



Project Title	Customer Satisfaction Prediction
language	Machine learning, python, SQL, Excel
Tools	VS code, Jupyter notebook
Domain	Data Science
Project Difficulties level	Advance

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set](#)

About Dataset

The Customer Support Ticket Dataset is a dataset that includes customer support tickets for various tech products. It consists of customer inquiries related to hardware issues, software bugs, network problems, account access, data loss, and other support topics. The dataset provides information about the customer, the product purchased, the ticket type, the ticket channel, the ticket status, and other relevant details.

The dataset can be used for various analysis and modelling tasks in the customer service domain.

Features Description:

- Ticket ID: A unique identifier for each ticket.
- Customer Name: The name of the customer who raised the ticket.
- Customer Email: The email address of the customer (Domain name - @example.com is intentional for user data privacy concern).
- Customer Age: The age of the customer.
- Customer Gender: The gender of the customer.
- Product Purchased: The tech product purchased by the customer.
- Date of Purchase: The date when the product was purchased.
- Ticket Type: The type of ticket (e.g., technical issue, billing inquiry, product inquiry).
- Ticket Subject: The subject/topic of the ticket.
- Ticket Description: The description of the customer's issue or inquiry.
- Ticket Status: The status of the ticket (e.g., open, closed, pending customer response).
- Resolution: The resolution or solution provided for closed tickets.
- Ticket Priority: The priority level assigned to the ticket (e.g., low, medium, high, critical).
- Ticket Channel: The channel through which the ticket was raised (e.g., email, phone, chat, social media).
- First Response Time: The time taken to provide the first response to the customer.
- Time to Resolution: The time taken to resolve the ticket.
- Customer Satisfaction Rating: The customer's satisfaction rating for closed tickets (on a scale of 1 to 5).

Use Cases of such dataset:

- **Customer Support Analysis:** The dataset can be used to analyze customer support ticket trends, identify common issues, and improve support processes.
- **Natural Language Processing (NLP):** The ticket descriptions can be used for training NLP models to automate ticket categorization or sentiment analysis.
- **Customer Satisfaction Prediction:** The dataset can be used to train models to predict customer satisfaction based on ticket information.
- **Ticket Resolution Time Prediction:** The dataset can be used to build models for predicting the time it takes to resolve a ticket based on various factors.
- **Customer Segmentation:** The dataset can be used to segment customers based on their ticket types, issues, or satisfaction levels.
- **Recommender Systems:** The dataset can be used to build recommendation systems for suggesting relevant solutions or products based on customer inquiries.

Example: You can get the basic idea how you can create a project from here

Customer Satisfaction Prediction Machine Learning Project

Project Overview

The goal of this project is to predict customer satisfaction using historical data. This involves using machine learning algorithms to analyze factors that influence customer satisfaction and build a predictive model.

Dataset

A commonly used dataset for this type of project is the "Customer Satisfaction Survey" dataset, which includes features such as:

- **CustomerID**
- **Age**
- **Gender**
- **Income**
- **Education Level**
- **Product Purchased**
- **Purchase Frequency**
- **Customer Service Interactions**
- **Feedback Scores**
- **Overall Satisfaction**

This dataset can be found on platforms like Kaggle or UCI Machine Learning Repository.

Steps and Implementation

1. **Data Preprocessing**
2. **Exploratory Data Analysis (EDA)**
3. **Feature Engineering**

4. **Model Building**
5. **Model Evaluation**
6. **Visualization**

Implementation Code

Here is a sample implementation in Python:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('customer_satisfaction.csv')

# Display basic info about the dataset
print(data.info())

# Data Preprocessing
# Handling missing values
data = data.dropna()

# Encoding categorical variables
```

```
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])

# Define features and target variable
X = data.drop(['CustomerID', 'Overall Satisfaction'], axis=1)
y = data['Overall Satisfaction']

# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Building
# Train a Random Forest Classifier
rfc = RandomForestClassifier(random_state=42)
rfc.fit(X_train, y_train)

# Predict on the test set
y_pred = rfc.predict(X_test)

# Model Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Visualization of Results
# Feature Importance
feature_importances = pd.Series(rfc.feature_importances_, index=X.columns)
feature_importances.nlargest(10).plot(kind='barh')
plt.title('Top 10 Feature Importances')
plt.show()
```

Example: You can get the basic idea how you can create a project from here

Explanation of Code

1. Data Preprocessing:

- Load the dataset and display basic information.
- Handle missing values by dropping rows with NA values.
- Encode categorical variables using `LabelEncoder`.

2. Exploratory Data Analysis (EDA):

- Although not shown in the code snippet, EDA typically involves visualizing data distributions, correlations, and patterns using libraries like `matplotlib` and `seaborn`.

3. Feature Engineering:

- Define the feature set `X` and the target variable `y`.
- Split the data into training and testing sets using `train_test_split`.

4. Feature Scaling:

- Standardize the features using `StandardScaler` to ensure all features contribute equally to the model.

5. Model Building:

- Train a `RandomForestClassifier` on the training data.

- Predict customer satisfaction on the test data.

6. Model Evaluation:

- Evaluate the model using metrics like accuracy, classification report, and confusion matrix.
- Visualize the top 10 feature importances to understand which factors contribute most to customer satisfaction.

Additional Resources

- [Customer Satisfaction Survey Data on Kaggle](#)
- Random Forest Classifier Documentation
- Handling Missing Data in Pandas
- Feature Scaling with StandardScaler

This implementation provides a framework for predicting customer satisfaction using machine learning. You can extend it by experimenting with different algorithms, fine-tuning hyperparameters, and incorporating additional features to improve the model's performance.

Sample code with output

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```



```

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans

# Load the dataset
data =
pd.read_csv("/kaggle/input/customer-support-ticket-dataset/customer_support_tickets.csv")

# Display the first few rows of the dataset
print(data.head())

# Perform initial exploratory data analysis (EDA)
print(data.info())
print(data.describe())

```

	Ticket ID	Customer Name	Customer Email
Customer Age \			
0	1	Marisa Obrien	carrollallison@example.com
32			
1	2	Jessica Rios	clarkeashley@example.com
42			
2	3	Christopher Robbins	gonzalestracy@example.com
48			

3	4	Christina Dillon	bradleyolson@example.org
27			
4	5	Alexander Carroll	bradleymark@example.com
67			

Customer	Gender	Product	Purchased	Date of Purchase	Ticket Type \
0	Other	GoPro Hero	2021-03-22	Technical	issue
1	Female	LG Smart TV	2021-05-22	Technical	issue
2	Other	Dell XPS	2020-07-14	Technical	issue
3	Female	Microsoft Office	2020-11-13	Billing	inquiry
4	Female	Autodesk AutoCAD	2020-02-04	Billing	inquiry

Ticket Subject \
0 Product setup
1 Peripheral compatibility
2 Network problem
3 Account access
4 Data loss

Ticket Description \

- 0 I'm having an issue with the {product_purchase...
- 1 I'm having an issue with the {product_purchase...
- 2 I'm facing a problem with my {product_purchase...
- 3 I'm having an issue with the {product_purchase...
- 4 I'm having an issue with the {product_purchase...

Ticket Status

Resolution \

0 Pending Customer Response

NaN

1 Pending Customer Response

NaN

2 Closed Case maybe show recently my computer follow.

3 Closed Try capital clearly never color toward story.

4 Closed West decision evidence bit.

Ticket Priority Ticket Channel First Response Time Time to Resolution \

0 Critical Social media 2023-06-01 12:15:36

NaN

1 Critical Chat 2023-06-01 16:45:38

NaN

2 Low Social media 2023-06-01 11:14:38

2023-06-01 18:05:38

3 Low Social media 2023-06-01 07:29:40

2023-06-01 01:57:40

4 Low Email 2023-06-01 00:12:42

2023-06-01 19:53:42

Customer Satisfaction Rating

0 NaN

1 NaN

2 3.0

3 3.0

4 1.0

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8469 entries, 0 to 8468

Data columns (total 17 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	Ticket ID	8469 non-null	int64
1	Customer Name	8469 non-null	object
2	Customer Email	8469 non-null	object
3	Customer Age	8469 non-null	int64
4	Customer Gender	8469 non-null	object
5	Product Purchased	8469 non-null	object

6	Date of Purchase	8469 non-null	object
7	Ticket Type	8469 non-null	object
8	Ticket Subject	8469 non-null	object
9	Ticket Description	8469 non-null	object
10	Ticket Status	8469 non-null	object
11	Resolution	2769 non-null	object
12	Ticket Priority	8469 non-null	object
13	Ticket Channel	8469 non-null	object
14	First Response Time	5650 non-null	object
15	Time to Resolution	2769 non-null	object
16	Customer Satisfaction Rating	2769 non-null	float64

dtypes: float64(1), int64(2), object(14)

memory usage: 1.1+ MB

None

	Ticket ID	Customer Age	Customer Satisfaction Rating
count	8469.000000	8469.000000	2769.000000
mean	4235.000000	44.026804	2.991333
std	2444.934048	15.296112	1.407016
min	1.000000	18.000000	1.000000
25%	2118.000000	31.000000	2.000000
50%	4235.000000	44.000000	3.000000
75%	6352.000000	57.000000	4.000000
max	8469.000000	70.000000	5.000000

In [2]:

```
# Print column names
```

```
print(data.columns)
```

```
Index(['Ticket ID', 'Customer Name', 'Customer Email',  
      'Customer Age',  
      'Customer Gender', 'Product Purchased', 'Date of  
Purchase',  
      'Ticket Type', 'Ticket Subject', 'Ticket Description',  
      'Ticket Status',  
      'Resolution', 'Ticket Priority', 'Ticket Channel',  
      'First Response Time', 'Time to Resolution',  
      'Customer Satisfaction Rating'],  
      dtype='object')
```

In [3]:

```
#Analyze customer support ticket trends
```

```
# Identify common issues
```

```
common_issues = data['Ticket Subject'].value_counts().head(10)
```

```
print("Top 10 Common Issues:")
```

```
print(common_issues)
```

```
# Plotting ticket trends over time
```

```

data['Date of Purchase'] = pd.to_datetime(data['Date of
Purchase'])
data['YearMonth'] = data['Date of Purchase'].dt.to_period('M')
ticket_trends = data.groupby('YearMonth').size()

plt.figure(figsize=(10, 6))
ticket_trends.plot(kind='line', marker='o')
plt.title('Customer Support Ticket Trends Over Time')
plt.xlabel('Year-Month')
plt.ylabel('Number of Tickets')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

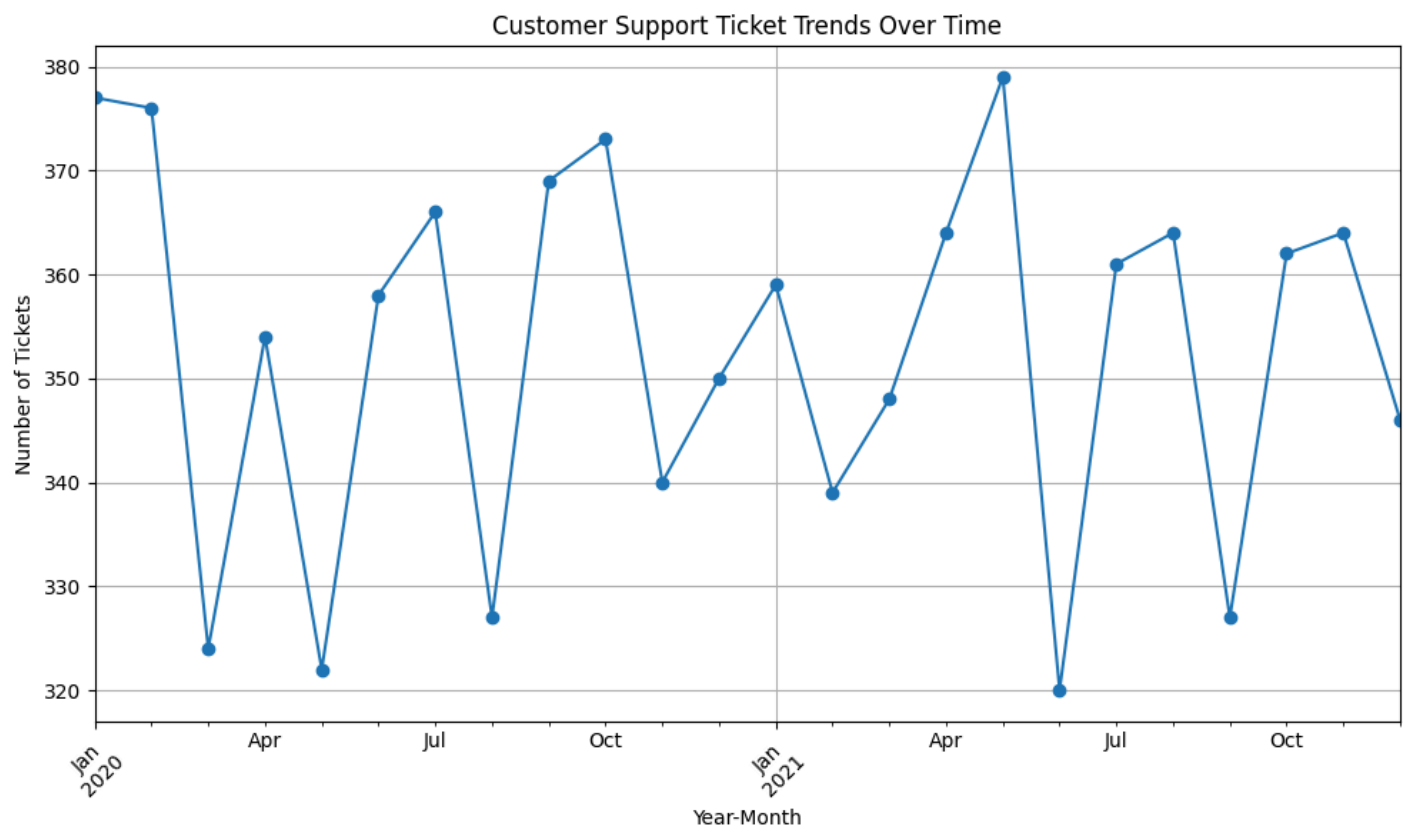
```

Top 10 Common Issues:

Ticket Subject

Refund request	576
Software bug	574
Product compatibility	567
Delivery problem	561
Hardware issue	547
Battery life	542
Network problem	539

```
Installation support      530
Product setup             529
Payment issue             526
Name: count, dtype: int64
```



```
In [4]:
```

```
# Segment customers
```

```
# Segment based on ticket types
```

```
ticket_type_segmentation = data.groupby('Ticket Type').size()
```

```
print("\nSegmentation based on Ticket Types:")
```

```
print(ticket_type_segmentation)
```



```
# Segment based on satisfaction levels
satisfaction_segmentation = data.groupby('Customer Satisfaction
Rating').size()
print("\nSegmentation based on Customer Satisfaction Levels:")
print(satisfaction_segmentation)
```

Segmentation based on Ticket Types:

Ticket Type

Billing inquiry	1634
Cancellation request	1695
Product inquiry	1641
Refund request	1752
Technical issue	1747

dtype: int64

Segmentation based on Customer Satisfaction Levels:

Customer Satisfaction Rating

1.0	553
2.0	549
3.0	580
4.0	543
5.0	544

dtype: int64

In [5]:

```
# Set up the plotting aesthetics
```

```
sns.set(style="whitegrid")
```

```
#Customer Satisfaction Distribution
```

```
plt.figure(figsize=(10, 6))
```

```
sns.histplot(data['Customer Satisfaction Rating'], bins=5,  
kde=True, color='skyblue')
```

```
plt.title('Customer Satisfaction Distribution')
```

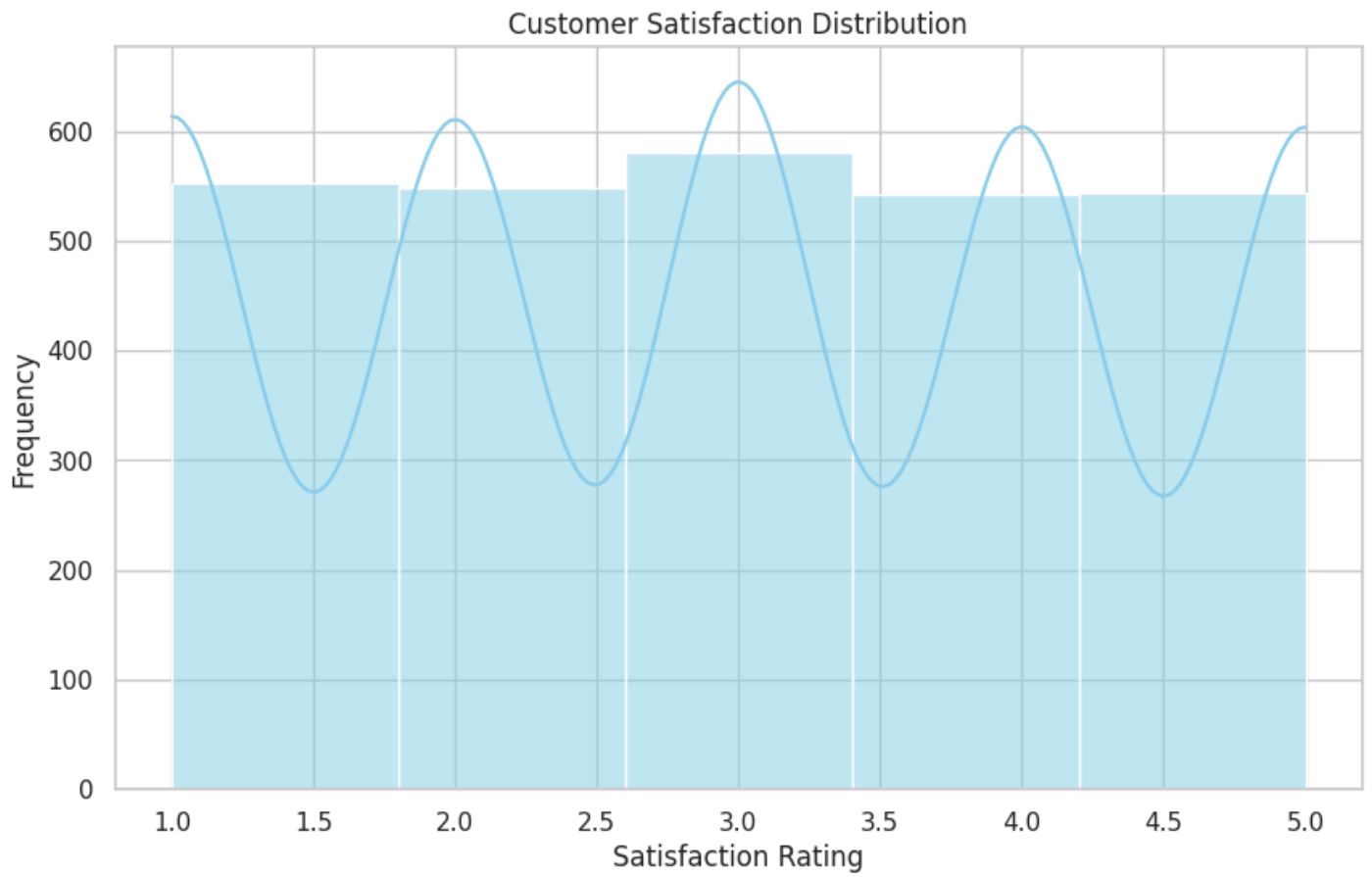
```
plt.xlabel('Satisfaction Rating')
```

```
plt.ylabel('Frequency')
```

```
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111  
9: FutureWarning: use_inf_as_na option is deprecated and will  
be removed in a future version. Convert inf values to NaN  
before operating instead.
```

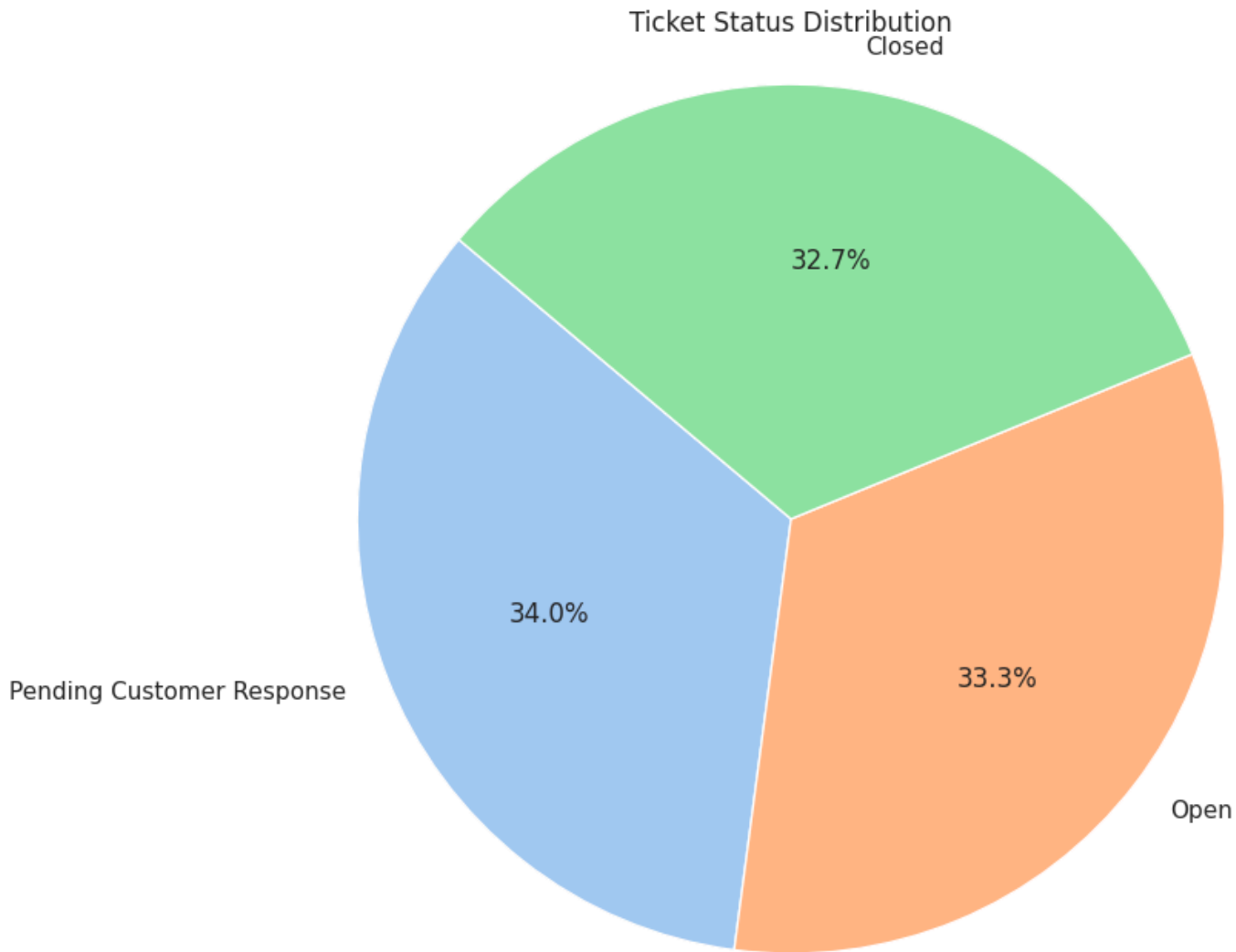
```
with pd.option_context('mode.use_inf_as_na', True):
```



In [6]:

#Ticket Status Distribution

```
ticket_status_distribution = data['Ticket  
Status'].value_counts()  
plt.figure(figsize=(8, 8))  
plt.pie(ticket_status_distribution,  
labels=ticket_status_distribution.index, autopct='%1.1f%%',  
colors=sns.color_palette('pastel'), startangle=140)  
plt.title('Ticket Status Distribution')  
plt.axis('equal')  
plt.show()
```



In [7]:

```
#Customer Age Distribution
```

```
plt.figure(figsize=(10, 6))
```

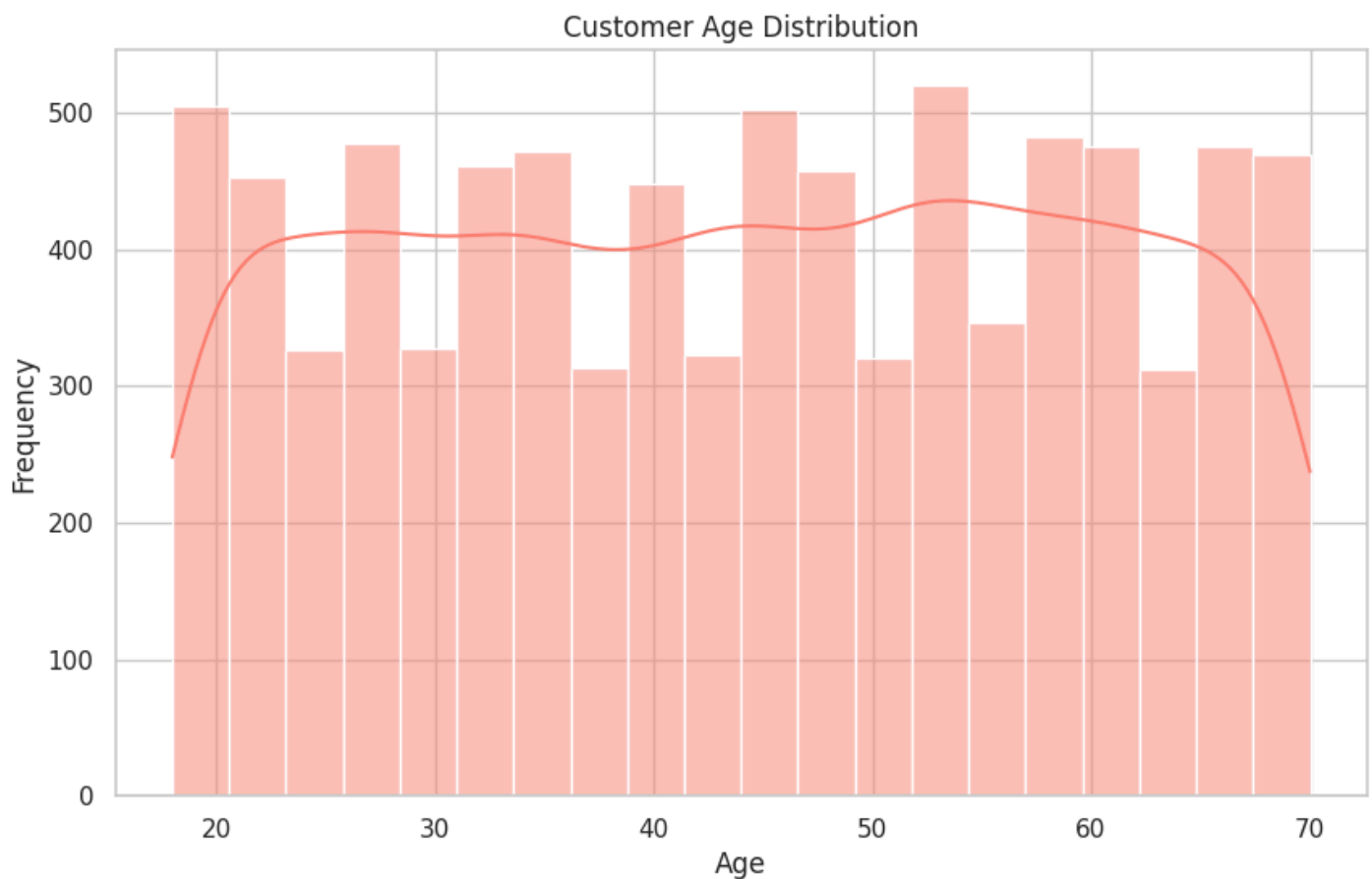
```
sns.histplot(data['Customer Age'], bins=20, kde=True,  
color='salmon')
```

```
plt.title('Customer Age Distribution')
```

```
plt.xlabel('Age')
```

```
plt.ylabel('Frequency')  
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111  
9: FutureWarning: use_inf_as_na option is deprecated and will  
be removed in a future version. Convert inf values to NaN  
before operating instead.  
with pd.option_context('mode.use_inf_as_na', True):
```



In [8]:

```
#Customer Gender Distribution
```

```
customer_gender_distribution = data['Customer  
Gender'].value_counts()
```

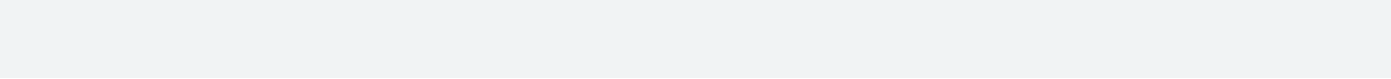
```
plt.figure(figsize=(8, 8))
```

```
plt.pie(customer_gender_distribution,  
labels=customer_gender_distribution.index, autopct='%1.1f%%',  
colors=sns.color_palette('Set2'), startangle=90)
```

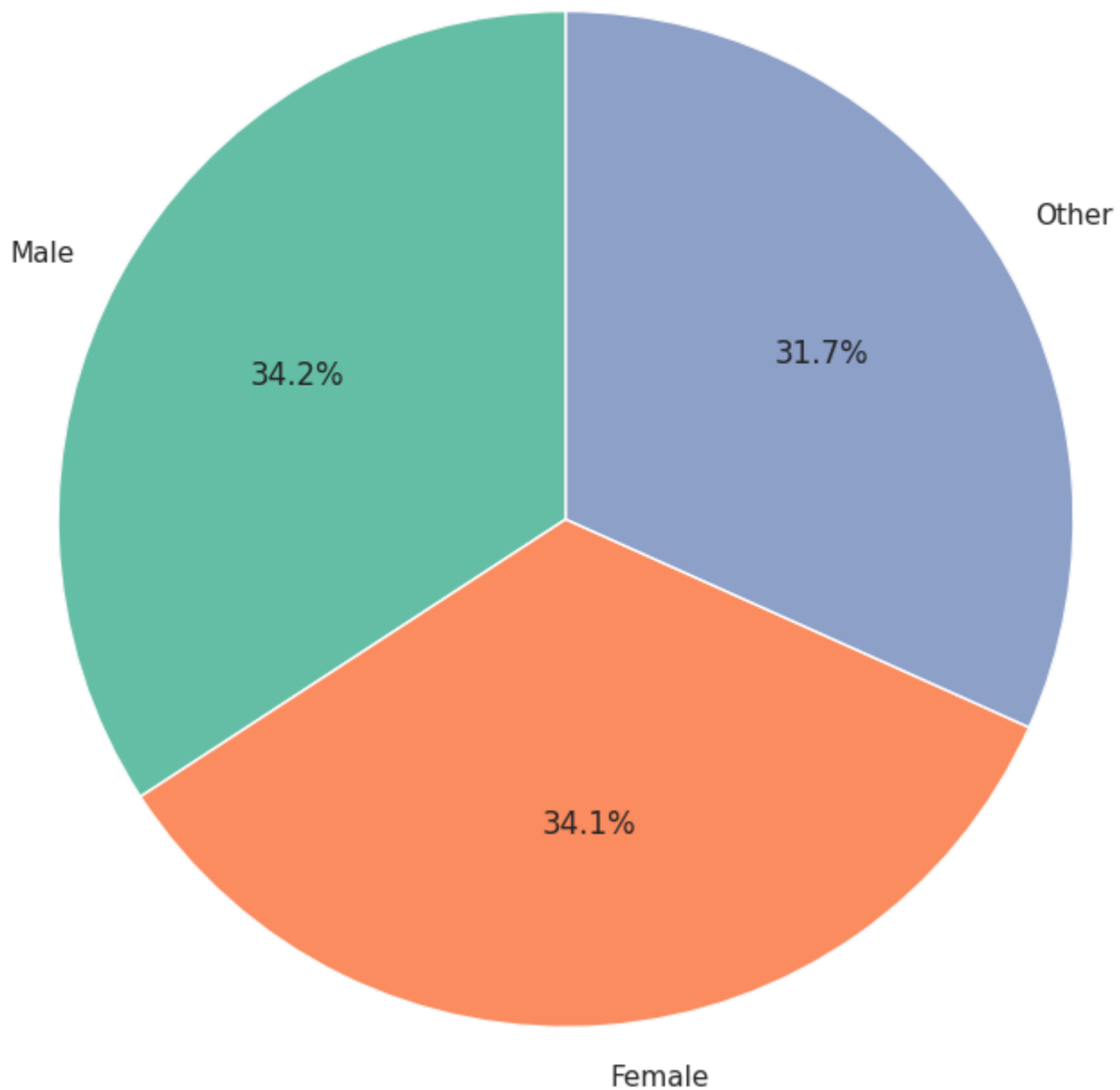
```
plt.title('Customer Gender Distribution')
```

```
plt.axis('equal')
```

```
plt.show()
```



Customer Gender Distribution



In [9]:

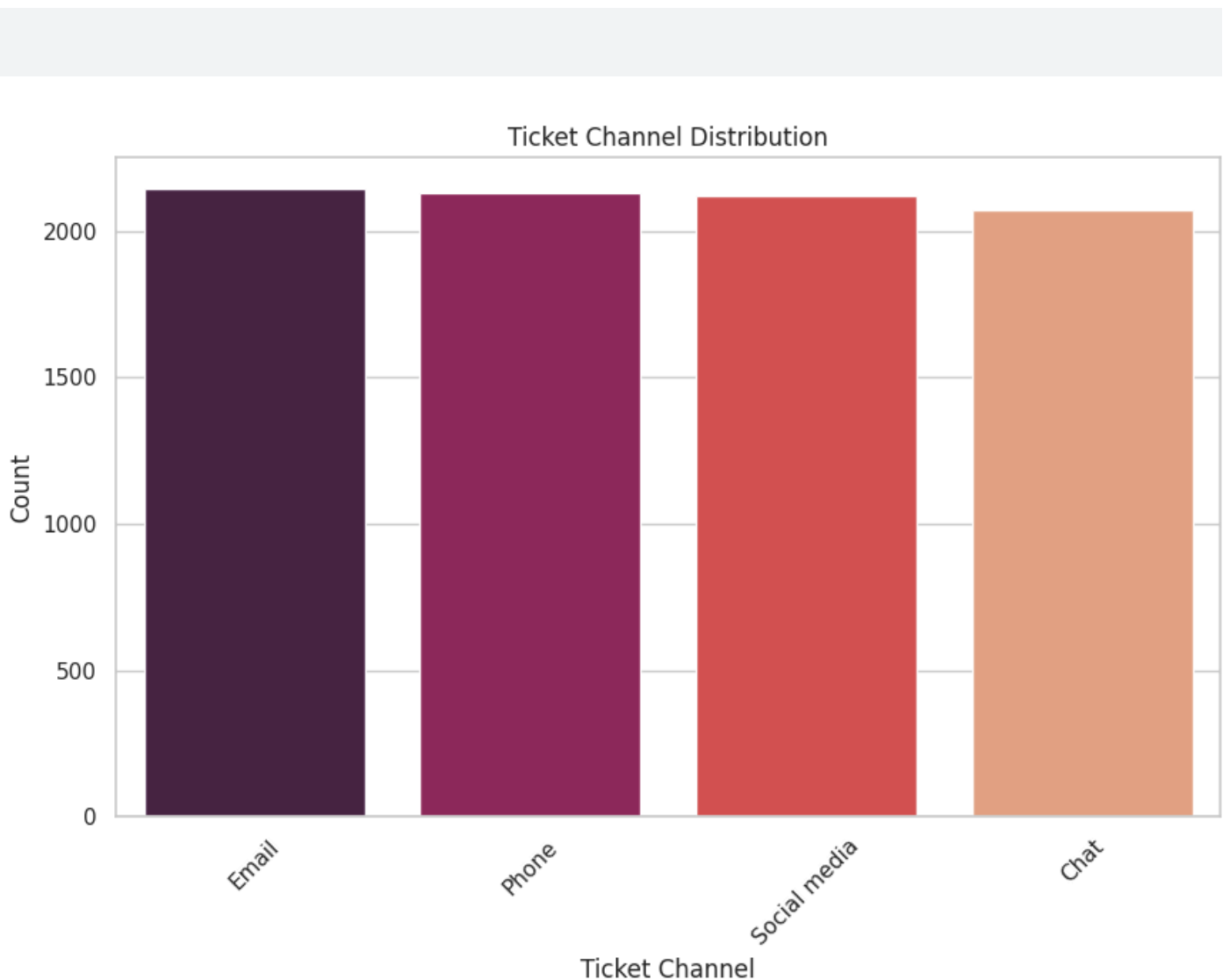
```
#Ticket Channel Distribution
```

```
plt.figure(figsize=(10, 6))
```

```
ticket_channel_distribution = data['Ticket  
Channel'].value_counts()
```

```
sns.barplot(x=ticket_channel_distribution.index,
```

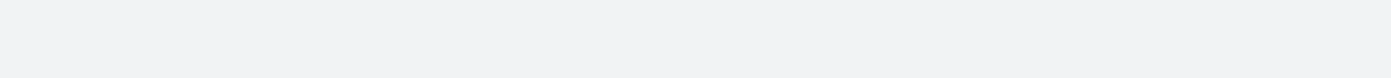
```
y=ticket_channel_distribution, palette='rocket')
plt.title('Ticket Channel Distribution')
plt.xlabel('Ticket Channel')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

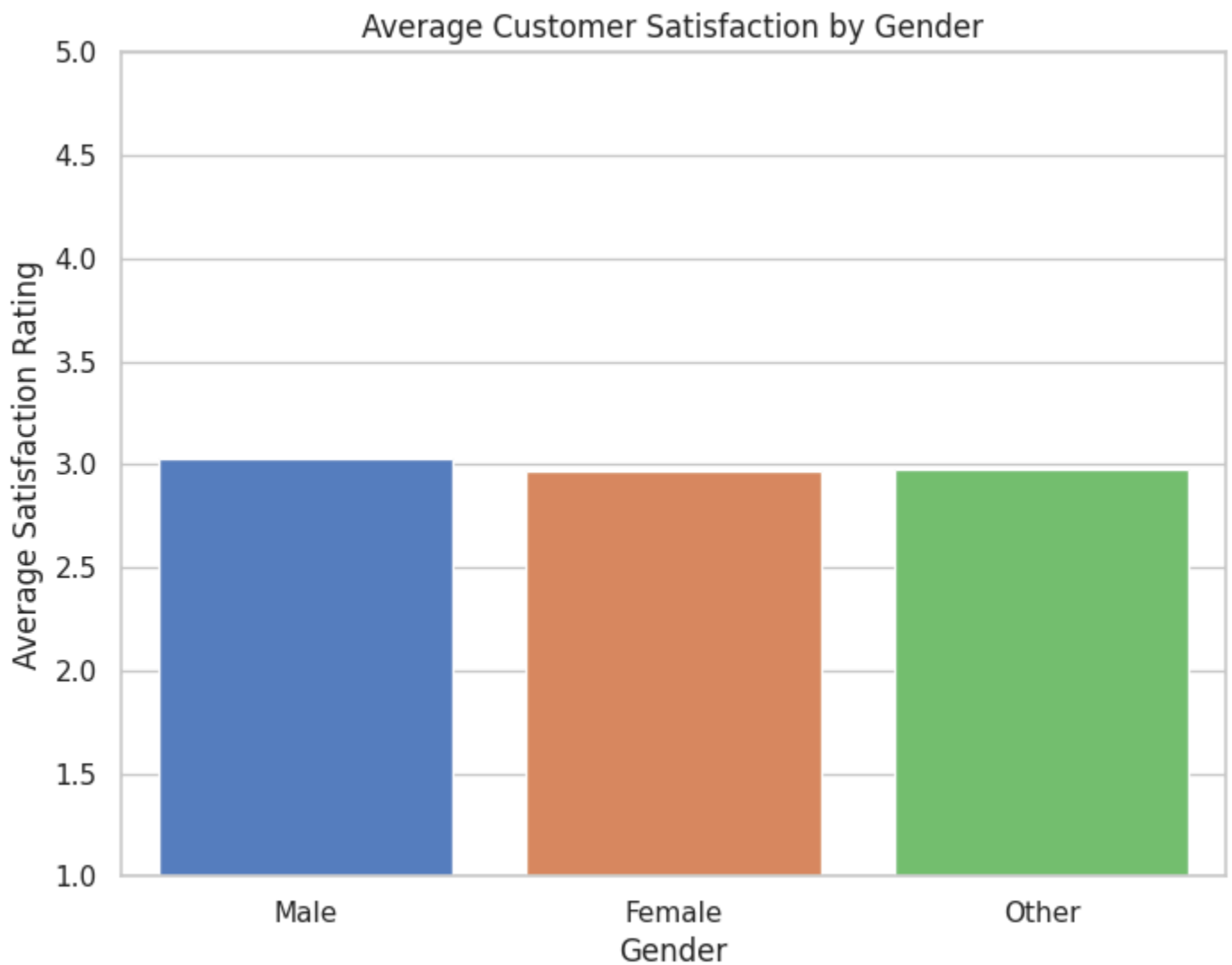


In [10]:

Chart 1: Average Customer Satisfaction by Gender (Bar Plot)


```
average_satisfaction = data.groupby('Customer  
Gender')['Customer Satisfaction Rating'].mean().reset_index()  
  
plt.figure(figsize=(8, 6))  
sns.barplot(x='Customer Gender', y='Customer Satisfaction  
Rating', data=average_satisfaction, palette='muted',  
order=['Male', 'Female', 'Other'])  
plt.title('Average Customer Satisfaction by Gender')  
plt.xlabel('Gender')  
plt.ylabel('Average Satisfaction Rating')  
plt.ylim(1, 5) # Adjust y-axis limit if needed  
plt.show()
```





In [11]:

```
#Product Purchased Distribution
```

```
plt.figure(figsize=(10, 6))
```

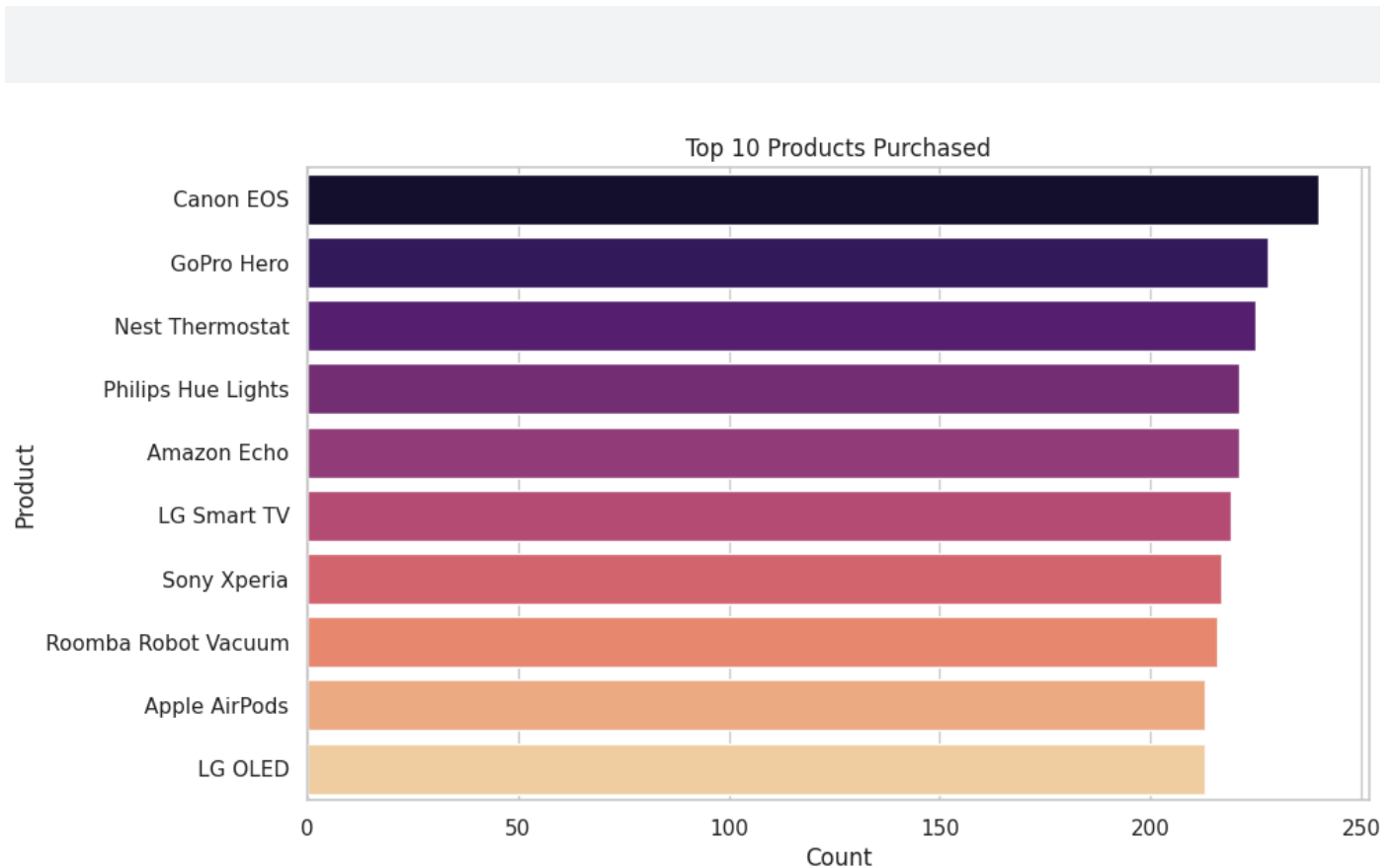
```
product_purchased_distribution = data['Product  
Purchased'].value_counts().head(10)
```

```
sns.barplot(y=product_purchased_distribution.index,  
x=product_purchased_distribution, palette='magma')
```

```
plt.title('Top 10 Products Purchased')
```

```
plt.xlabel('Count')
```

```
plt.ylabel('Product')
plt.show()
```



In [12]:

```
# Chart 2: Top Items Purchased by Gender (Horizontal Bar Chart)
```

```
plt.figure(figsize=(15, 6))
```

```
# Top Items Purchased by Males
```

```
plt.subplot(1, 3, 1)
```

```
top_items_male = data[data['Customer Gender'] ==  
'Male']['Product Purchased'].value_counts().head(5)
```

```
top_items_male.plot(kind='barh', color='skyblue')
```

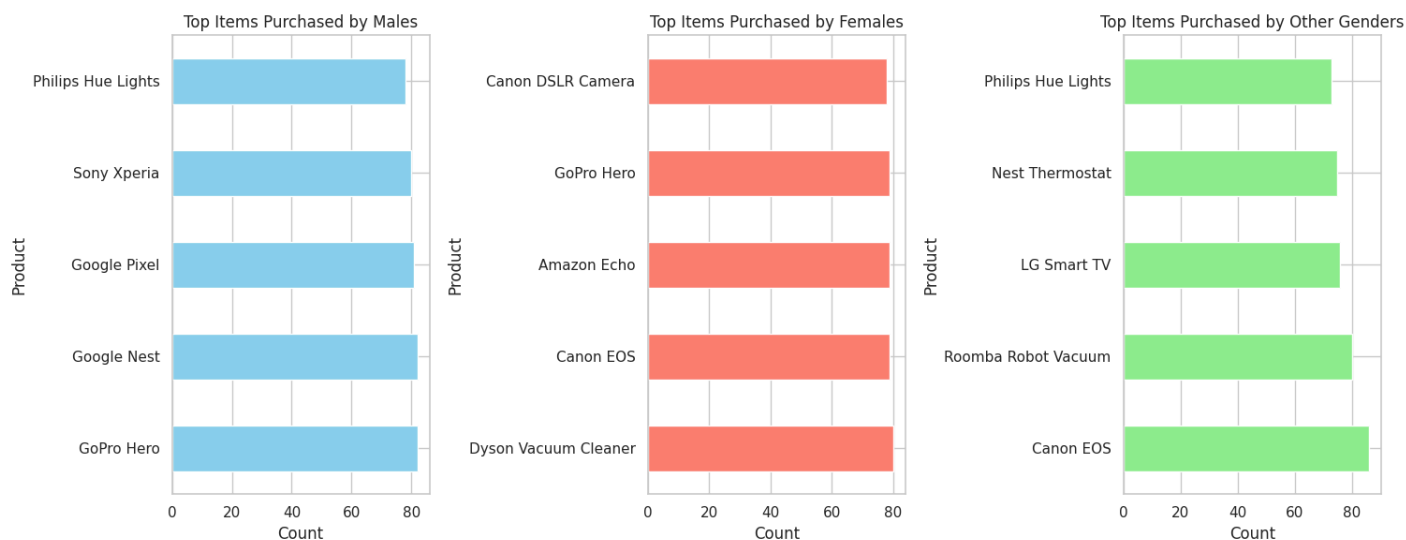
```
plt.title('Top Items Purchased by Males')
```

```
plt.xlabel('Count')
plt.ylabel('Product')

# Top Items Purchased by Females
plt.subplot(1, 3, 2)
top_items_female = data[data['Customer Gender'] ==
'Female']['Product Purchased'].value_counts().head(5)
top_items_female.plot(kind='barh', color='salmon')
plt.title('Top Items Purchased by Females')
plt.xlabel('Count')
plt.ylabel('Product')

# Top Items Purchased by Other Gender
plt.subplot(1, 3, 3)
top_items_other = data[data['Customer Gender'] ==
'Other']['Product Purchased'].value_counts().head(5)
top_items_other.plot(kind='barh', color='lightgreen')
plt.title('Top Items Purchased by Other Genders')
plt.xlabel('Count')
plt.ylabel('Product')

plt.tight_layout()
plt.show()
```



In [13]:

```
# Count ticket types
```

```
ticket_type_distribution = data['Ticket Type'].value_counts()
```

```
# Plot
```

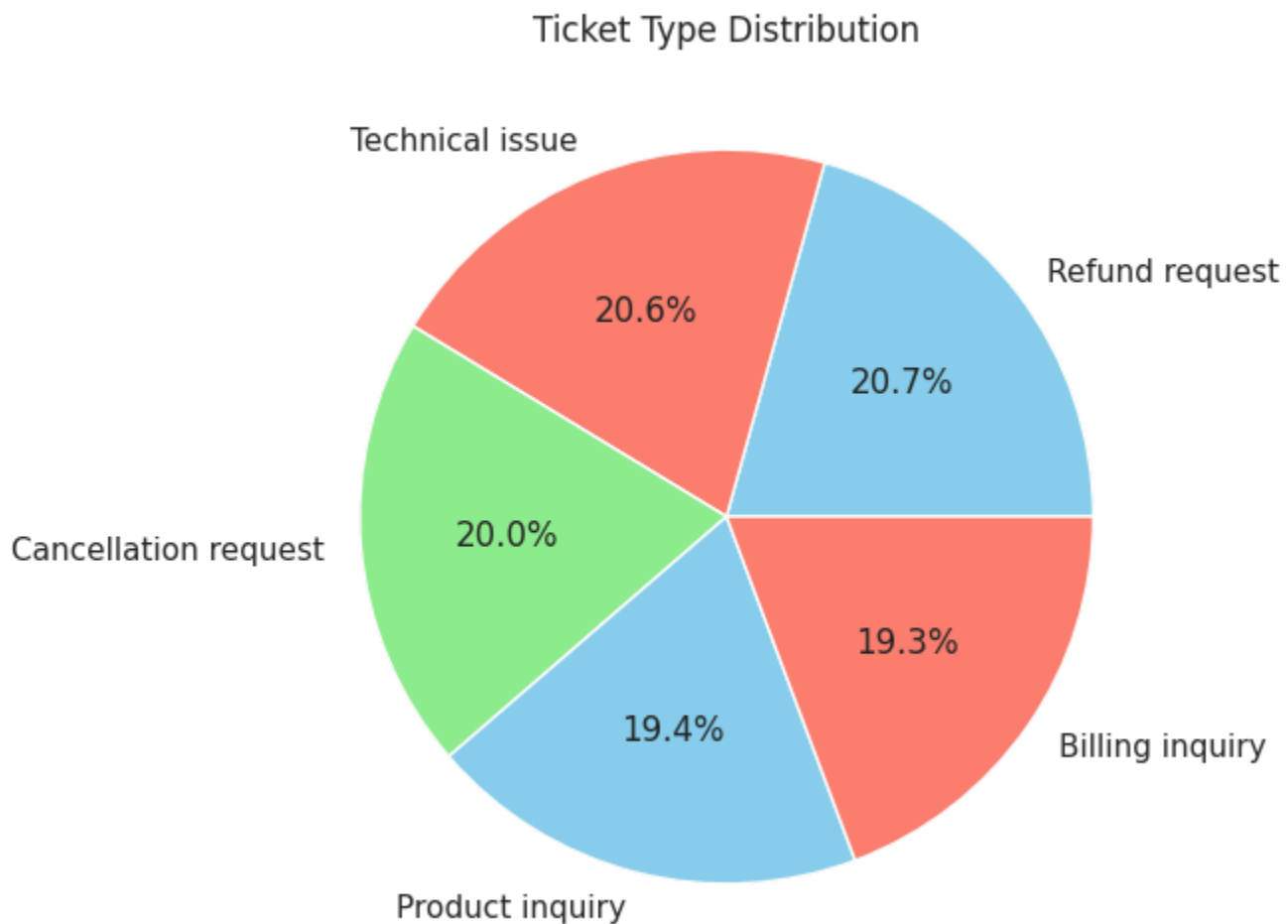
```
plt.figure(figsize=(8, 6))
```

```
ticket_type_distribution.plot(kind='pie', autopct='%1.1f%%',
colors=['skyblue', 'salmon', 'lightgreen'])
```

```
plt.title('Ticket Type Distribution')
```

```
plt.ylabel('')
```

```
plt.show()
```



In [14]:

```
# Count ticket priorities
```

```
priority_distribution = data['Ticket Priority'].value_counts()
```

```
# Plot
```

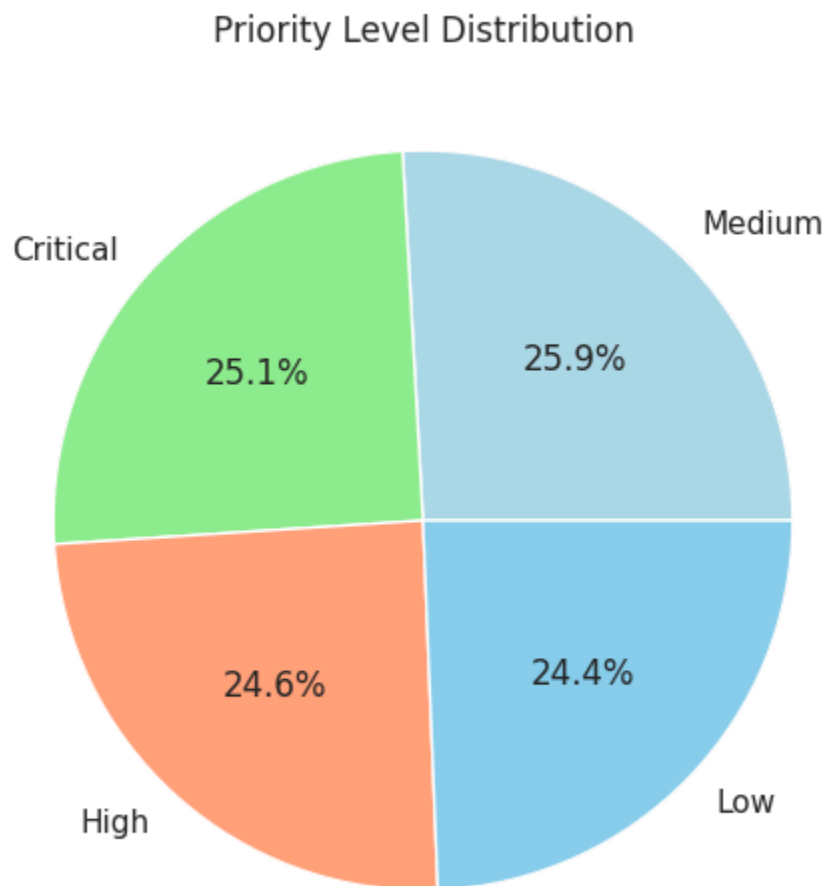
```
plt.figure(figsize=(8, 6))
```

```
priority_distribution.plot(kind='pie', autopct='%1.1f%%',  
colors=['lightblue', 'lightgreen', 'lightsalmon', 'skyblue'])
```

```
plt.title('Priority Level Distribution')
```

```
plt.ylabel('')
```

```
plt.show()
```



In [15]:

```
# Define age groups
```

```
bins = [0, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '61-70',  
'71-80', '81-90', '91-100']
```

```
# Categorize customers into age groups
```

```
data['Age Group'] = pd.cut(data['Customer Age'], bins=bins,  
labels=labels, right=False)
```

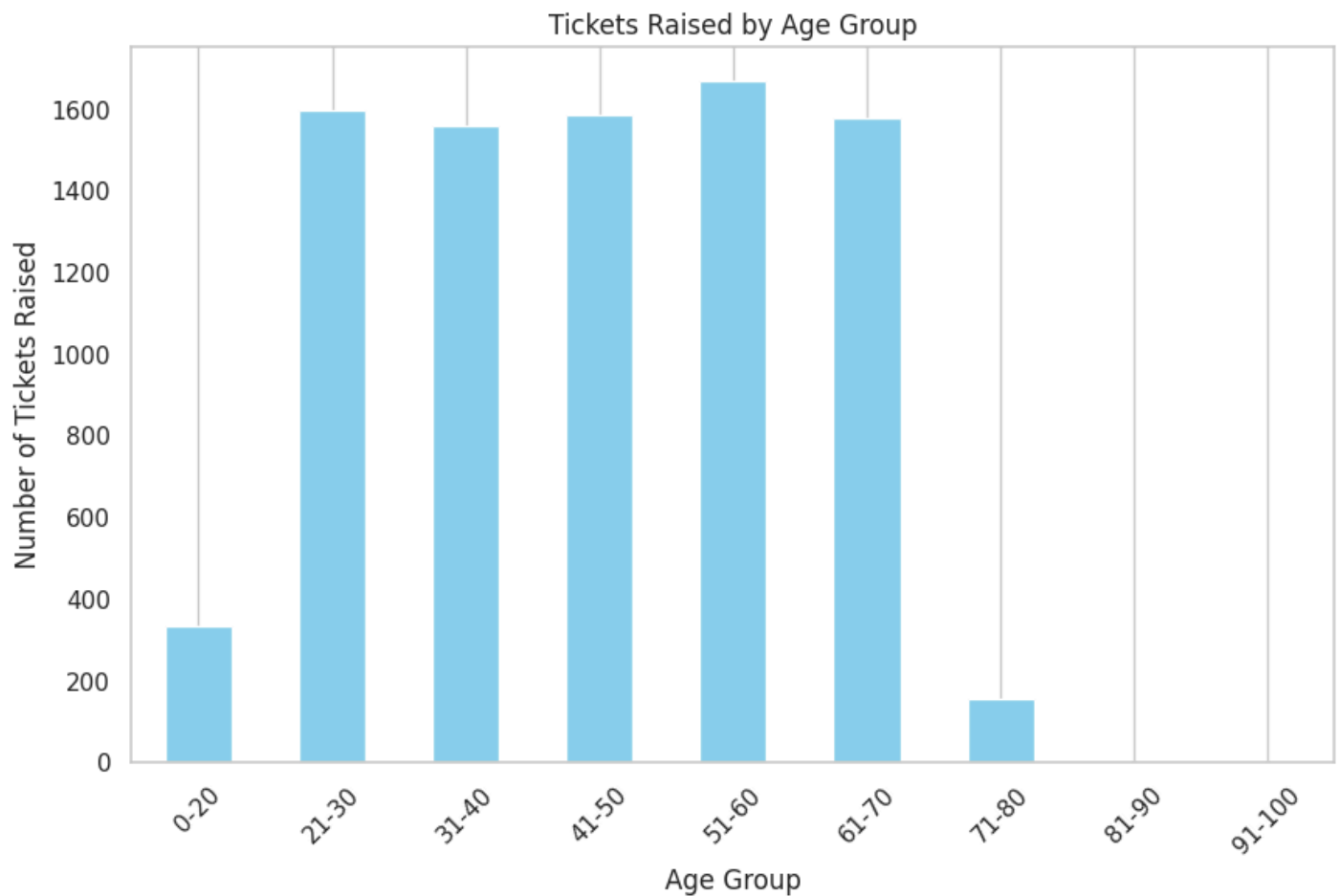
```
# Calculate number of tickets raised by each age group  
tickets_by_age_group = data.groupby('Age Group').size()
```

```
# Plot
```

```
plt.figure(figsize=(10, 6))  
tickets_by_age_group.plot(kind='bar', color='skyblue')  
plt.title('Tickets Raised by Age Group')  
plt.xlabel('Age Group')  
plt.ylabel('Number of Tickets Raised')  
plt.xticks(rotation=45)  
plt.grid(axis='y')  
plt.show()
```

/tmp/ipykernel_18/91670186.py:9: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.

```
    tickets_by_age_group = data.groupby('Age Group').size()
```

In [16]:

linkcode

Replace inf values with NaN

```
data.replace([np.inf, -np.inf], np.nan, inplace=True)
```

Create a facet grid for each ticket type

```
g = sns.FacetGrid(data, col='Ticket Type', col_wrap=3,  
height=5, aspect=1.5)
```

```
g.map(sns.histplot, 'Customer Age', bins=20, kde=True)
```

```
# Set titles and labels
g.set_titles('{col_name}')
g.set_axis_labels('Age', 'Number of Tickets')

# Adjust layout
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Distribution of Ticket Types by Age')

# Show plot
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
before operating instead.
    with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
```

before operating instead.

```
with pd.option_context('mode.use_inf_as_na', True):  
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111  
9: FutureWarning: use_inf_as_na option is deprecated and will  
be removed in a future version. Convert inf values to NaN  
before operating instead.
```

```
with pd.option_context('mode.use_inf_as_na', True):  
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111  
9: FutureWarning: use_inf_as_na option is deprecated and will  
be removed in a future version. Convert inf values to NaN  
before operating instead.
```

```
with pd.option_context('mode.use_inf_as_na', True):
```

