In [1]:
```
# Import the necessary library for Data manipulation and visualization
import pandas as pd #for laoding and data manipulation
import numpy as np #for mathematical operations
import matplotlib.pyplot as plt #for graphical representation of data
import seaborn as sns #for graphical representation of data
```

In [2]:
```
df = pd.read_csv('C:/Users/Sanayak/Desktop/customer_purchase_data.csv')
print(df.head(10))#inspect the first 10 entries of our dataset
```

```
   Age  Gender   AnnualIncome  NumberOfPurchases  ProductCategory  \
0   40       1   66120.267939                  8                0   
1   20       1   23579.773583                  4                2   
2   27       1  127821.306432                 11                2   
3   24       1  137798.623120                 19                3   
4   31       1   99300.964220                 19                1   
5   66       1   37758.117475                 14                4   
6   39       1  126883.385286                 16                3   
7   64       1   39707.359724                 13                2   
8   43       0  102797.301269                 20                1   
9   20       1   63854.921080                 16                0   

   TimeSpentOnWebsite  LoyaltyProgram  DiscountsAvailed  PurchaseStatus  
0           30.568601               0                 5               1  
1           38.240097               0                 5               0  
2           31.633212               1                 0               1  
3           46.167059               0                 4               1  
4           19.823592               0                 0               1  
5           17.827493               0                 2               0  
6           42.085384               1                 4               1  
7           17.190292               1                 0               0  
8            6.023475               0                 3               0  
9           38.572466               0                 5               1  
```

In [3]:
```
print(df.duplicated().sum())#check for duplicate entries
```

```
112
```

In [4]:
```
df.drop_duplicates(inplace=True)#drop the duplicated entries and update the dataset
print(df.duplicated().sum())
```

```
0
```

In [5]:
```
print(df.info())#call the .info()function and inspect the data type
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1388 entries, 0 to 1499
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Age                 1388 non-null   int64  
 1   Gender              1388 non-null   int64  
 2   AnnualIncome        1388 non-null   float64
 3   NumberOfPurchases   1388 non-null   int64  
 4   ProductCategory     1388 non-null   int64  
 5   TimeSpentOnWebsite  1388 non-null   float64
 6   LoyaltyProgram      1388 non-null   int64  
 7   DiscountsAvailed    1388 non-null   int64  
 8   PurchaseStatus      1388 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 108.4 KB
None
```

In [6]:

```
df.describe().T#call the .describe()function to check for statistical information a
```

Out[6]:

| | count | mean | std | min | 25% |
|---|---|---|---|---|---|
| **Age** | 1388.0 | 43.939481 | 15.487533 | 18.000000 | 30.750000 |
| **Gender** | 1388.0 | 0.501441 | 0.500178 | 0.000000 | 0.000000 |
| **AnnualIncome** | 1388.0 | 84699.045444 | 37541.136478 | 20001.512518 | 53766.895806 |
| **NumberOfPurchases** | 1388.0 | 10.548991 | 5.869383 | 0.000000 | 6.000000 |
| **ProductCategory** | 1388.0 | 2.002882 | 1.422851 | 0.000000 | 1.000000 |
| **TimeSpentOnWebsite** | 1388.0 | 30.747545 | 16.976852 | 1.037023 | 16.379635 |
| **LoyaltyProgram** | 1388.0 | 0.333573 | 0.471659 | 0.000000 | 0.000000 |
| **DiscountsAvailed** | 1388.0 | 2.609510 | 1.699984 | 0.000000 | 1.000000 |
| **PurchaseStatus** | 1388.0 | 0.466859 | 0.499080 | 0.000000 | 0.000000 |

The above statistical information shows that the Dataset has no outlier as the discrepancy between the mean and the 50thpercentile (median) can be ignored

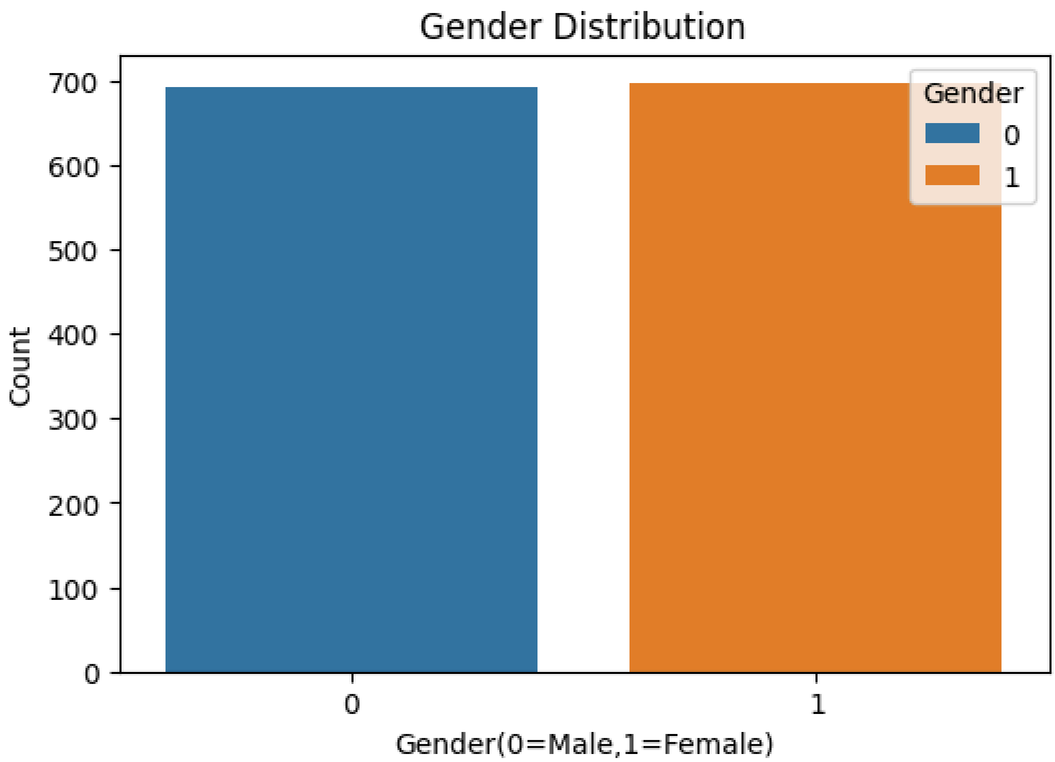
In [7]:

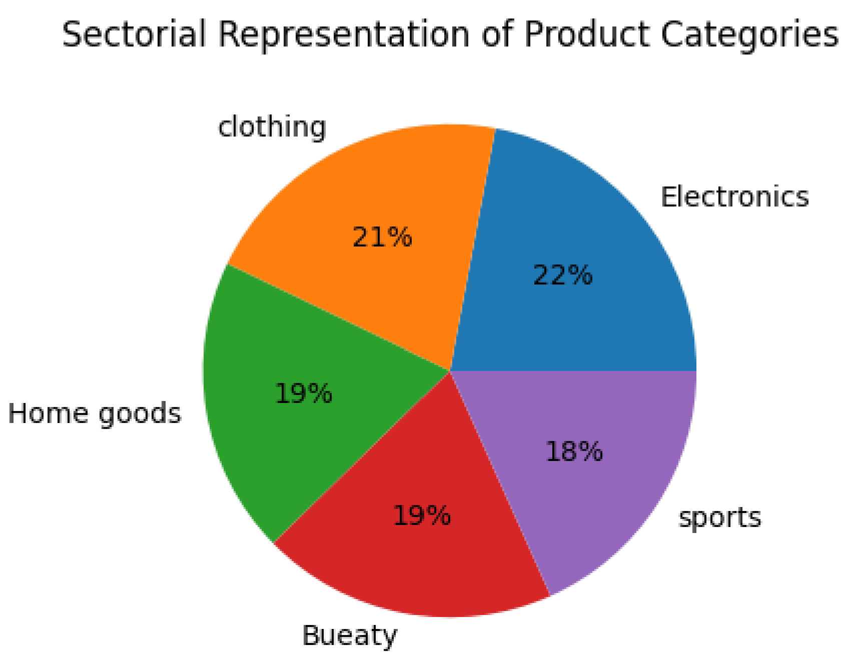
```
df[['AnnualIncome','TimeSpentOnWebsite']] = df[['AnnualIncome','TimeSpentOnWebsite'
print(df.head())#roundoff the Annualincome & Time spent on website column to 2decim
```

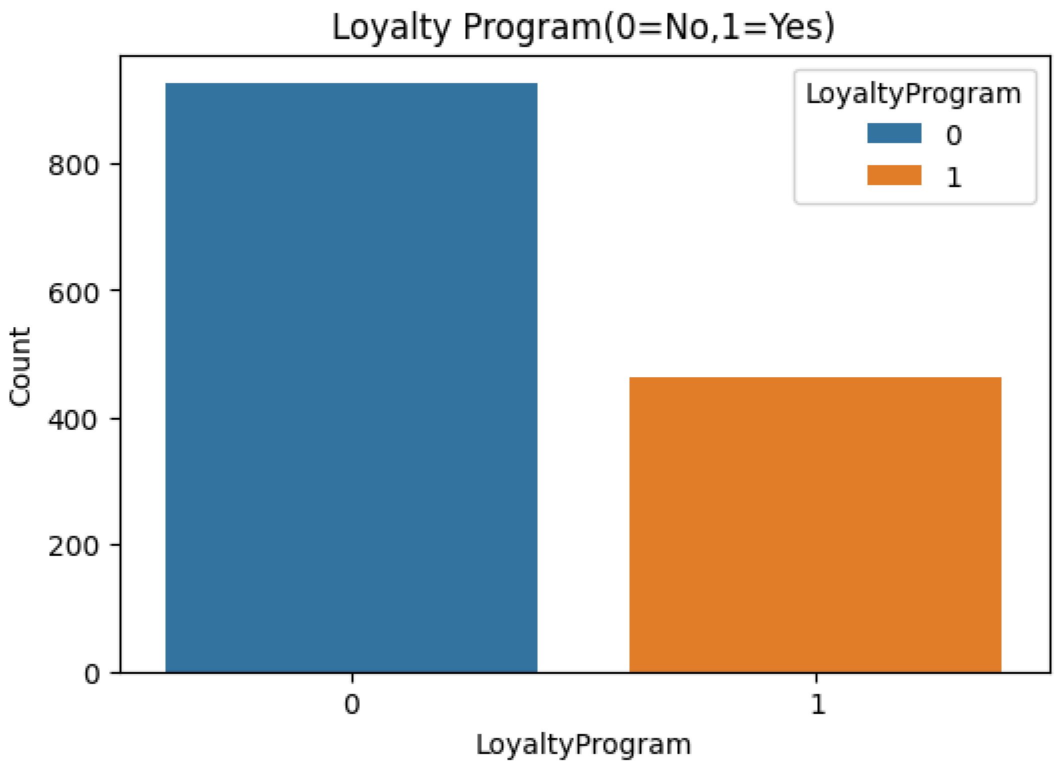
```
   Age  Gender  AnnualIncome  NumberOfPurchases  ProductCategory  \
0   40       1      66120.27                  8                0
1   20       1      23579.77                  4                2
2   27       1     127821.31                 11                2
3   24       1     137798.62                 19                3
4   31       1      99300.96                 19                1

   TimeSpentOnWebsite  LoyaltyProgram  DiscountsAvailed  PurchaseStatus
0               30.57               0                 5               1
1               38.24               0                 5               0
2               31.63               1                 0               1
3               46.17               0                 4               1
4               19.82               0                 0               1
```

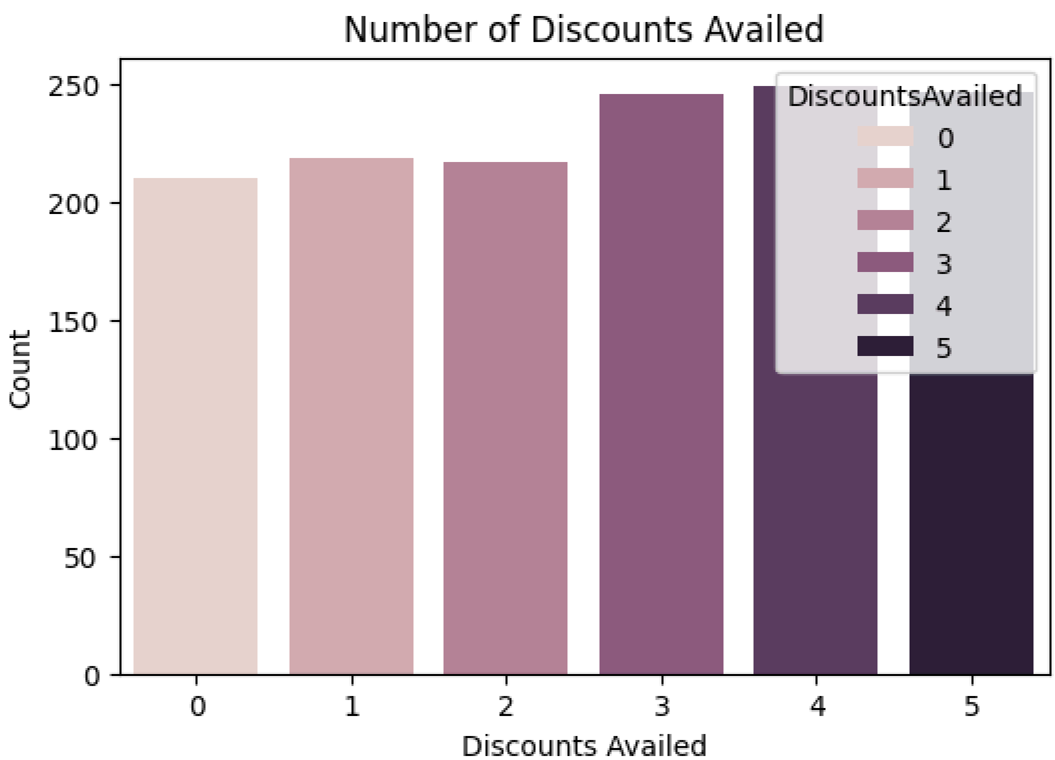
Data Visualization

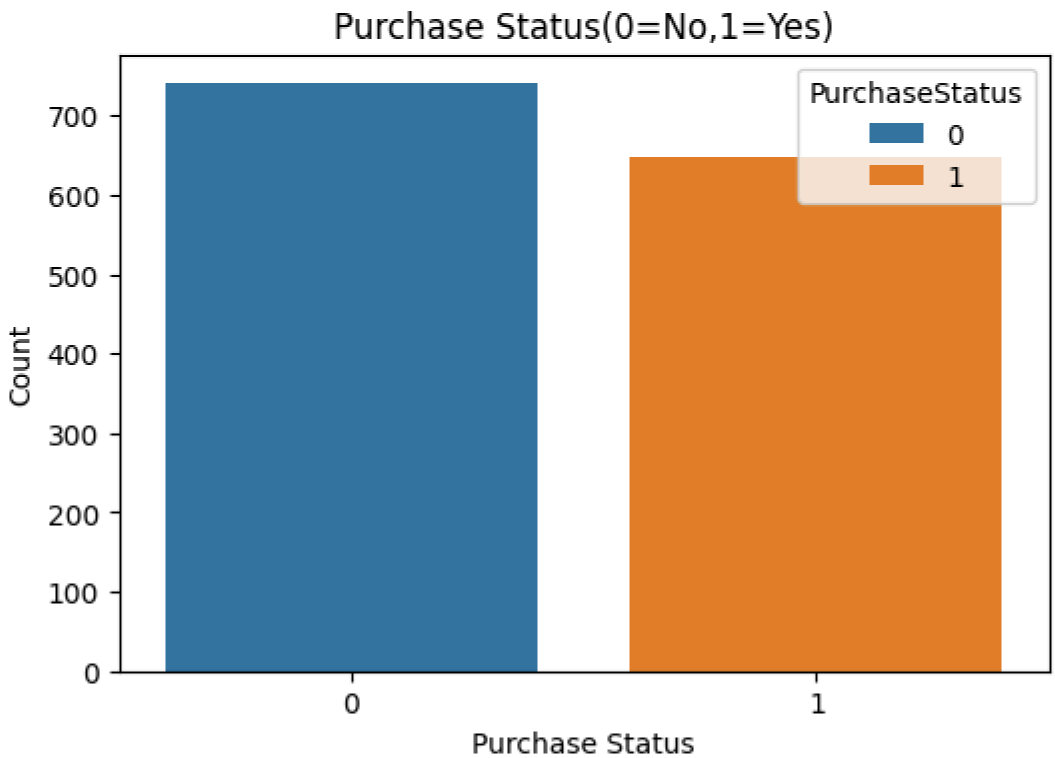
In [8]:
```
# Gender Distribution
plt.figure(figsize=(6,4))
sns.countplot(x='Gender',data=df,hue='Gender')
plt.title('Gender Distribution')
plt.xlabel('Gender(0=Male,1=Female)')
plt.ylabel('Count')
plt.show()

# Product Categories Distribution
product_cat_counts = df['ProductCategory'].value_counts()
labels = ['Electronics','clothing','Home goods','Bueaty','sports']
plt.figure(figsize=(6,4))
plt.pie(product_cat_counts,labels=labels,autopct="%1.f%%")
plt.title('Sectorial Representation of Product Categories')
plt.show()

# Loyalty Program Distribution
plt.figure(figsize=(6,4))
sns.countplot(x='LoyaltyProgram',data=df,hue='LoyaltyProgram')
plt.title('Loyalty Program(0=No,1=Yes)')
plt.xlabel('LoyaltyProgram')
plt.ylabel('Count')
plt.show()
# Number of Discounts Availed Distribution
plt.figure(figsize=(6,4))
sns.countplot(x='DiscountsAvailed',data=df,hue='DiscountsAvailed')
plt.title('Number of Discounts Availed')
plt.xlabel('Discounts Availed')
plt.ylabel('Count')
plt.show()
# Purchase Status Distribution
plt.figure(figsize=(6,4))
sns.countplot(x='PurchaseStatus',data=df,hue='PurchaseStatus')
plt.title('Purchase Status(0=No,1=Yes)')
plt.xlabel('Purchase Status')
plt.ylabel('Count')
plt.show()
```


Gender Distribution
Gender
0
1
Count
700
600
500
400
300
200
100
0
0
1
Gender(0=Male,1=Female)

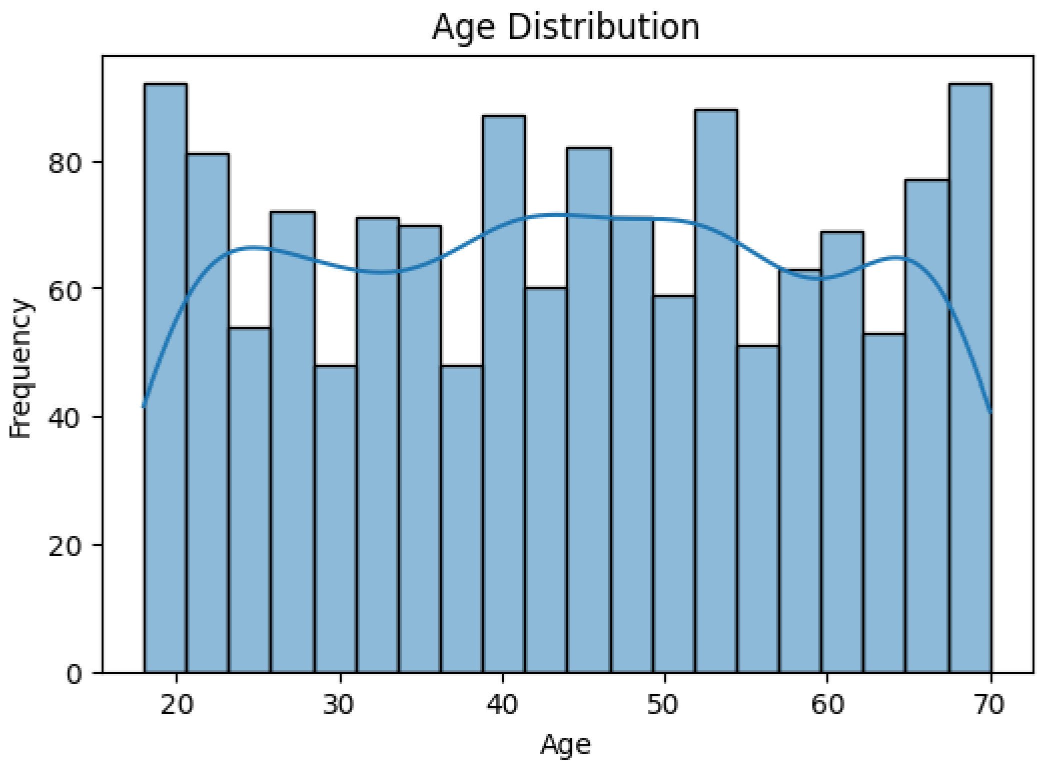

Sectorial Representation of Product Categories
clothing
Electronics
21%
22%
Home goods
19%
18%
19%
sports
Bueaty

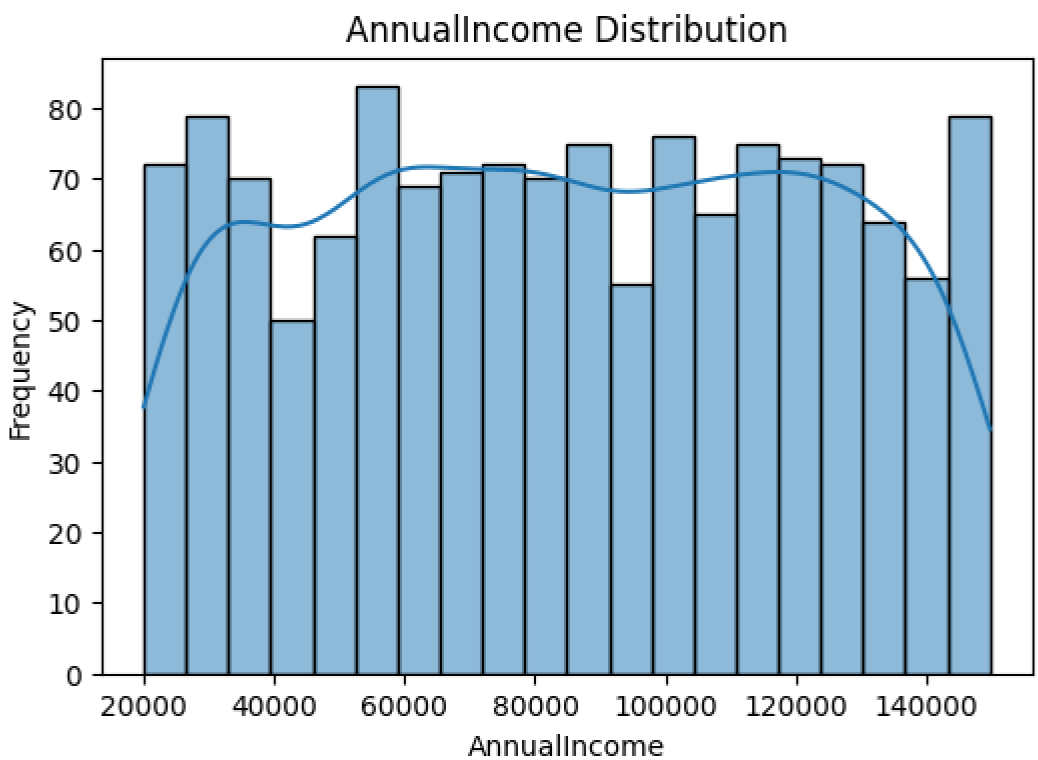

Loyalty Program(0=No,1=Yes)
LoyaltyProgram
0
1
Count
0
200
400
600
800
LoyaltyProgram

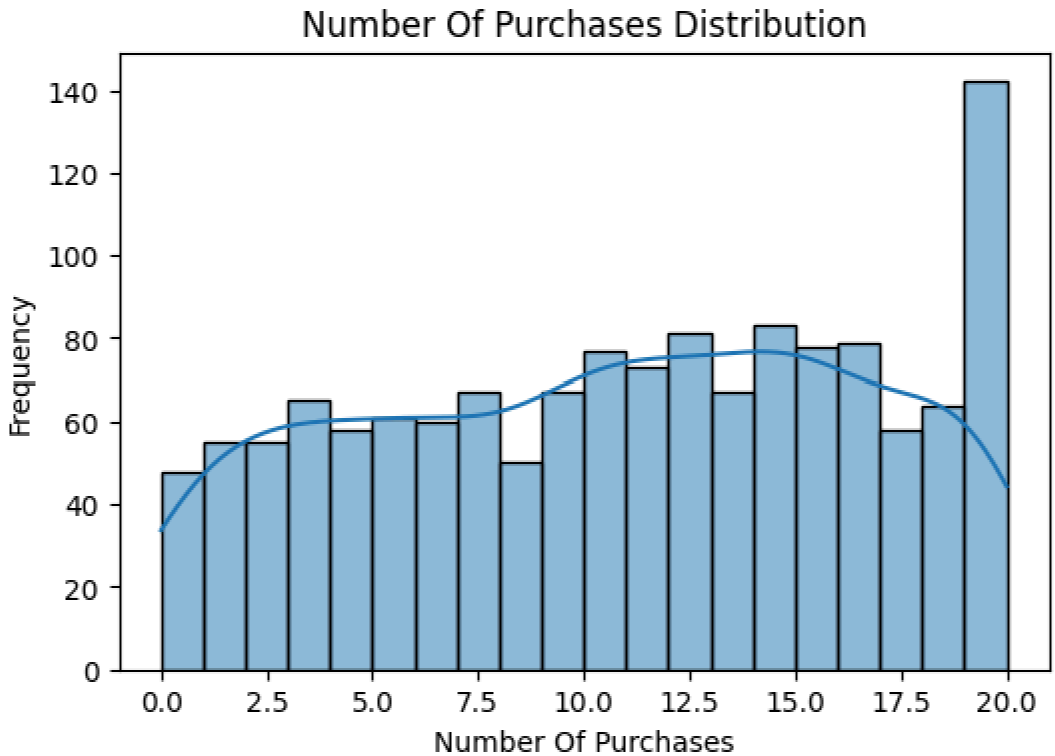

Number of Discounts Availed
DiscountsAvailed
0
1
2
3
4
5
Count
0
50
100
150
200
250
Discounts Availed

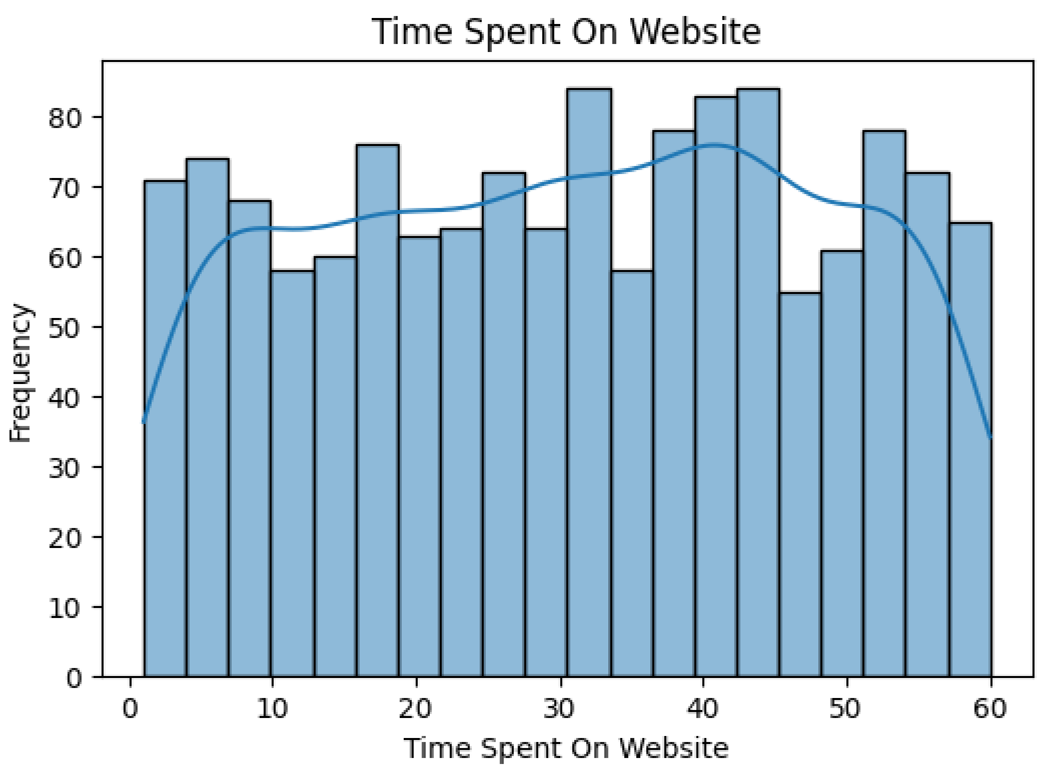



In [9]:
```
# Age Distribution
plt.figure(figsize=(6,4))
sns.histplot(df['Age'],kde=True,bins=20)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Annual Income Distribution
plt.figure(figsize=(6,4))
sns.histplot(df['AnnualIncome'],kde=True,bins=20)
plt.title('AnnualIncome Distribution')
plt.xlabel('AnnualIncome')
plt.ylabel('Frequency')
plt.show()
# Number of Purchases Distribution
plt.figure(figsize=(6,4))
sns.histplot(df['NumberOfPurchases'],kde=True,bins=20)
plt.title('Number Of Purchases Distribution')
plt.xlabel('Number Of Purchases')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(6,4))
sns.histplot(df['TimeSpentOnWebsite'],kde=True,bins=20)
plt.title('Time Spent On Website')
plt.xlabel('Time Spent On Website')
plt.ylabel('Frequency')
plt.show()
```


Age Distribution
Frequency
0
20
40
60
80
20
30
40
50
60
70
Age

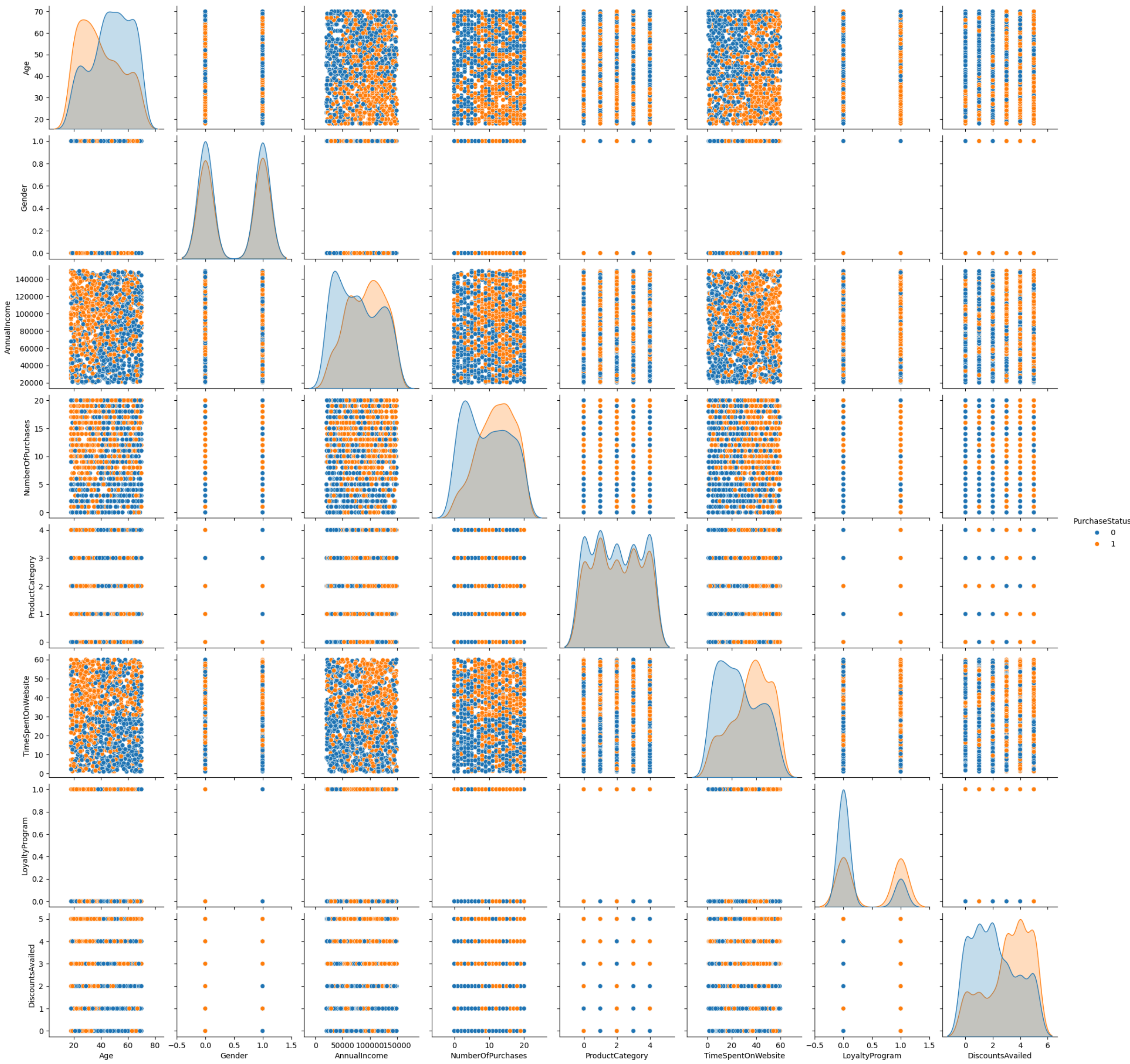

AnnualIncome Distribution
Frequency
0
10
20
30
40
50
60
70
80
20000
40000
60000
80000
100000
120000
140000
AnnualIncome



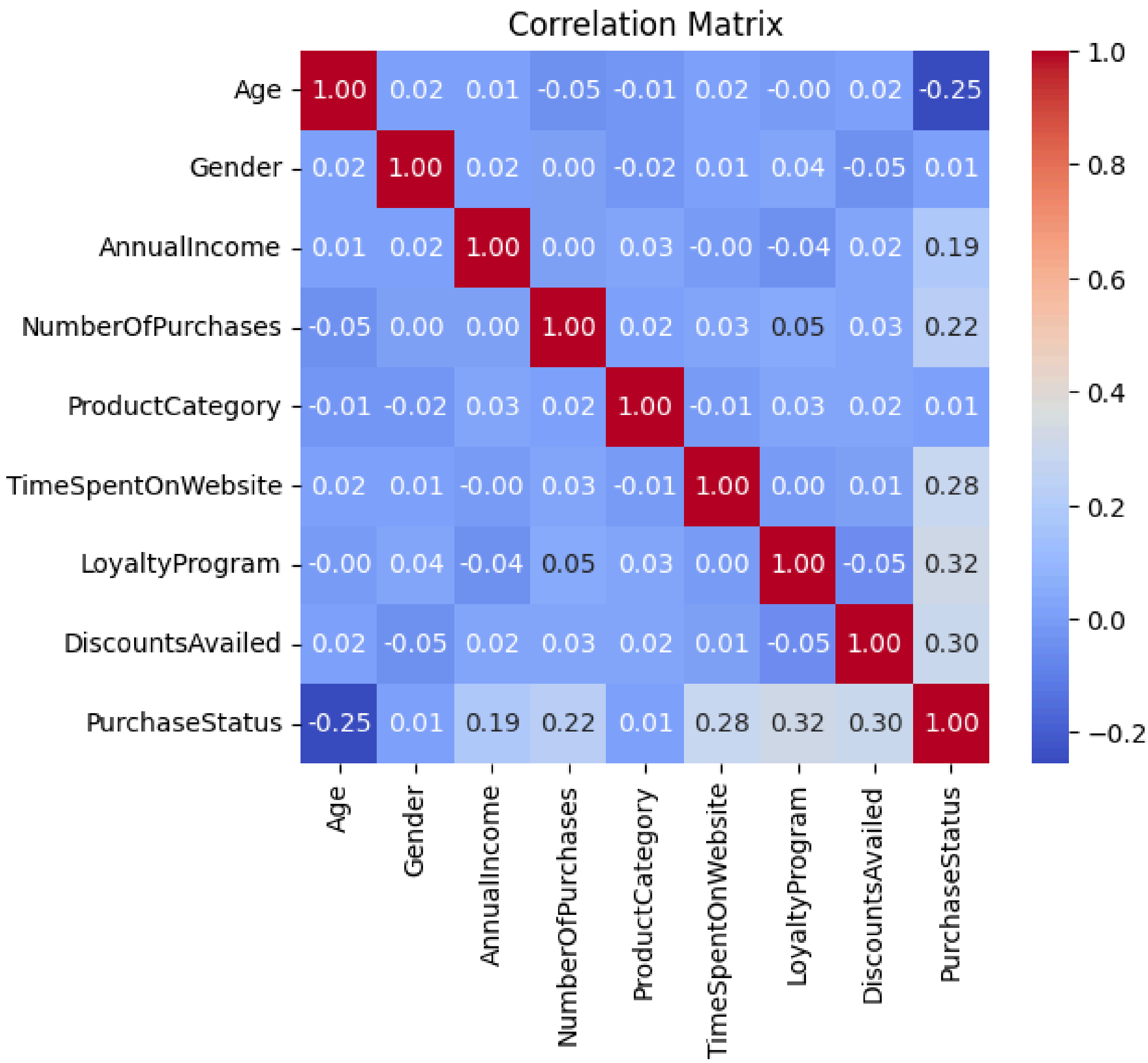

In [10]:
```
sns.pairplot(df,hue='PurchaseStatus')
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x28f7cd38b00>




In [11]:

```python
# check for Multicollinearity
plt.figure(figsize=(6,5))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm',fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```




The above Correlation matrix indicates that the features of the Dataset are not linearly dependent. This check is important as Multicollinearity among independent variables will result in less statistical inferences.

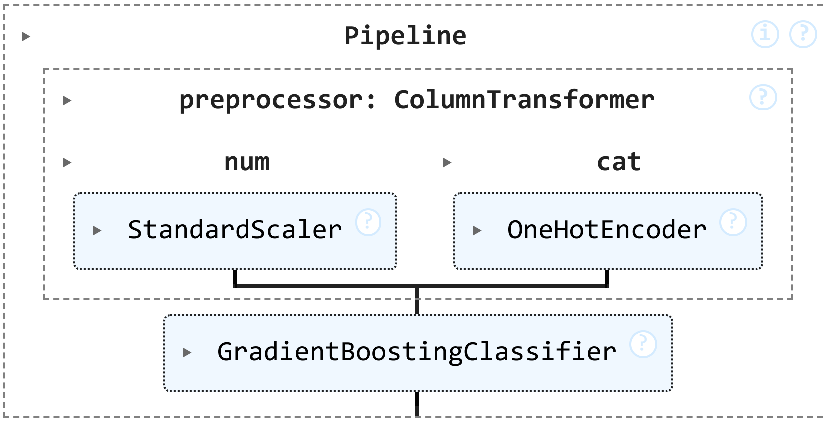
In [12]:
```
# import the relevant module for our predictive model
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [13]:
```
# Define feature columns and target column
X = df.drop('PurchaseStatus',axis=1)
y = df['PurchaseStatus']
# Identify numerical columns and categorical columns
cat_cols = ['Gender','ProductCategory','LoyaltyProgram','DiscountsAvailed','Purchas
num_cols = ['Age','AnnualIncome','NumberOfPurchases','TimeSpentOnWebsite']

# create a ColumnTransformer for preprocessing
```

```
preprocessor = ColumnTransformer(transformers=[('num',StandardScaler(),num_cols),
                                               ('cat',OneHotEncoder(),cat_cols[:-1])])
# create a pipeline that first preprocess the data then applies the model
pipeline = Pipeline([('preprocessor',preprocessor),
                     ('classifier',GradientBoostingClassifier(random_state=42))])
```

In [14]:
```
# Split the data into training and testing stes with train size of 75% of our data
X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 train_size = .75,
                                                 stratify = y,
                                                 random_state = 42)
# Train the model
pipeline.fit(X_train,y_train)
```

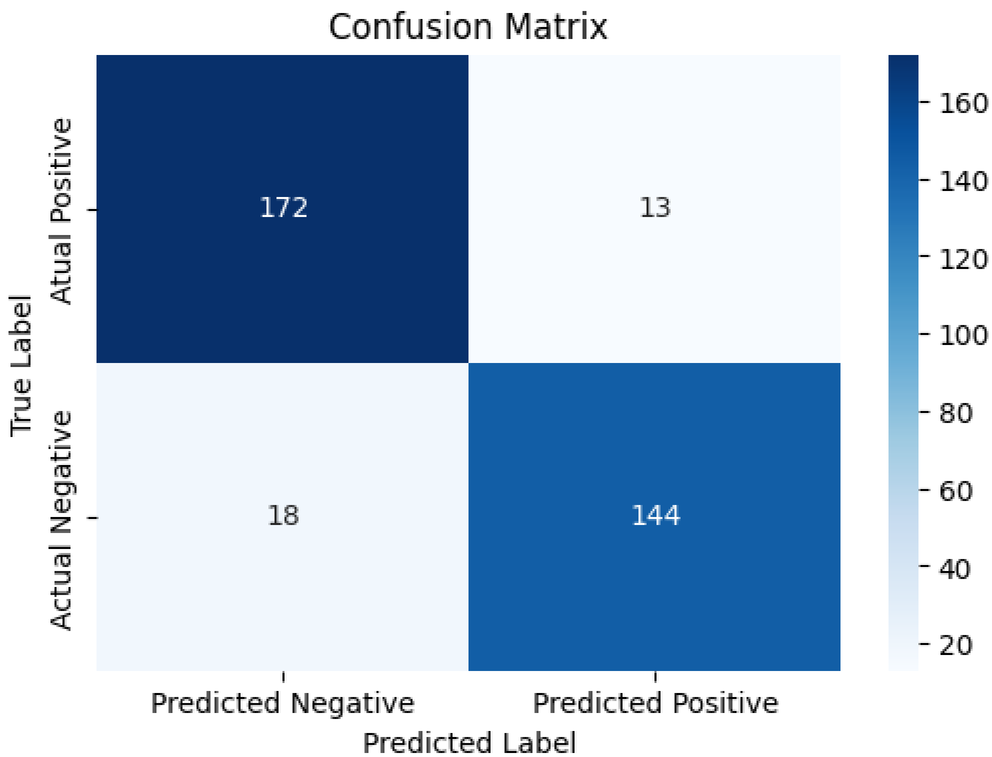
Out[14]:




In [15]:
```
# Make predictions
y_pred = pipeline.predict(X_test)
# Evaluate model performance on test set
accuracy = accuracy_score(y_test,y_pred)
class_report = classification_report(y_test,y_pred)
cm = confusion_matrix(y_test,y_pred)
```

In [16]:
```
# Output the results
print('Model Test Accuracy:',accuracy)
print('Classification Report:\n',class_report)

# Visualize the confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot = True,
            fmt='d',cmap='Blues',
            xticklabels=(['Predicted Negative','Predicted Positive']),
            yticklabels=(['Atual Positive','Actual Negative']))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
Model Test Accuracy: 0.9106628242074928
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       185
           1       0.92      0.89      0.90       162

    accuracy                           0.91       347
   macro avg       0.91      0.91      0.91       347
weighted avg       0.91      0.91      0.91       347
```



## PREDICTING CUSTOMER'S BEHAVIOUR

**Description:**

This dataset contains information on customer purchase behavior across various attributes, aiming to help understand the factors influencing purchase decisions. The dataset includes purchasing habits and other relevant features.

**Features**:

**Age**: Customer's age

**Gender**: Customer's gender (0: Male, 1: Female)

**Annual Income**: Annual income of the customer in dollars

**Number of Purchases**: Total number of purchases made by the customer

**Product Category**: Category of the purchased product (0: Electronics, 1: Clothing, 2: Home Goods, 3: Beauty, 4: Sports)

**Time Spent on Website**: Time spent by the customer on the website in minutes

**Loyalty Program**: Whether the customer is a member of the loyalty program (0: No, 1: Yes)

**Discounts Availed**: Number of discounts availed by the customer (range: 0-5)

**Purchase Status** (Target Variable): Likelihood of the customer making a purchase (0: No, 1: Yes)

## ANALYZING THE MODEL PERFORMANCE

**Model Test Accuracy**: The model correctly predicted 91% of the test samples. This is a relatively high accuracy, indicating that the model is performing well overall.

**Precision**: Measures the proportion of correct predictions among those predicted as a class. For class 0: 91% of instances predicted as 0 were actually 0. For class 1: 92% of instances predicted as 1 were actually 1.

**Recall**: Measures the proportion of correctly predicted instances among the actual instances of a class. For class 0: 93% of actual 0 instances were correctly predicted. For class 1: 89% of actual 1 instance were correctly predicted.

**F1-score**: The harmonic mean of precision and recall, balancing both metrics.

**Support**: The number of instances for each class.

**Overall Conclusion**: The model exhibits strong performance, achieving an overall accuracy of 91%. Both classes demonstrate high precision and recall, indicating that the model is able to accurately identify instances of both classes. The macro average and weighted average of precision, recall, and F1-score further reinforce the model's balanced performance across both classes.

**Key Takeaways**: The model is well-suited for this classification task.

## METHODOLOGY

**Tools & Libraries used for Analysis:**

**Pandas:** For Data manipulation and

**Numpy:** |For Numerical Operations

**Matplotlib & Seaborn:** For Graphical representation of the Dataset.

**Scikit-learn:** The GradientBoostingClassifier class of Scikit-learn which is an Ensemble Machine learning technic for model classification is employed.

**Reference material:** https://www.kaggle.com/datasets/rabieelkharoua/predict-customer-purchase-behavior-dataset