the given data

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Gender           400 non-null    object
 1   Age              400 non-null    int64 
 2   Salary           400 non-null    int64 
 3   Purchase Iphone  400 non-null    int64 
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

Describing the data

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

Out[7]:

	Age	Salary	Purchase Iphone
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

Finding name of all columns

In [8]:
```
data.columns
```

Out[8]:
```
Index(['Gender', 'Age', 'Salary', 'Purchase Iphone'], dtype='objec
t')
```

Data Preprocessing

Missing values

In [9]:
```
data.isna().sum()
```

Out[9]:
```
Gender             0
Age                0
Salary             0
Purchase Iphone    0
dtype: int64
```

As we can see here there are no missing values present inside the dataset

Duplicates Values

In [10]:
```
duplicate=data.duplicated()
sum(duplicate)
```

Out[10]:
```
20
```

As here are duplicate values we will use drop commands to drop the duplicates values

In [11]:
```
data=data.drop_duplicates()
data
```

Out[11]:

	Gender	Age	Salary	Purchase Iphone
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
...	...	...	...	...
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

380 rows × 4 columns

Checking if the duplicated values are removed or not

In [12]:
```
duplicate=data.duplicated()
sum(duplicate)
```

Out[12]: 0

Label-Encoding

We would prefrom this to convert the categorical (text) data into numbers, so that machine learning models can understand it.

In [13]:
```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
label
```

Out[13]:

▾ LabelEncoder ⓘ ? (https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.Lab

LabelEncoder()

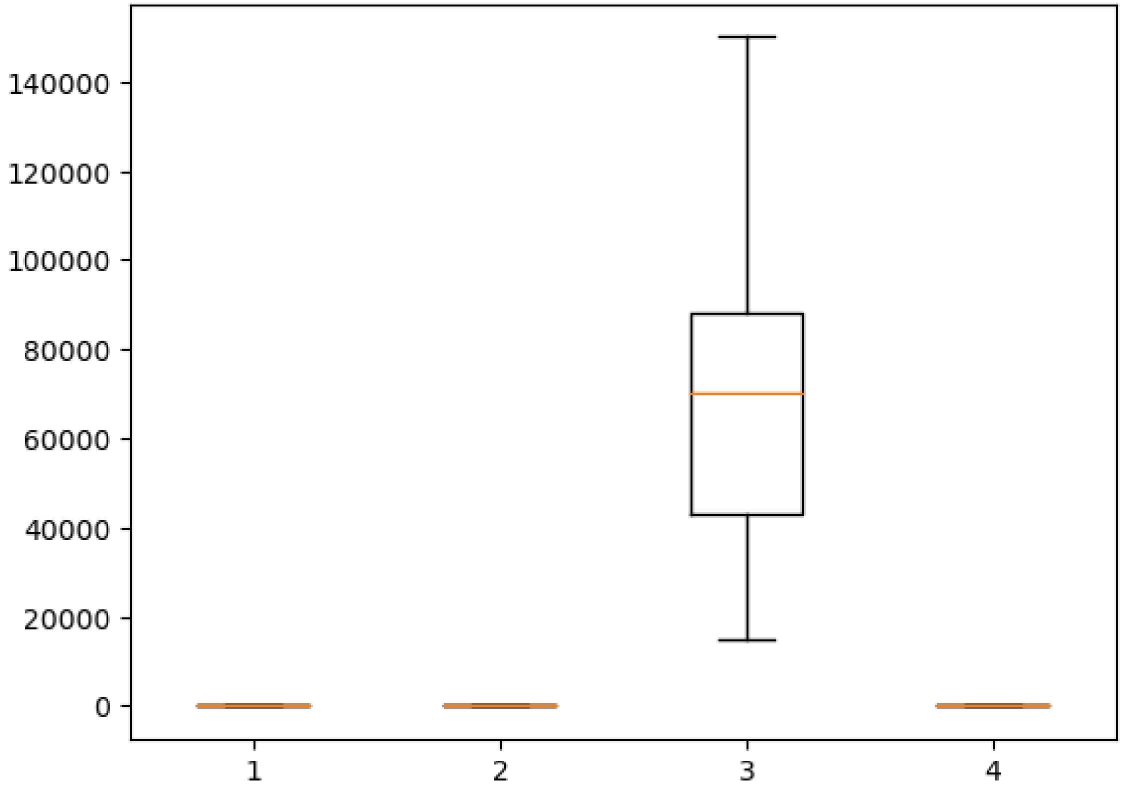
In [14]:
```
data["Gender"]=label.fit_transform(data["Gender"])
data.head()
```

Out[14]:

	Gender	Age	Salary	Purchase Iphone
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

Outlier Removal

In [15]:
```
fig,ax=plt.subplots()
ax.boxplot(data.iloc[:,:])
plt.show()
```



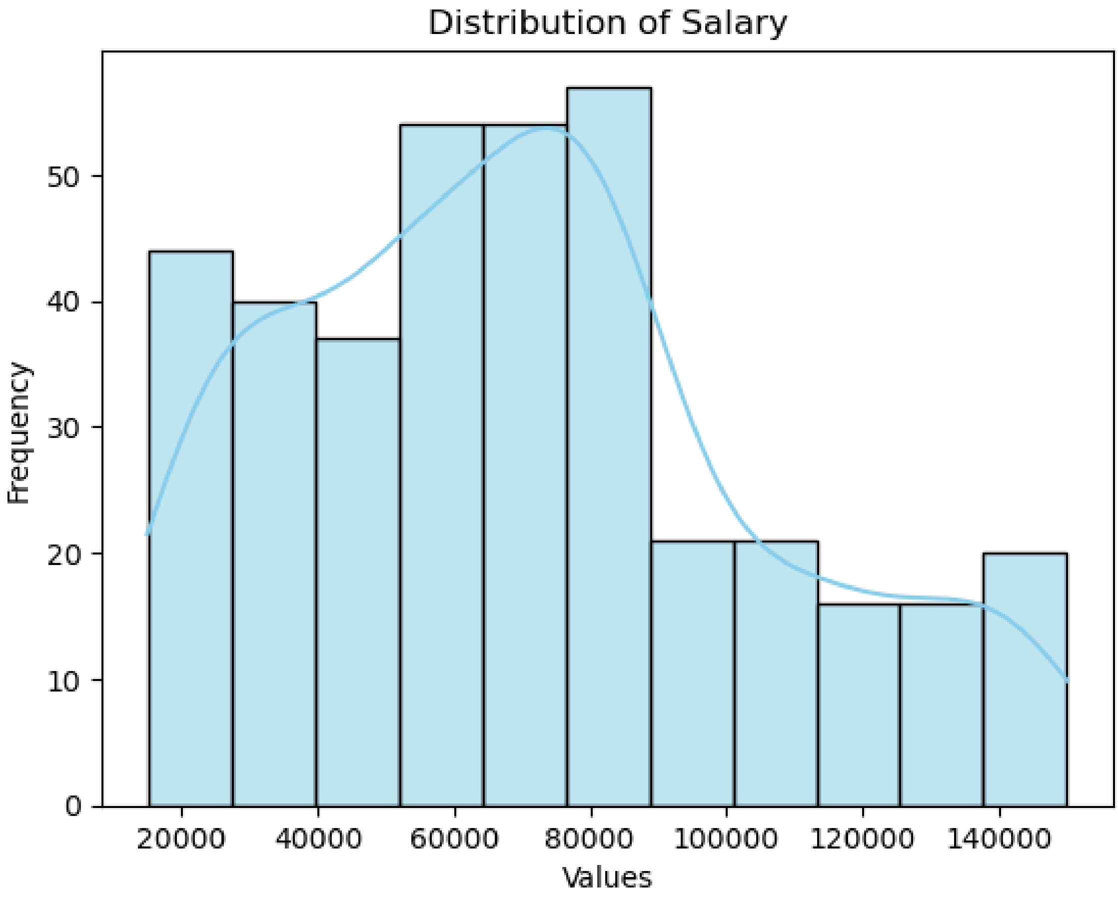
There are no outliers present in this plot

Visualisation

Histogram- View distribution of numeric data.

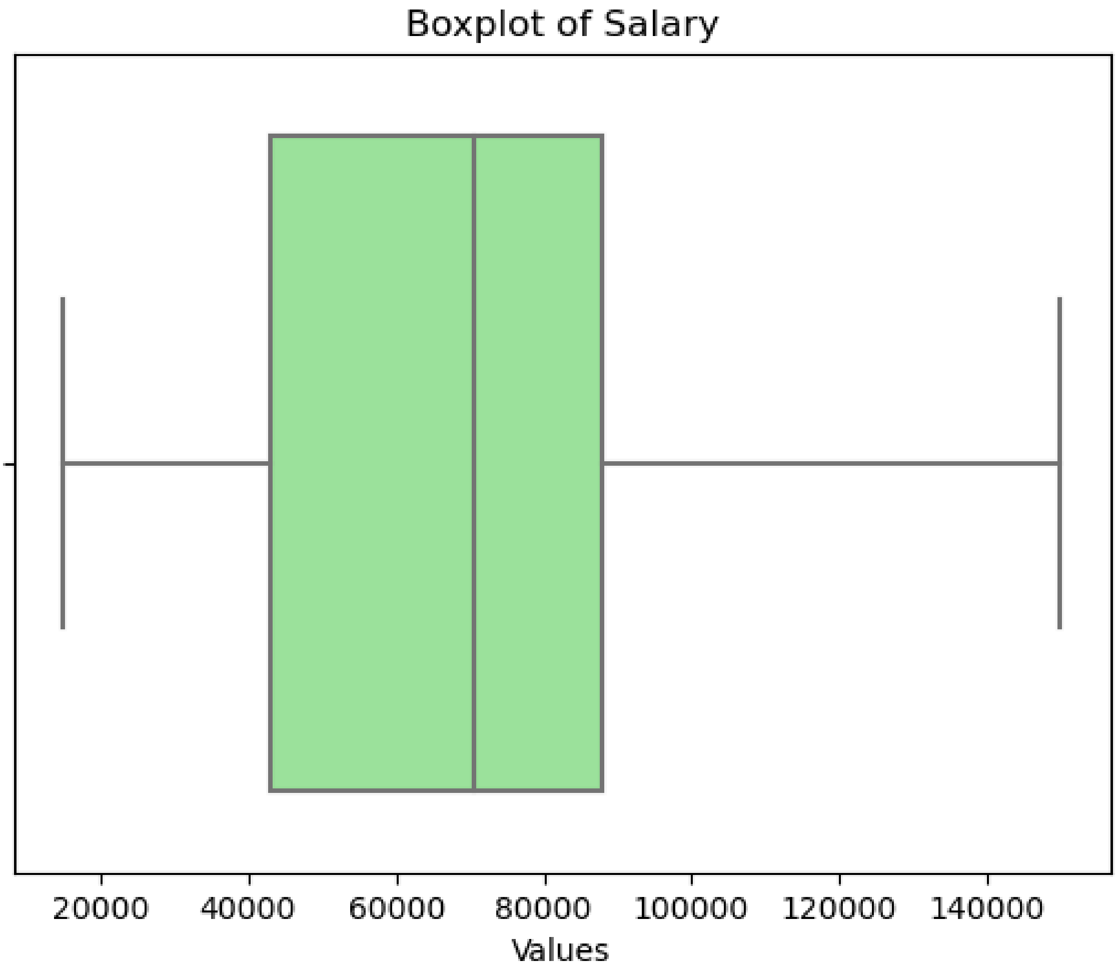
In [16]:
```
sns.histplot(data['Salary'], kde=True, color='skyblue')
plt.title("Distribution of Salary")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```

```
C:\Users\Srushti\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1
119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



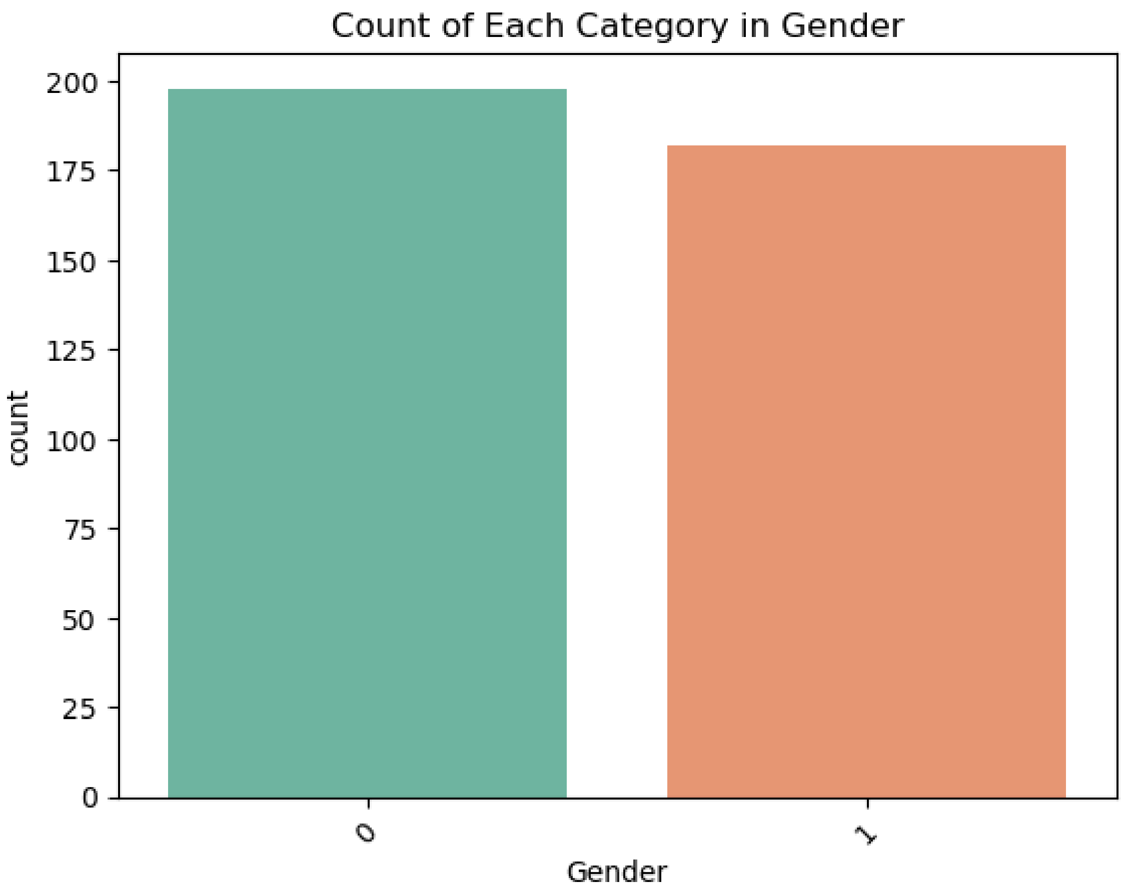
Boxplot-Detect outliers and spread of numeric data

In [17]:
```
sns.boxplot(x=data['Salary'], color='lightgreen')
plt.title("Boxplot of Salary")
plt.xlabel("Values")
plt.show()
```



Countplot-Show counts of each category. Used when you want to see frequency of labels.

In [18]:
```
sns.countplot(data=data, x='Gender', palette='Set2')
plt.title("Count of Each Category in Gender")
plt.xticks(rotation=45)
plt.show()
```

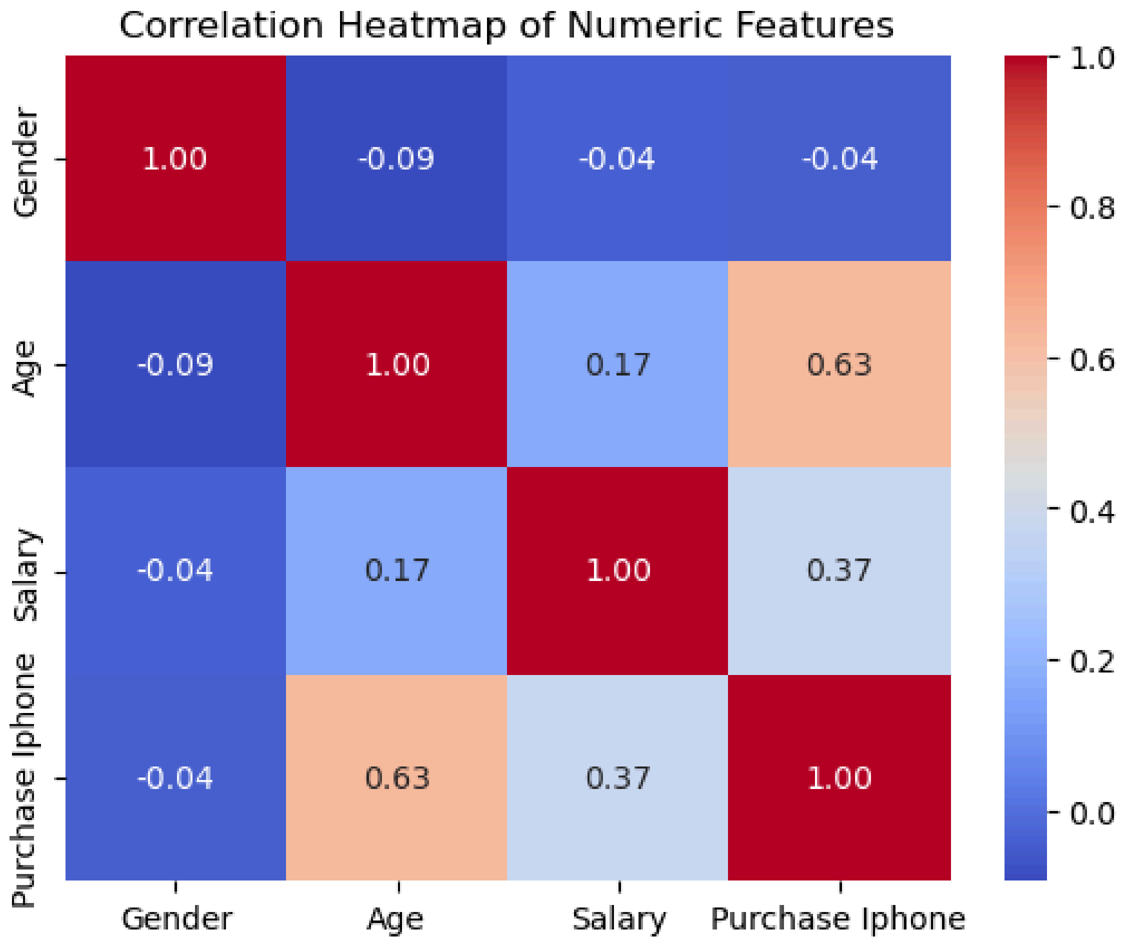


Barplot-Compare categories based on a numeric value (like average).We can use it when we want to compare groups

In [19]:
```
sns.barplot(data=data, x='Gender', y='Salary', palette='pastel')
plt.title("Bar Plot of Salary by Gender")
plt.xticks(rotation=45)
plt.show()
```

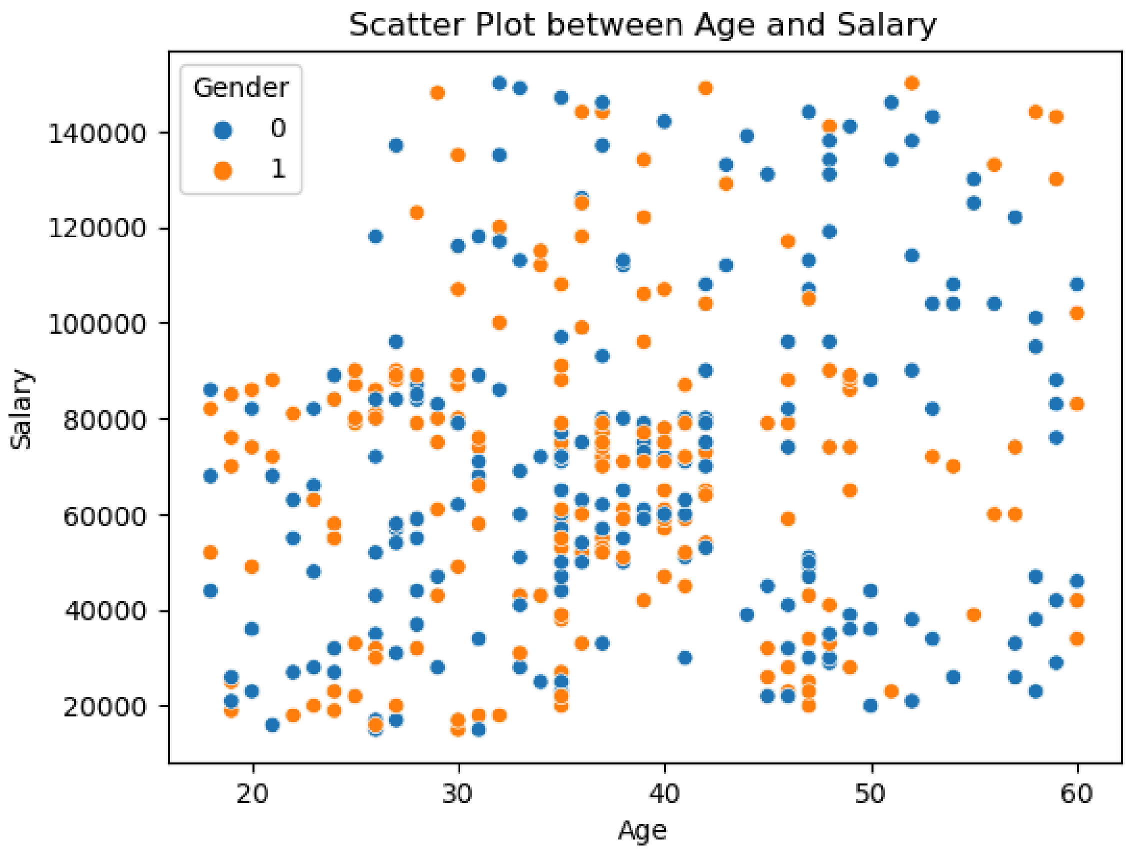
Heatmap- Show correlation between numeric features

In [20]:
```
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwar
plt.title("Correlation Heatmap of Numeric Features")
plt.show()
```



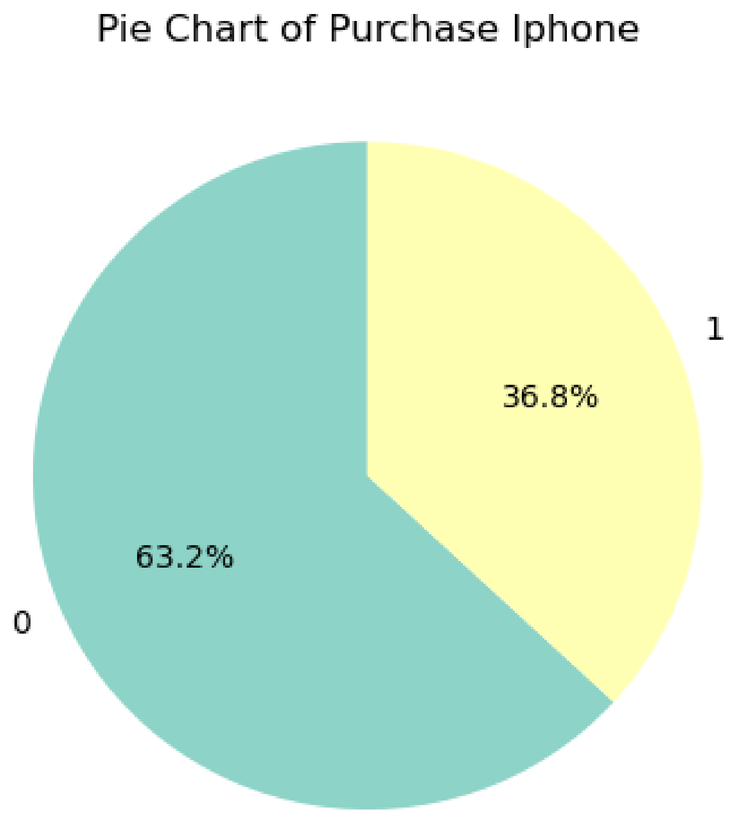
Scatter Plot-Visualize relationship between two numeric variables

In [21]:
```
sns.scatterplot(data=data, x='Age', y='Salary', hue='Gender')
plt.title("Scatter Plot between Age and Salary")
plt.show()
```



Pie Chart-Show proportion/percentage of categories. Used when you want to visualize share of each group.

In [22]:
```
data['Purchase Iphone'].value_counts().plot.pie(
    autopct='%1.1f%%', startangle=90, colors=sns.color_palette("Set
)
plt.title("Pie Chart of Purchase Iphone")
plt.ylabel("")  # Remove y-axis label
plt.show()
```



Insights of the visuals

1.Histogram – Salary Distribution

Most people earn below ₹80,000, with the salary data slightly right-skewed showing a few high earners.

2.Boxplot – Salary

The salary data has no strong outliers, and most salaries lie between ₹43,000 and ₹88,000.

3.Countplot – Gender

The dataset has a nearly equal number of males and females, ensuring a balanced gender distribution.

4. Barplot – Gender vs Salary

Males have a slightly higher average salary than females, but the difference is not very large.

5. Heatmap – Correlation

Salary is positively correlated with iPhone purchase, meaning higher income increases the chance of buying.

6. Scatter Plot – Age vs Salary

There’s no strong trend, but younger people tend to have lower salaries than older ones.

7. Pie Chart – Purchase iPhone

Only 36% of people bought an iPhone, indicating an imbalance in the target variable.

EDA

statistical analysis

first moment of business decision

In [23]:
```
data.mean()
```

Out[23]:
```
Gender                  0.478947
Age                    37.586842
Salary              70421.052632
Purchase Iphone         0.368421
dtype: float64
```

In [24]:
```
data.median()
```

Out[24]:
```
Gender                0.0
Age                  37.0
Salary            70500.0
Purchase Iphone       0.0
dtype: float64
```

In [25]:
```
from scipy import stats
stats.mode(data)
```

Out[25]:
```
ModeResult(mode=array([    0,    35, 72000,     0], dtype=int64),
count=array([198,  31,  10, 240], dtype=int64))
```

second moment of business decision

In [26]:
```
data.std()
```

Out[26]:
```
Gender                 0.500215
Age                   10.592492
Salary             34604.155483
Purchase Iphone        0.483012
dtype: float64
```

In [27]:
```
data.var()
```

Out[27]:
```
Gender             2.502152e-01
Age                1.122009e+02
Salary             1.197448e+09
Purchase Iphone    2.333009e-01
dtype: float64
```

In [28]:
```
range=max(data.Salary)-min(data.Salary)
range
```

Out[28]:
```
135000
```

third moment of business decision

In [29]:
```
data.skew()
```

Out[29]:
```
Gender             0.084620
Age                0.239843
Salary             0.461275
Purchase Iphone    0.547709
dtype: float64
```

fourth moment of business decision

In [30]:
```
data.kurtosis()
```

Out[30]:
```
Gender            -2.003412
Age               -0.674733
Salary            -0.490309
Purchase Iphone   -1.709038
dtype: float64
```