In [1]:
```
import os
os.getcwd()
```

Out[1]:
```
'C:\\Users\\guy74\\Documents\\NU Stuff\\ANA500\\ANA500_MicroProject'
```

In [3]:
```
import pandas as pd
import numpy as np

#Load in the data
df = pd.read_csv("airline.csv")

#There is a column which is presumably actually not named in the source but at some
#That variable is presumably not useful here analytically, and same goes for the "i
#The rest could serve useful and so we don't want to drop anything else yet at this
df = df.drop(columns=["Unnamed: 0", "id"])
#print(df)

#Loop through each column and calculate some data integrity metrics for each one to
column_checks = []
total_rows = df.shape[0]
for col in df.columns:
    col_data = df[col]

    populated_count = col_data.notnull().sum()
    missing_count = col_data.isnull().sum()
    distinct_count = col_data.nunique(dropna=True)
    data_type = col_data.dtype

    column_checks.append({
        "Column": col,
        "Populated Values": populated_count,
        "Missing Values": missing_count,
        "Distinct Values": distinct_count,
        "Data Type": str(data_type)
    })

integrity_df = pd.DataFrame(column_checks)

#This can sort it by missing values if we want the highest of those to rise to the
#integrity_df = integrity_df.sort_values(by="Missing Values", ascending=False)


print(integrity_df.to_string(index=False))
```

	Column	Populated Values	Missing Values	Distinct Values	Data Type
	Gender	129880	0	2	object
	Customer Type	129880	0	2	object
	Age	129880	0	75	int64
	Type of Travel	129880	0	2	object
	Class	129880	0	3	object
	Flight Distance	129880	0	3821	int64
	Inflight wifi service	129880	0	6	int64
	Departure/Arrival time convenient	129880	0	6	int64
	Ease of Online booking	129880	0	6	int64
	Gate location	129880	0	6	int64
	Food and drink	129880	0	6	int64
	Online boarding	129880	0	6	int64
	Seat comfort	129880	0	6	int64
	Inflight entertainment	129880	0	6	int64
	On-board service	129880	0	6	int64
	Leg room service	129880	0	6	int64
	Baggage handling	129880	0	5	int64
	Checkin service	129880	0	6	int64
	Inflight service	129880	0	6	int64
	Cleanliness	129880	0	6	int64
	Departure Delay in Minutes	129880	0	466	int64
	Arrival Delay in Minutes	129487	393	472	float64
	satisfaction	129880	0	2	object

In [4]:
```
#Let's also check for any duplicate rows where every single variable is exactly the
duplicate_count = df.duplicated().sum()
print(f"Total number of exact duplicate rows (excluding the two dropped columns): {
```

```
Total number of exact duplicate rows (excluding the two dropped columns): 0
```

In [7]:
```
#Swap out the column names so if any columns have spaces we change them to undersco
# Replace all spaces in column names with underscores
```

```
df.columns = df.columns.str.replace(" ", "_")
print(df)
```

```
        Gender      Customer_Type  Age   Type_of_Travel     Class  \
0         Male     Loyal Customer   13  Personal Travel  Eco Plus
1         Male  disloyal Customer   25  Business travel  Business
2       Female     Loyal Customer   26  Business travel  Business
3       Female     Loyal Customer   25  Business travel  Business
4         Male     Loyal Customer   61  Business travel  Business
...        ...                ...  ...              ...       ...
129875    Male  disloyal Customer   34  Business travel  Business
129876    Male     Loyal Customer   23  Business travel  Business
129877  Female     Loyal Customer   17  Personal Travel       Eco
129878    Male     Loyal Customer   14  Business travel  Business
129879  Female     Loyal Customer   42  Personal Travel       Eco

        Flight_Distance  Inflight_wifi_service  \
0                   460                      3
1                   235                      3
2                  1142                      2
3                   562                      2
4                   214                      3
...                 ...                    ...
129875              526                      3
129876              646                      4
129877              828                      2
129878             1127                      3
129879              264                      2

        Departure/Arrival_time_convenient  Ease_of_Online_booking  \
0                                       4                       3
1                                       2                       3
2                                       2                       2
3                                       5                       5
4                                       3                       3
...                                   ...                     ...
129875                                  3                       3
129876                                  4                       4
129877                                  5                       1
129878                                  3                       3
129879                                  5                       2

        Gate_location  ...  Inflight_entertainment  On-board_service  \
0                   1  ...                       5                 4
1                   3  ...                       1                 1
2                   2  ...                       5                 4
3                   5  ...                       2                 2
4                   3  ...                       3                 3
...               ...  ...                     ...               ...
129875              1  ...                       4                 3
129876              4  ...                       4                 4
129877              5  ...                       2                 4
129878              3  ...                       4                 3
129879              5  ...                       1                 1

        Leg_room_service  Baggage_handling  Checkin_service  Inflight_service  \
0                      3                 4                4                 5
1                      5                 3                1                 4
2                      3                 4                4                 4
```

```
3                    5                  3                  1                  4
4                    4                  4                  3                  3
...                ...                ...                ...                ...
129875               2                  4                  4                  5
129876               5                  5                  5                  5
129877               3                  4                  5                  4
129878               2                  5                  4                  5
129879               2                  1                  1                  1

        Cleanliness  Departure_Delay_in_Minutes  Arrival_Delay_in_Minutes  \
0                 5                         25                      18.0
1                 1                          1                       6.0
2                 5                          0                       0.0
3                 2                         11                       9.0
4                 3                          0                       0.0
...             ...                        ...                       ...
129875            4                          0                       0.0
129876            4                          0                       0.0
129877            2                          0                       0.0
129878            4                          0                       0.0
129879            1                          0                       0.0

                  satisfaction
0       neutral or dissatisfied
1       neutral or dissatisfied
2                     satisfied
3       neutral or dissatisfied
4                     satisfied
...                         ...
129875  neutral or dissatisfied
129876                satisfied
129877  neutral or dissatisfied
129878                satisfied
129879  neutral or dissatisfied

[129880 rows x 23 columns]
```

In [9]:
```
#If/when needed, export the new version of the file to another named .csv for furth
df.to_csv('AirlineSatisfaction_Transformed.csv', index=False)
```

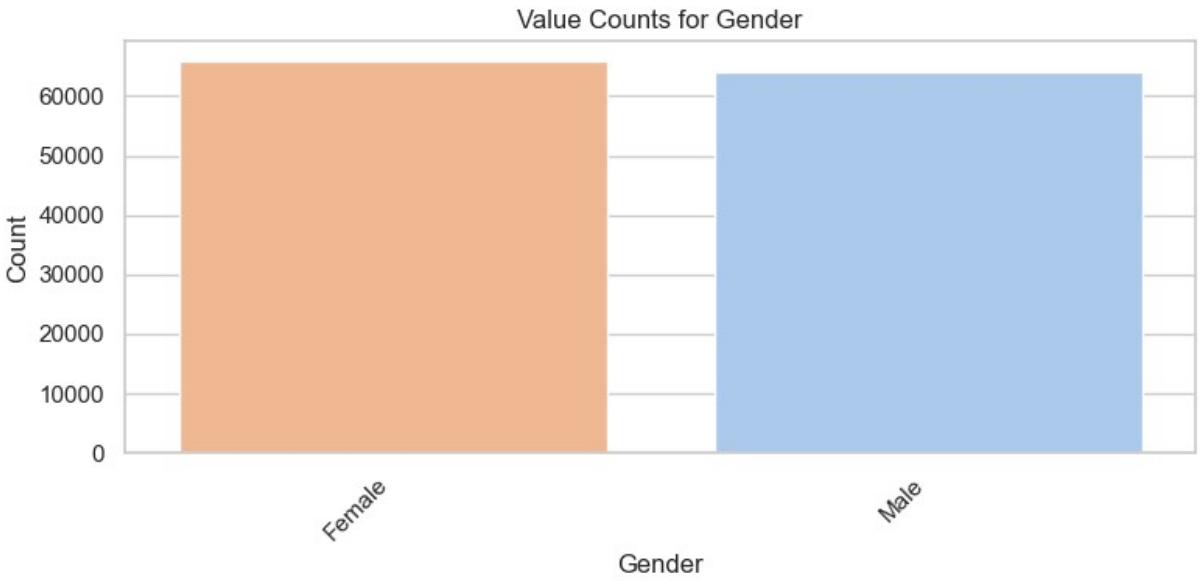
In [11]:
```
#This marks the end of week 1 and now continuing on for week 2
```

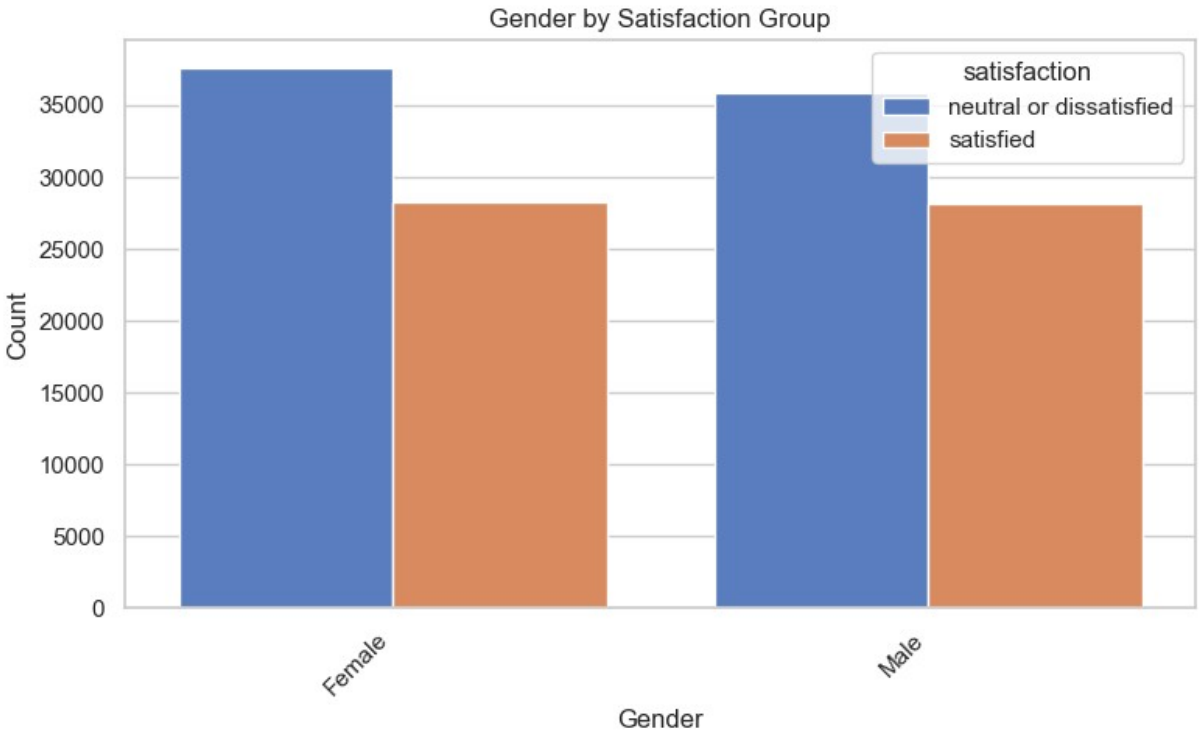
In [13]:
```
#Feeback from last week indicated:
#To strengthen your analysis, please dive deeper into your columns.
#For example, explore how the Age variable is distributed overall and within satisf
#How many records are labeled as "satisfied" versus "unsatisfied"? What does the ag
#These insights can help guide your feature engineering and modeling steps.

#Therefore below going to do a bit further exploratory analysis for distribution ov
#(for each column present in the dataset via a loop)
```

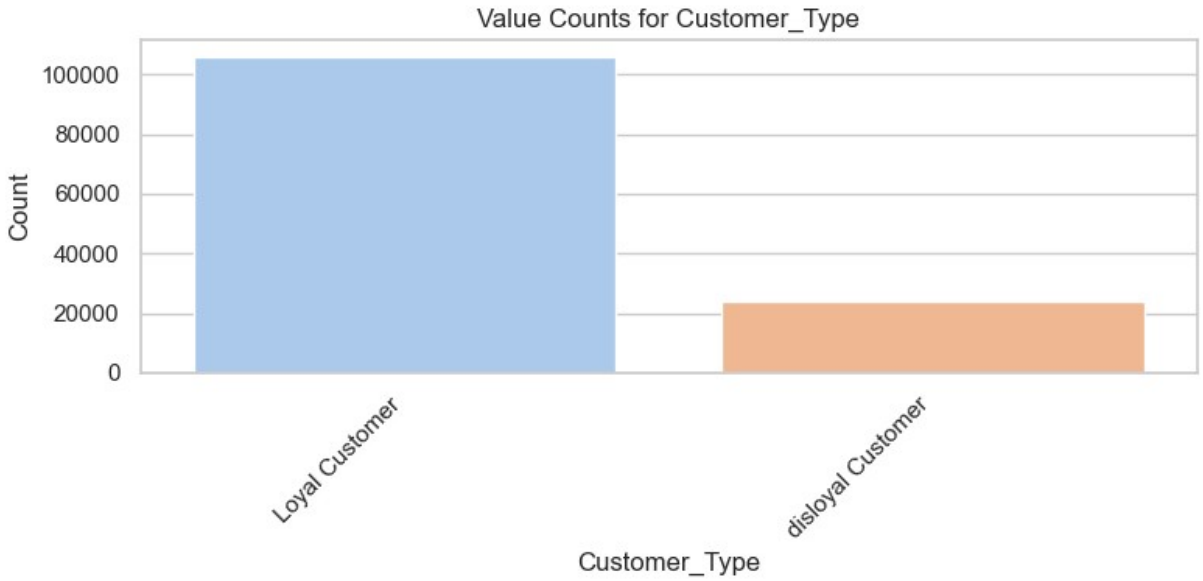
In [56]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

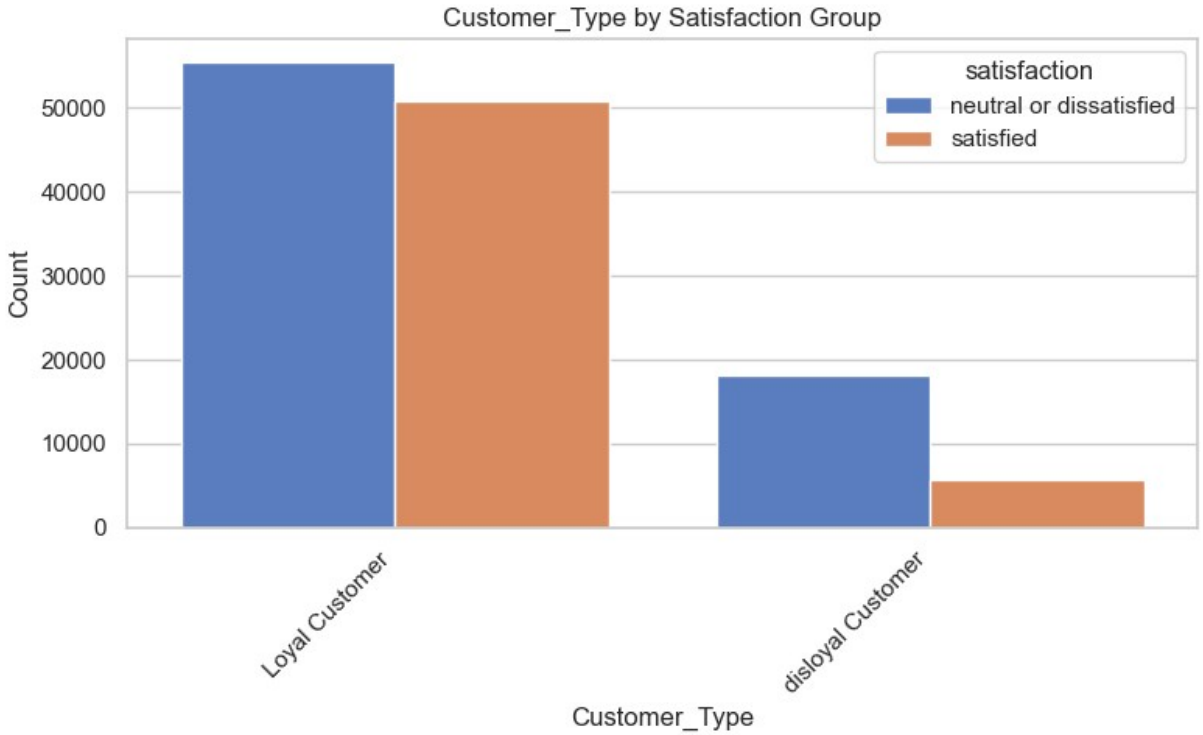
```
df = pd.read_csv("AirlineSatisfaction_Transformed.csv")
sns.set(style="whitegrid")

for col in df.columns:
    dtype = df[col].dtype

    if pd.api.types.is_numeric_dtype(dtype):
    #If it's numeric run distribution histograms
        #Overall distribution
        plt.figure(figsize=(8, 4))
        sns.histplot(df[col], bins=30, color="skyblue")
        plt.title(f"Overall Distribution of {col}")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.tight_layout()
        plt.show()

        #Grouped by satisfaction
        plt.figure(figsize=(8, 5))
        sns.histplot(
            data=df,
            x=col,
            hue="satisfaction",
            bins=30,
            element="step",
            stat="count",
            common_norm=False,
            palette="muted"
        )
        plt.title(f"{col} Distribution by Satisfaction Group")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.tight_layout()
        plt.show()

    elif pd.api.types.is_object_dtype(dtype) or pd.api.types.is_categorical_dtype(d
    #If it's categorical instead of numeric then now we'll run the counts

        #Overall
        plt.figure(figsize=(8, 4))
        sns.countplot(
            data=df,
            x=col,
            hue=col,
            order=df[col].value_counts().index,
            palette="pastel",
            legend=False
        )
        plt.title(f"Value Counts for {col}")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

        #by satisfaction
```

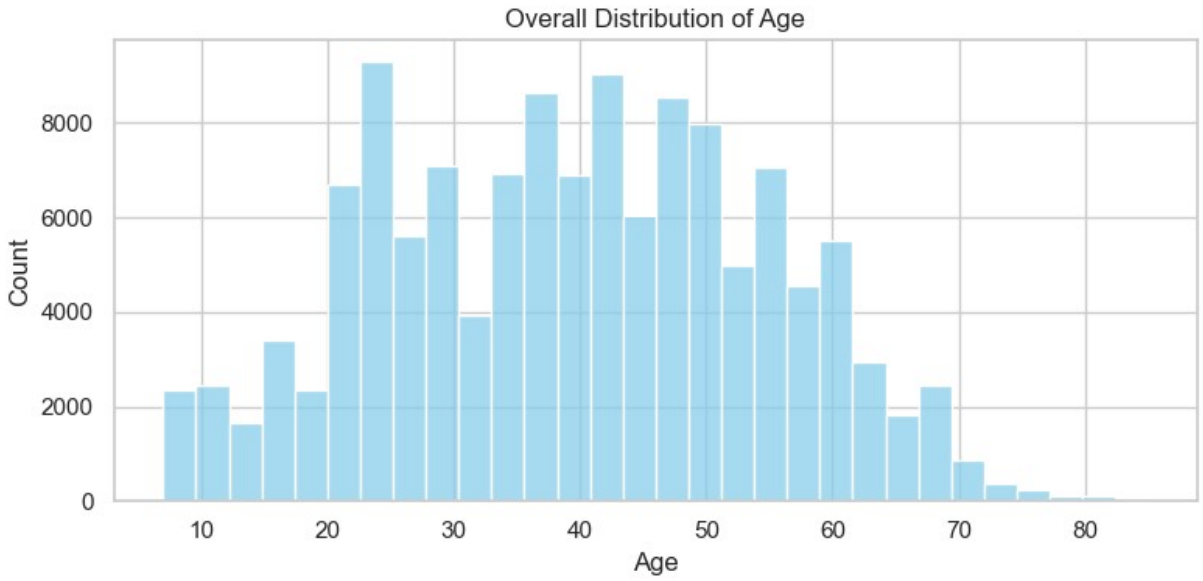
```
        plt.figure(figsize=(8, 5))
        sns.countplot(
            data=df,
            x=col,
            hue="satisfaction",
            order=df[col].value_counts().index,
            palette="muted"
        )
        plt.title(f"{col} by Satisfaction Group")
        plt.xlabel(col)
        plt.ylabel("Count")
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
```

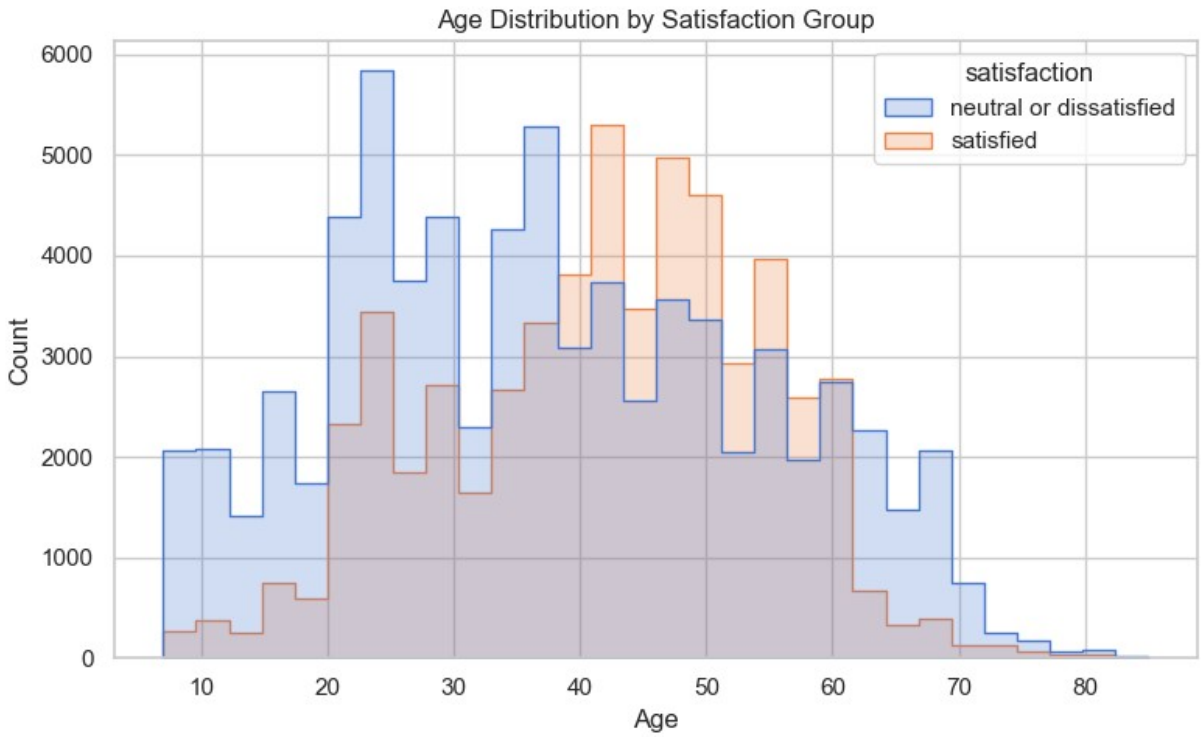


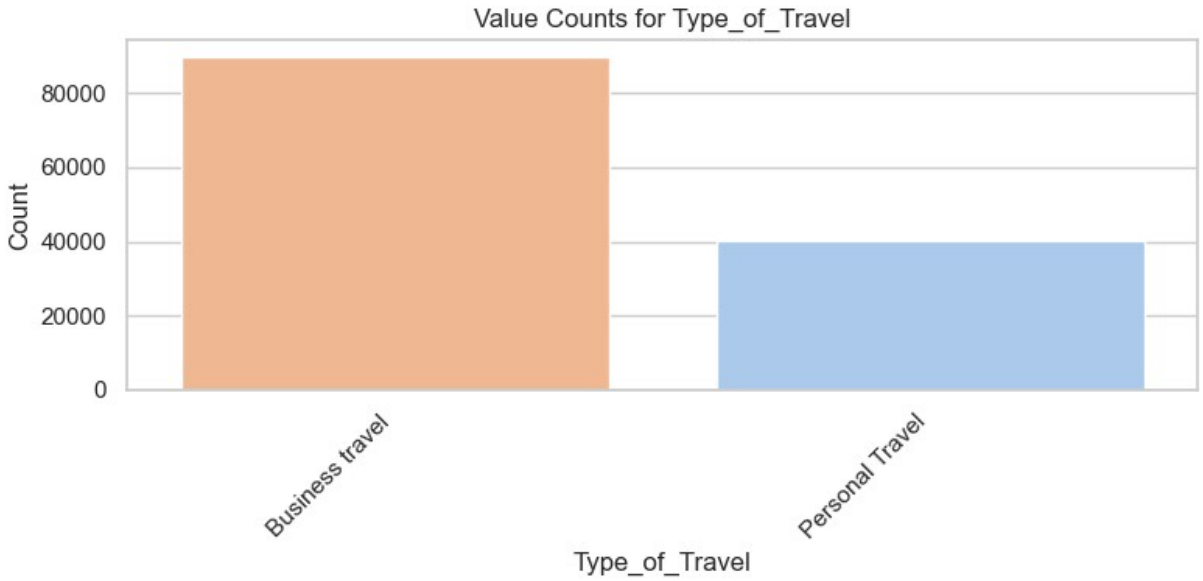


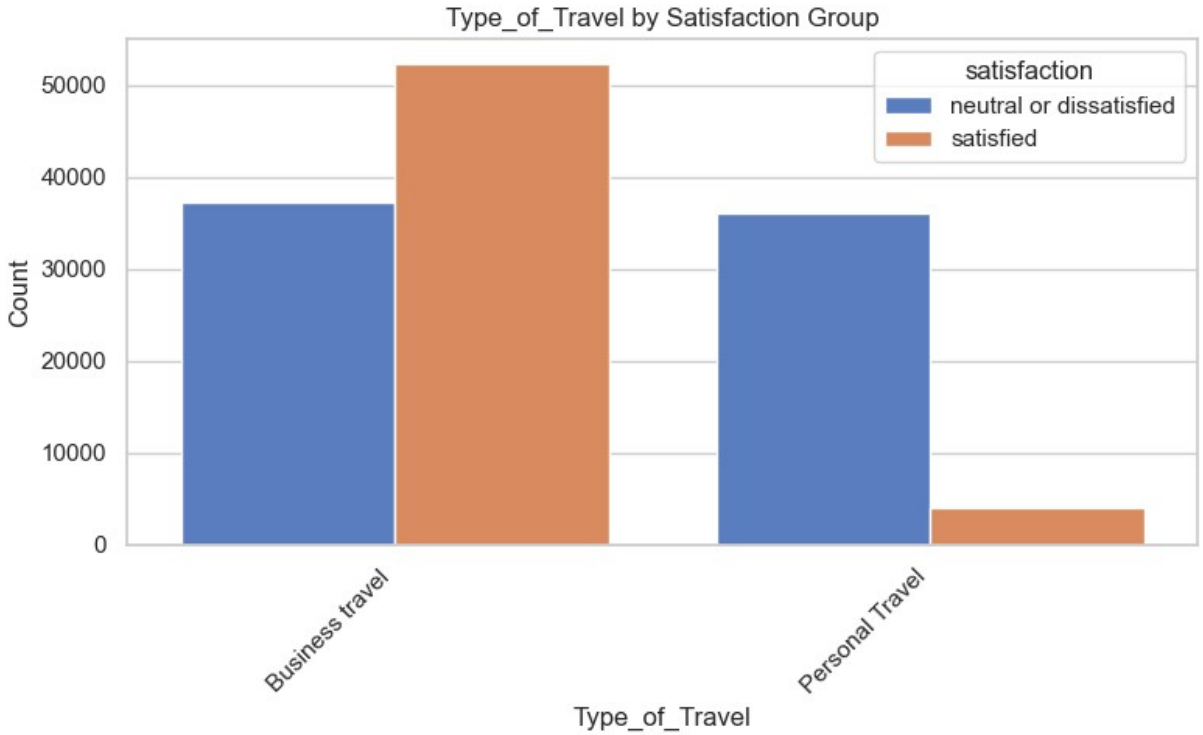

Value Counts for Customer_Type
Count
100000
80000
60000
40000
20000
0
Loyal Customer
disloyal Customer
Customer_Type


Customer_Type by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
50000
40000
30000
20000
10000
0
Loyal Customer
disloyal Customer
Customer_Type

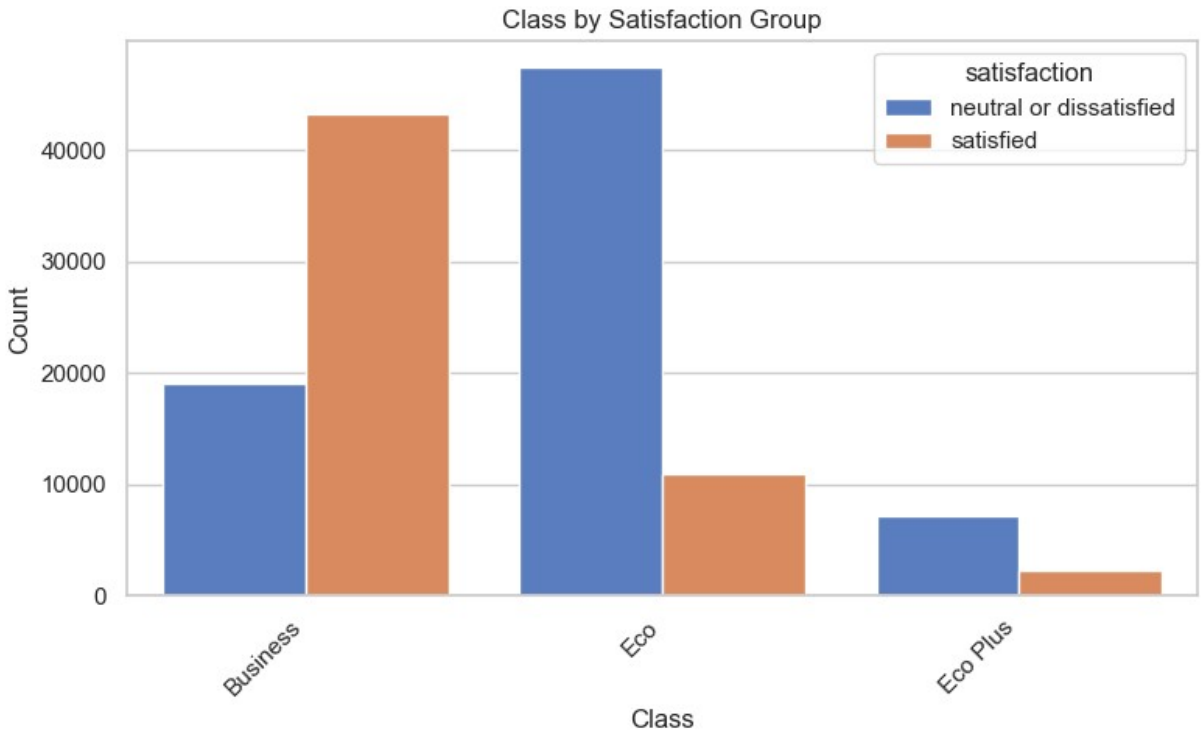

Overall Distribution of Age
Count
0
2000
4000
6000
8000
10
20
30
40
50
60
70
80
Age

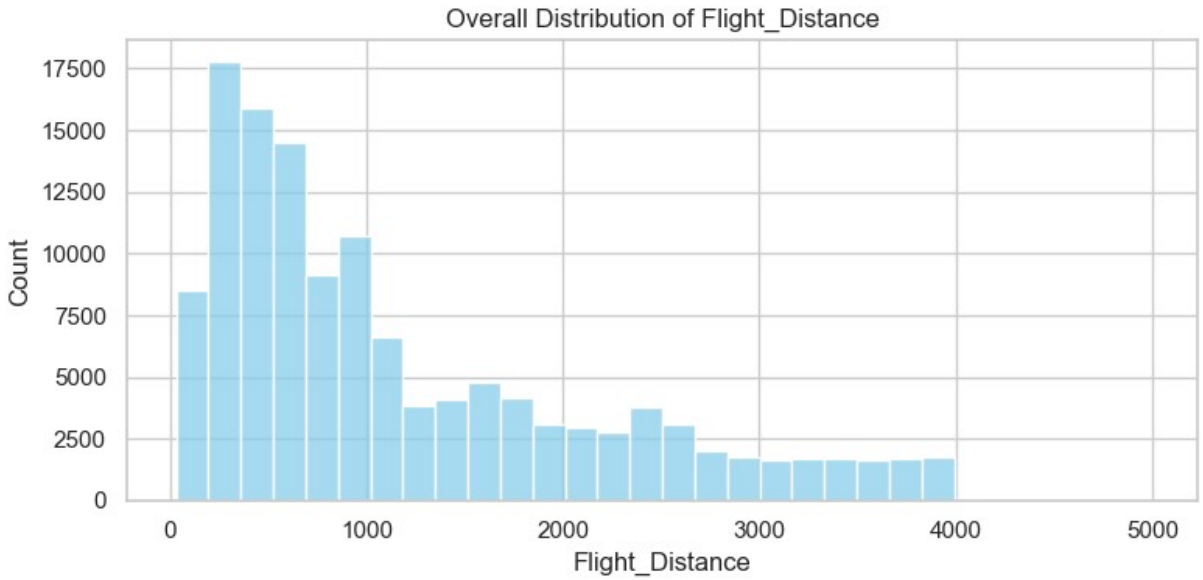

Age Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
0
1000
2000
3000
4000
5000
6000
10
20
30
40
50
60
70
80
Age

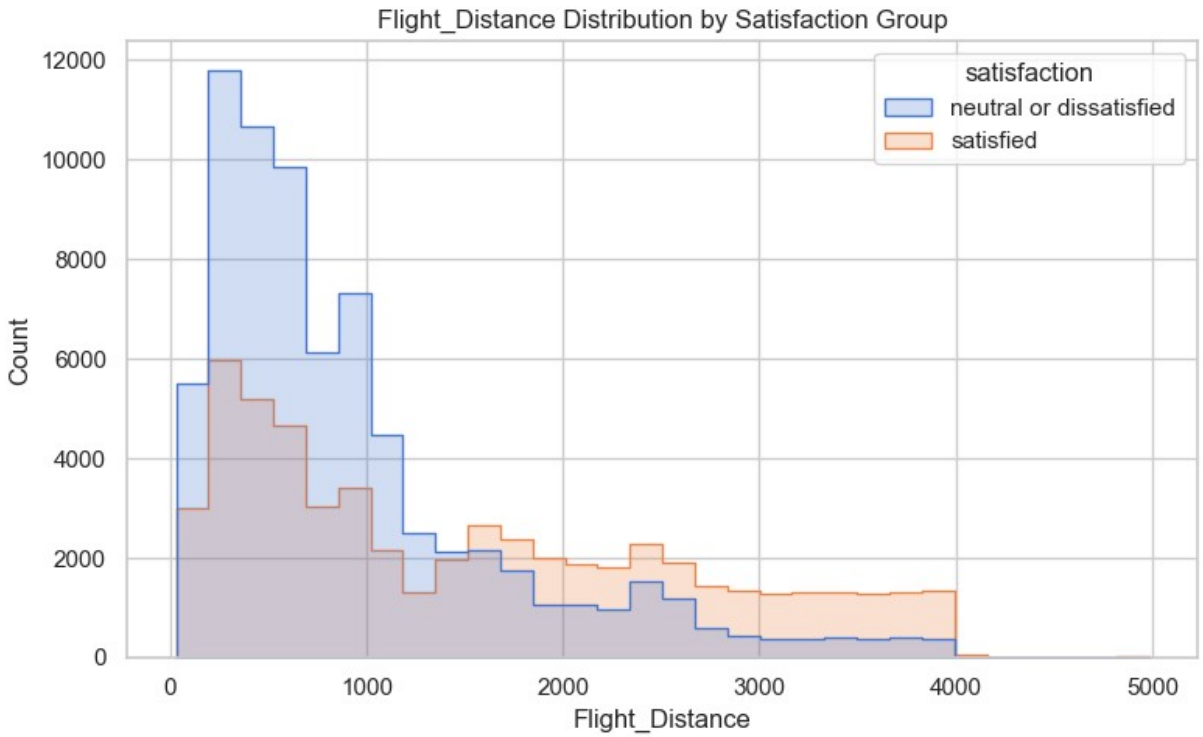

Value Counts for Type_of_Travel
Count
80000
60000
40000
20000
0
Business travel
Personal Travel
Type_of_Travel

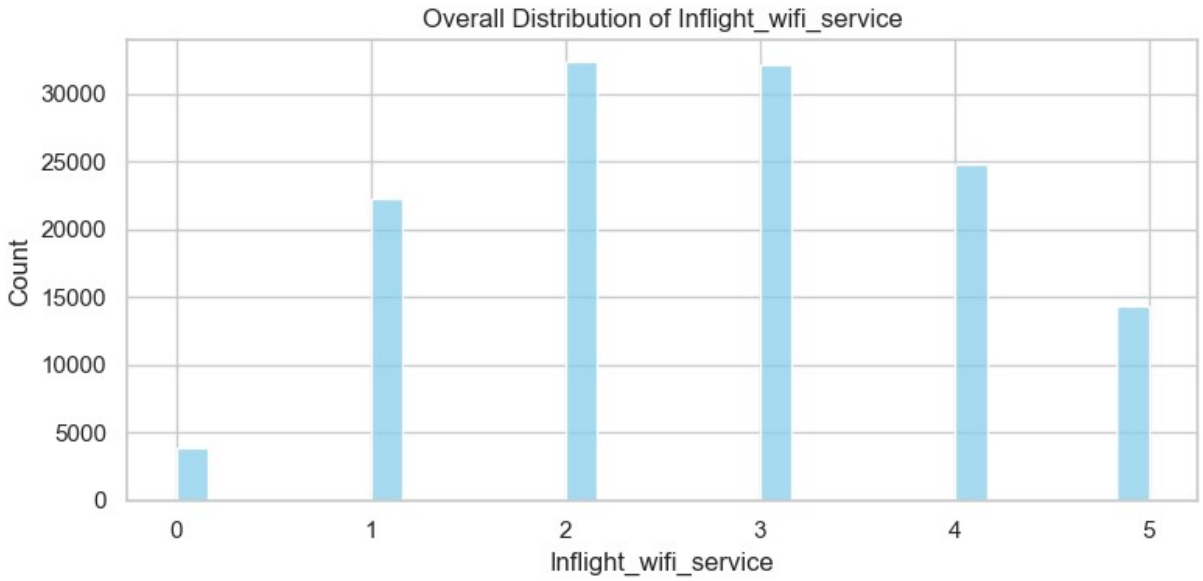

Type_of_Travel by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
50000
40000
30000
20000
10000
0
Business travel
Personal Travel
Type_of_Travel

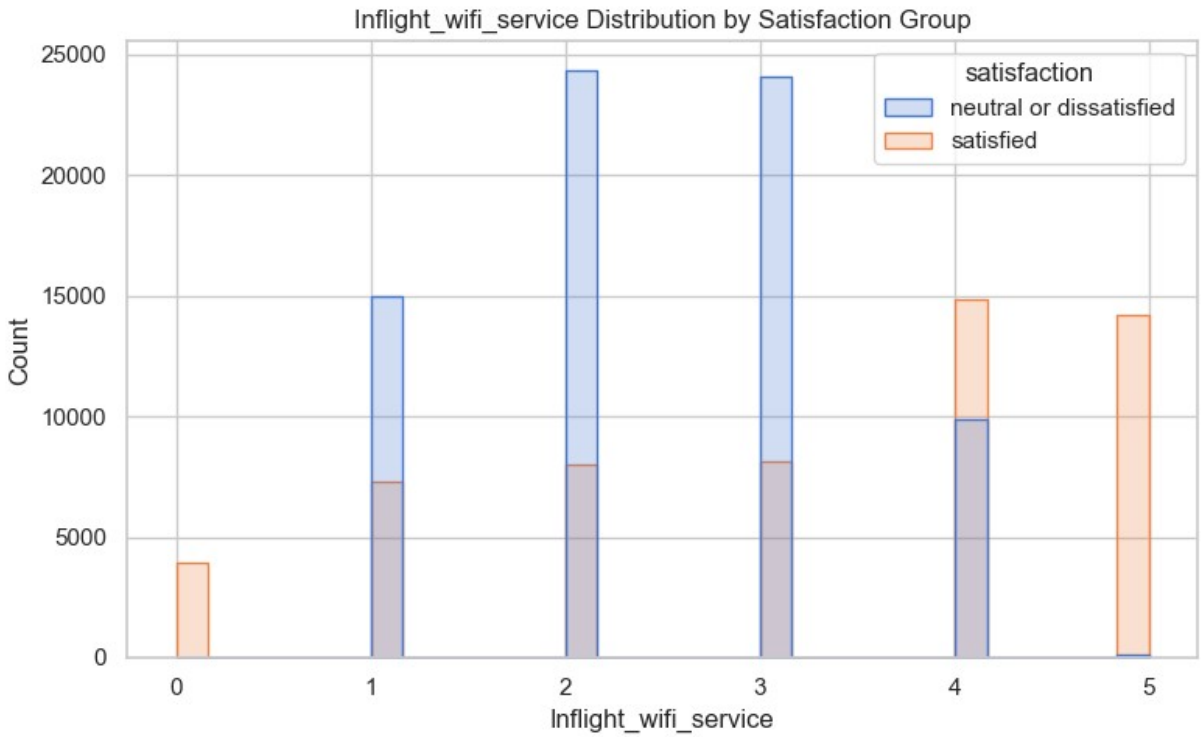

Value Counts for Class
Count
60000
50000
40000
30000
20000
10000
0
Business
Eco
Eco Plus
Class

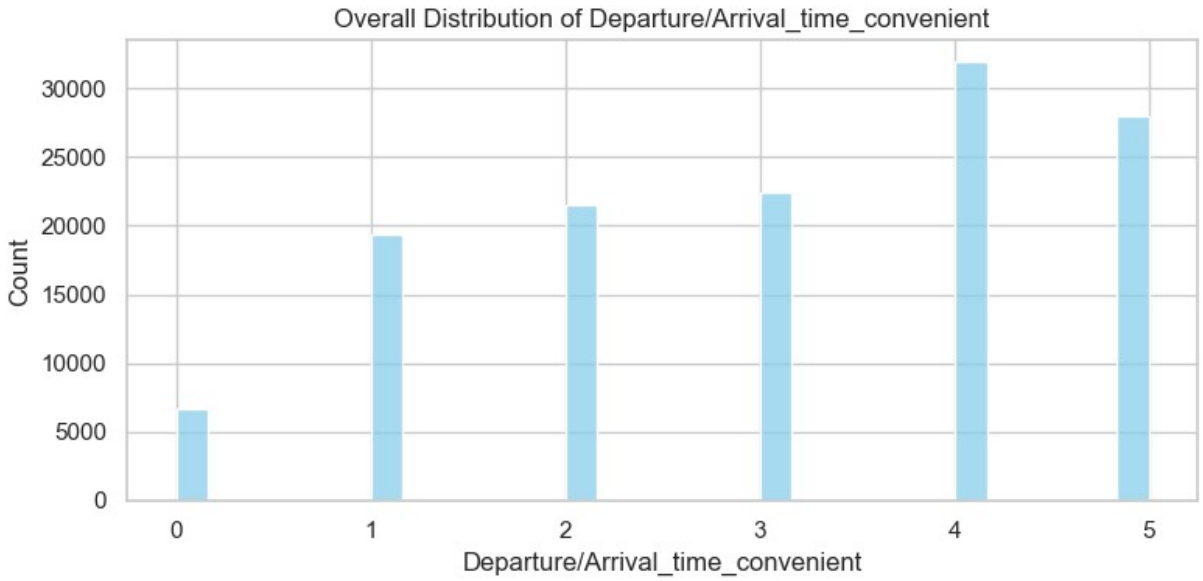

Class by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
40000
30000
20000
10000
0
Business
Eco
Eco Plus
Class

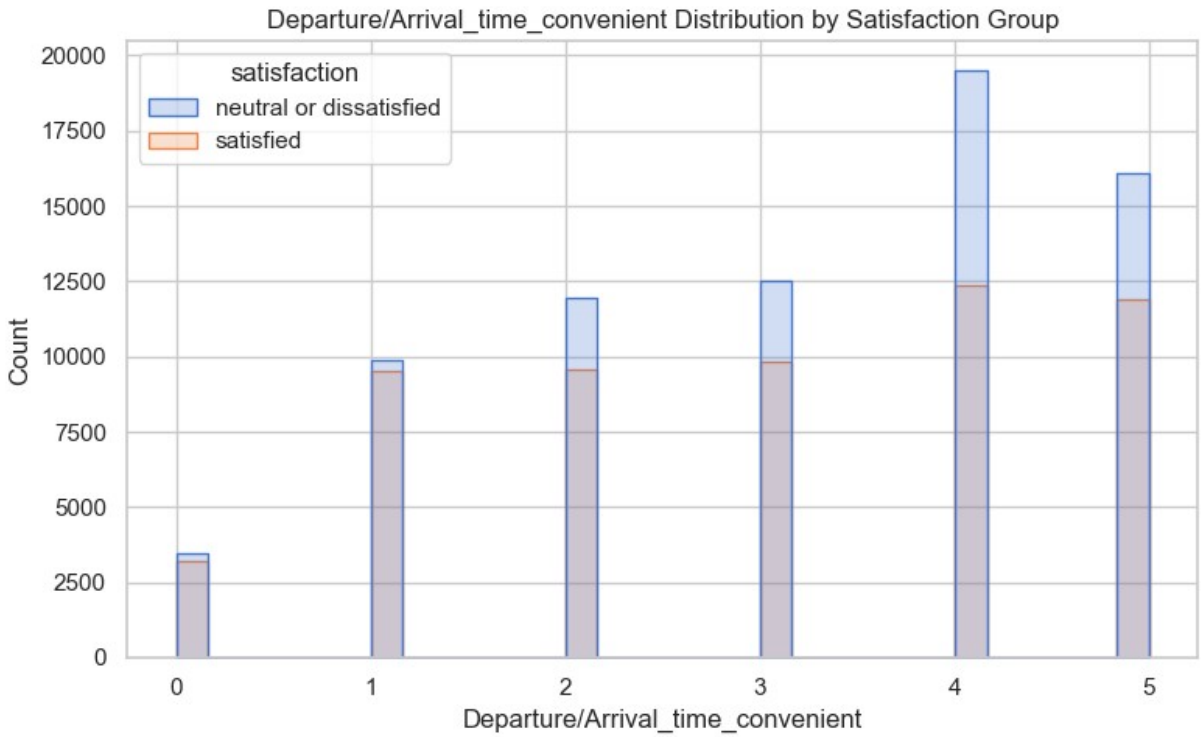

Overall Distribution of Flight_Distance
Count
Flight_Distance
0
2500
5000
7500
10000
12500
15000
17500
0
1000
2000
3000
4000
5000


Flight_Distance Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Flight_Distance
0
2000
4000
6000
8000
10000
12000
0
1000
2000
3000
4000
5000

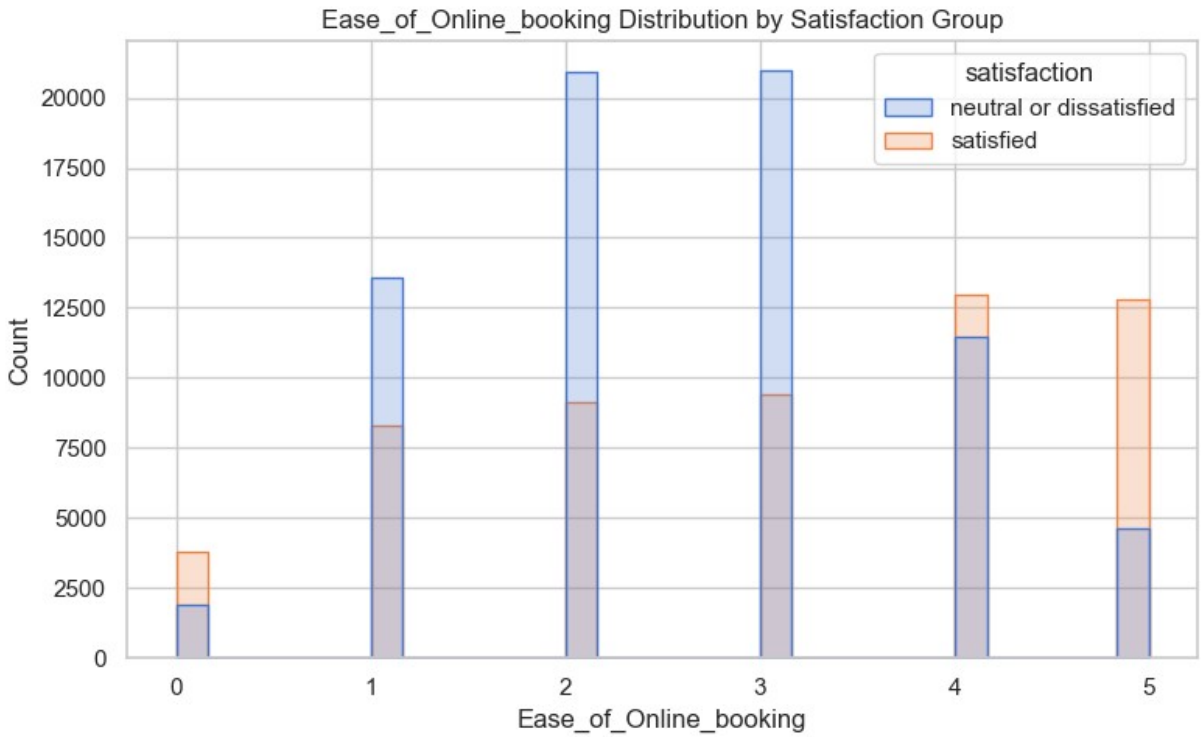

Overall Distribution of Inflight_wifi_service
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_wifi_service

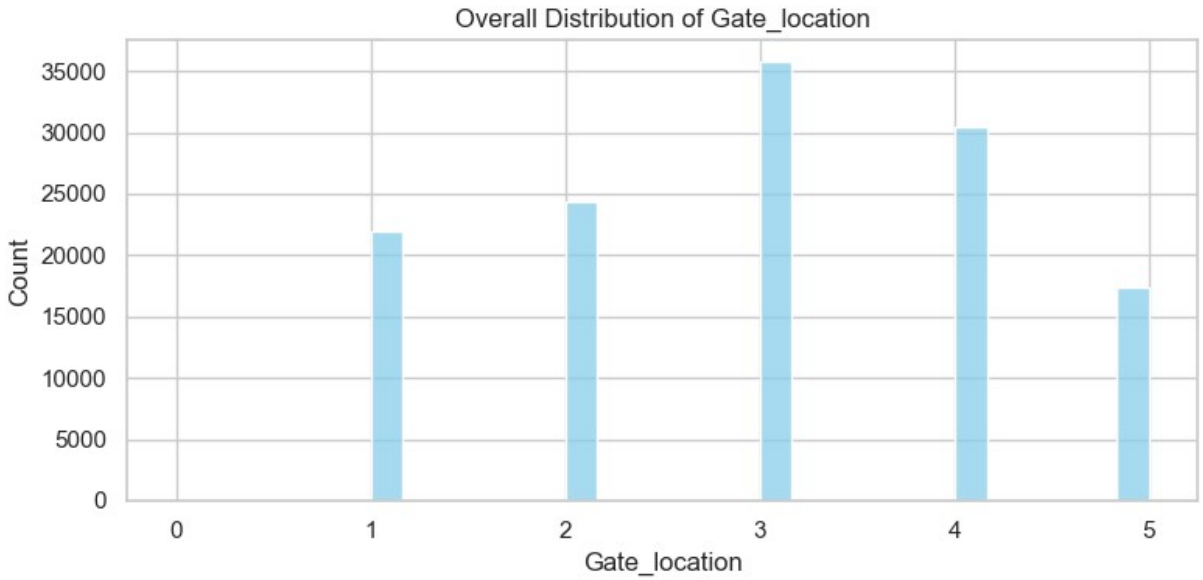

Inflight_wifi_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_wifi_service

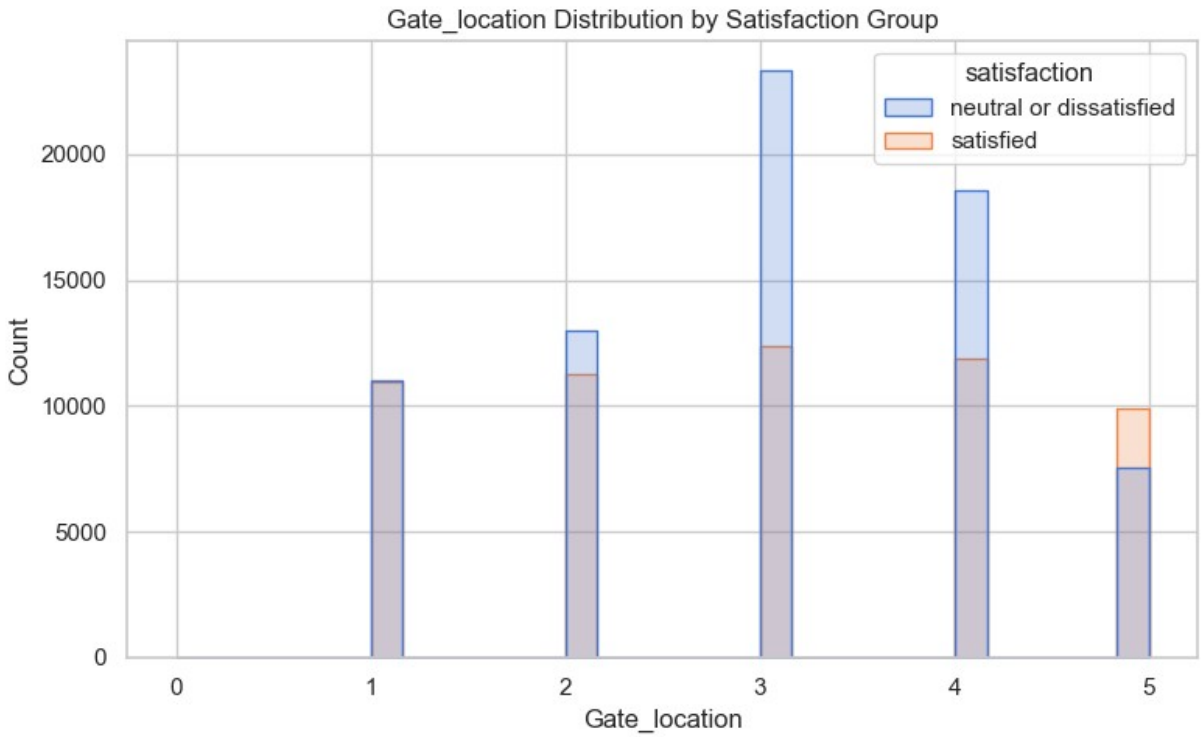

Overall Distribution of Departure/Arrival_time_convenient
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Departure/Arrival_time_convenient

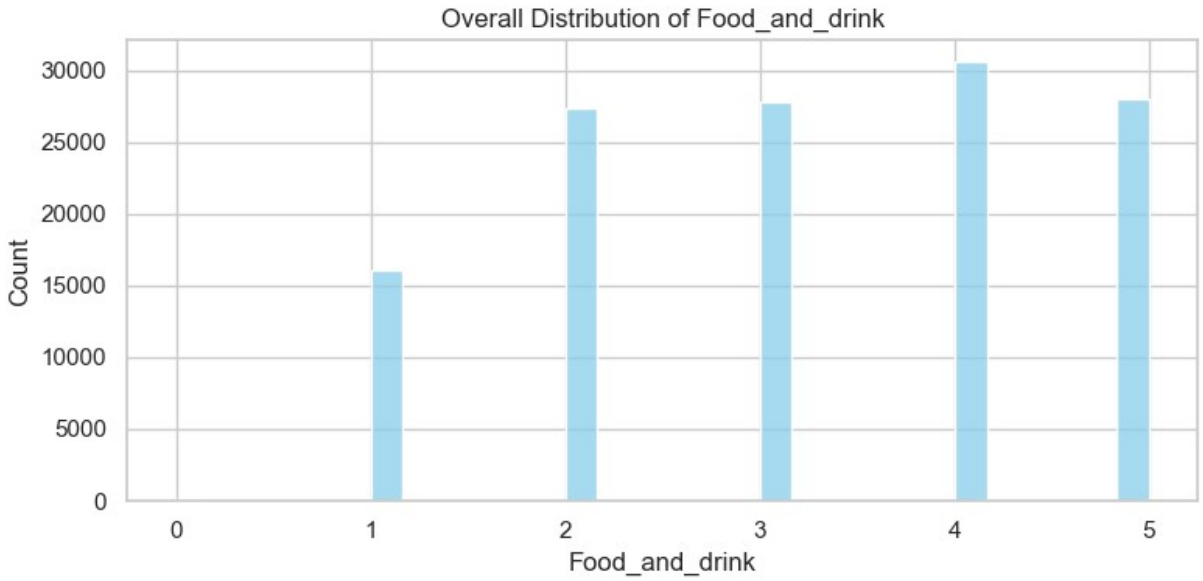

Departure/Arrival_time_convenient Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Departure/Arrival_time_convenient

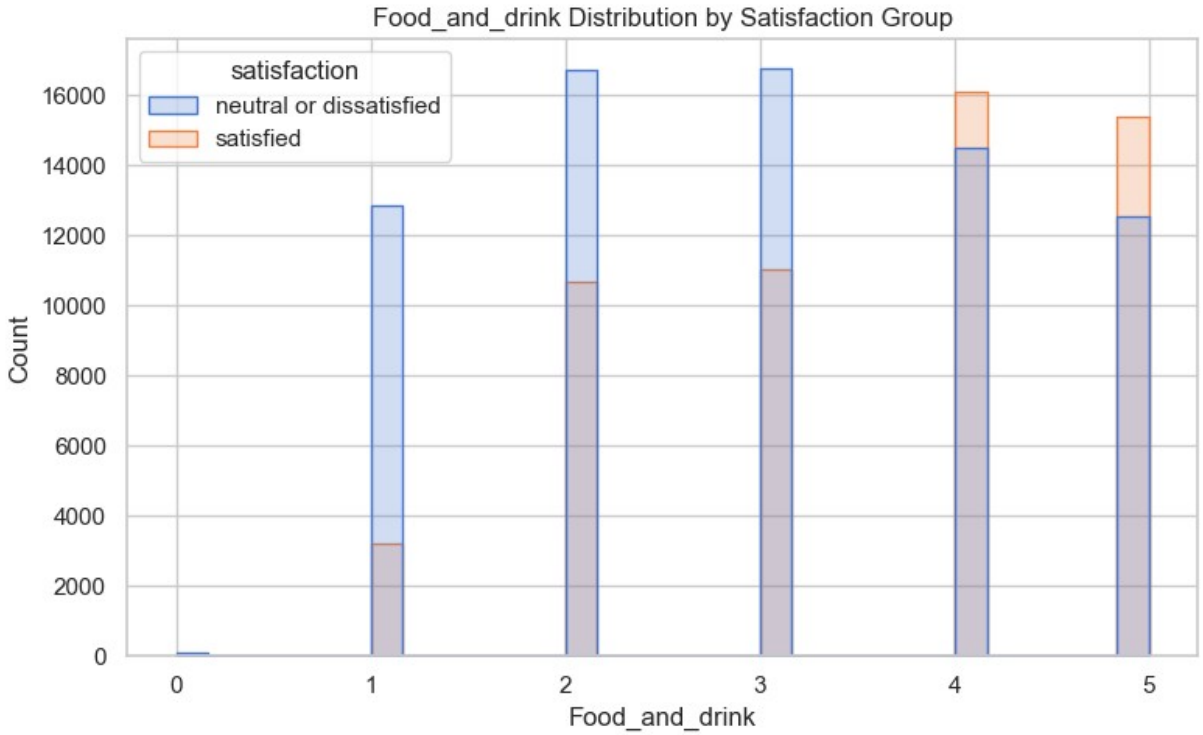

Overall Distribution of Ease_of_Online_booking
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Ease_of_Online_booking

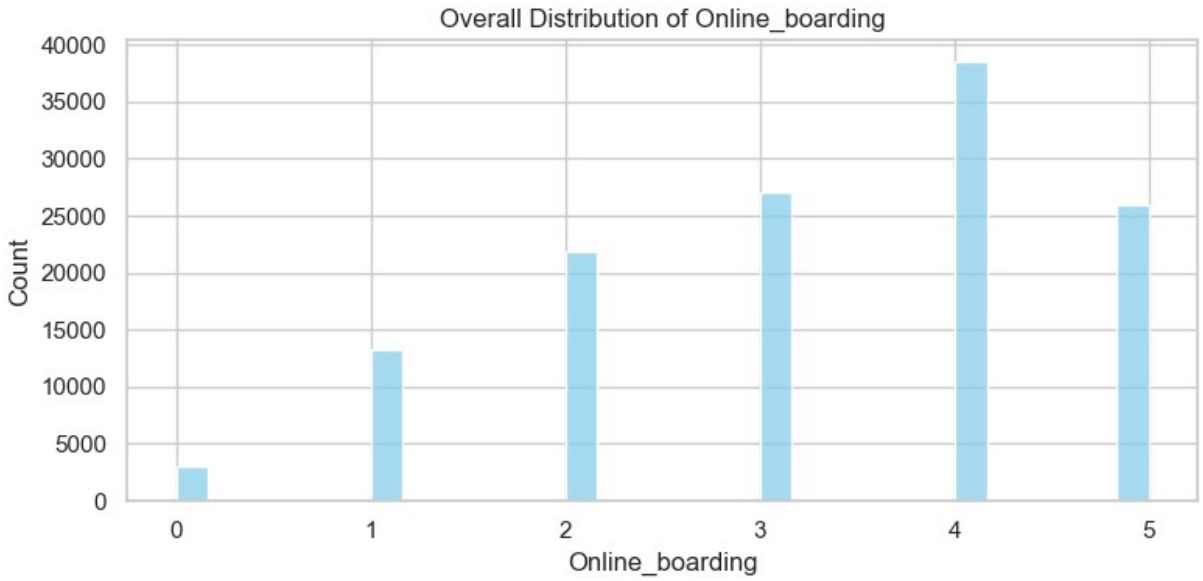

Ease_of_Online_booking Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Ease_of_Online_booking

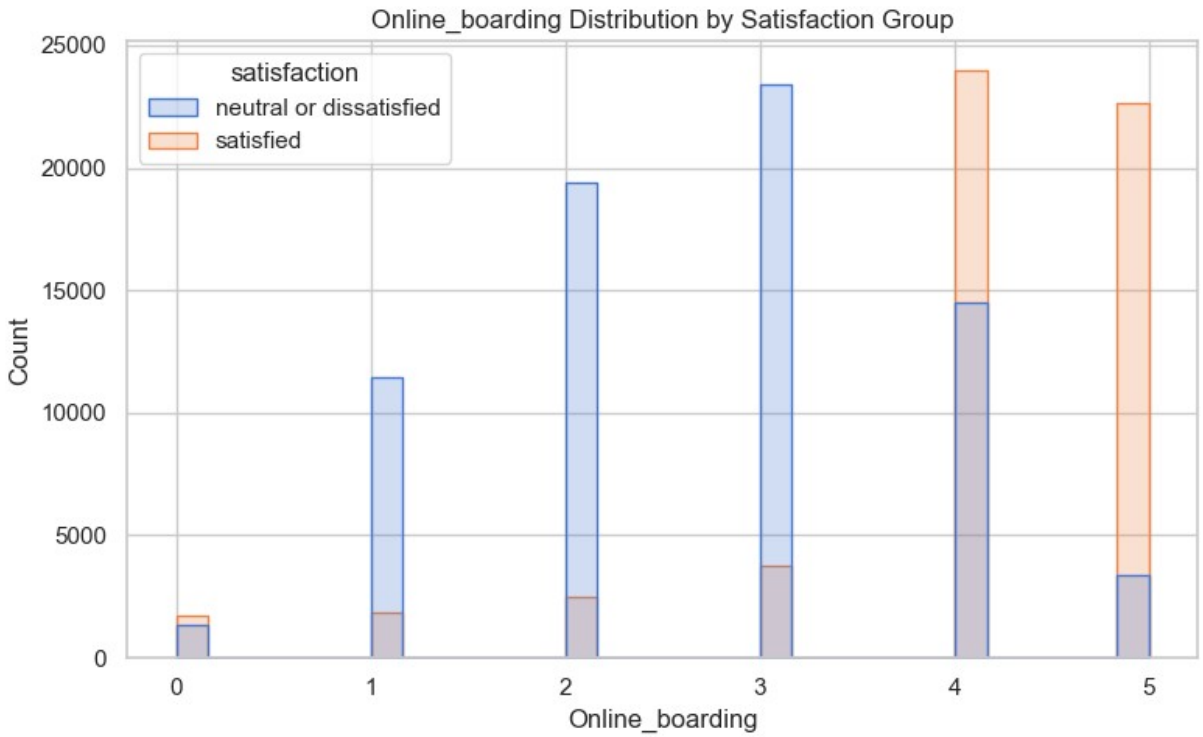

Overall Distribution of Gate_location
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Gate_location

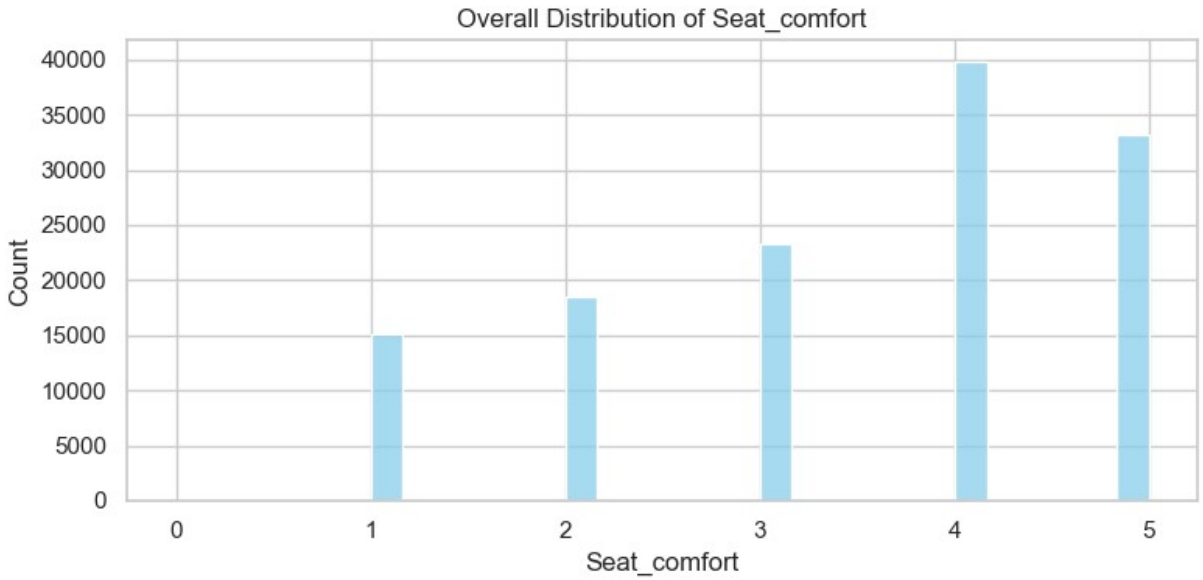

Gate_location Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
15000
10000
5000
0
0
1
2
3
4
5
Gate_location

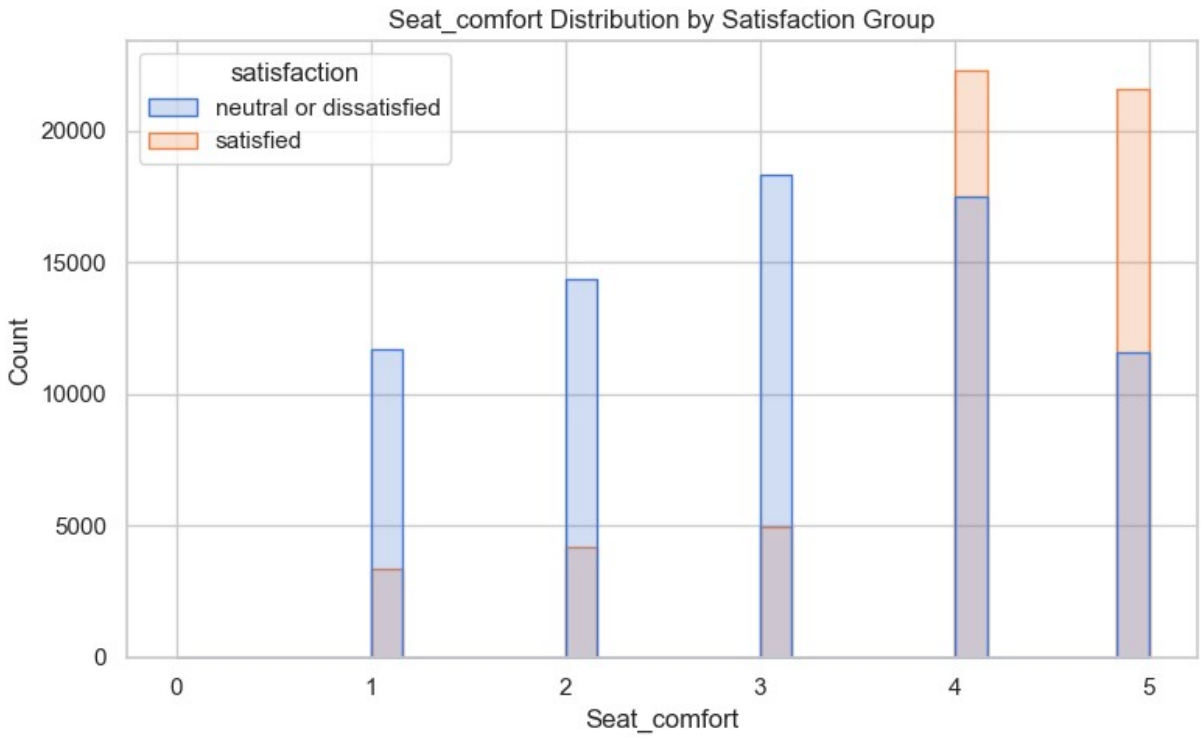

Overall Distribution of Food_and_drink
Count
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Food_and_drink

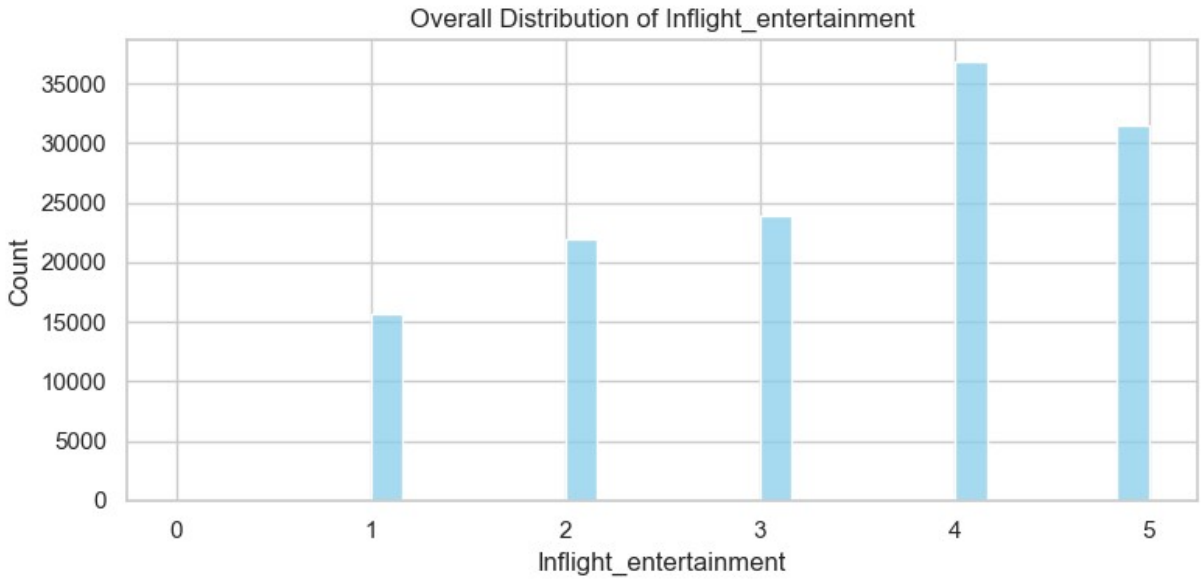

Food_and_drink Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
Food_and_drink

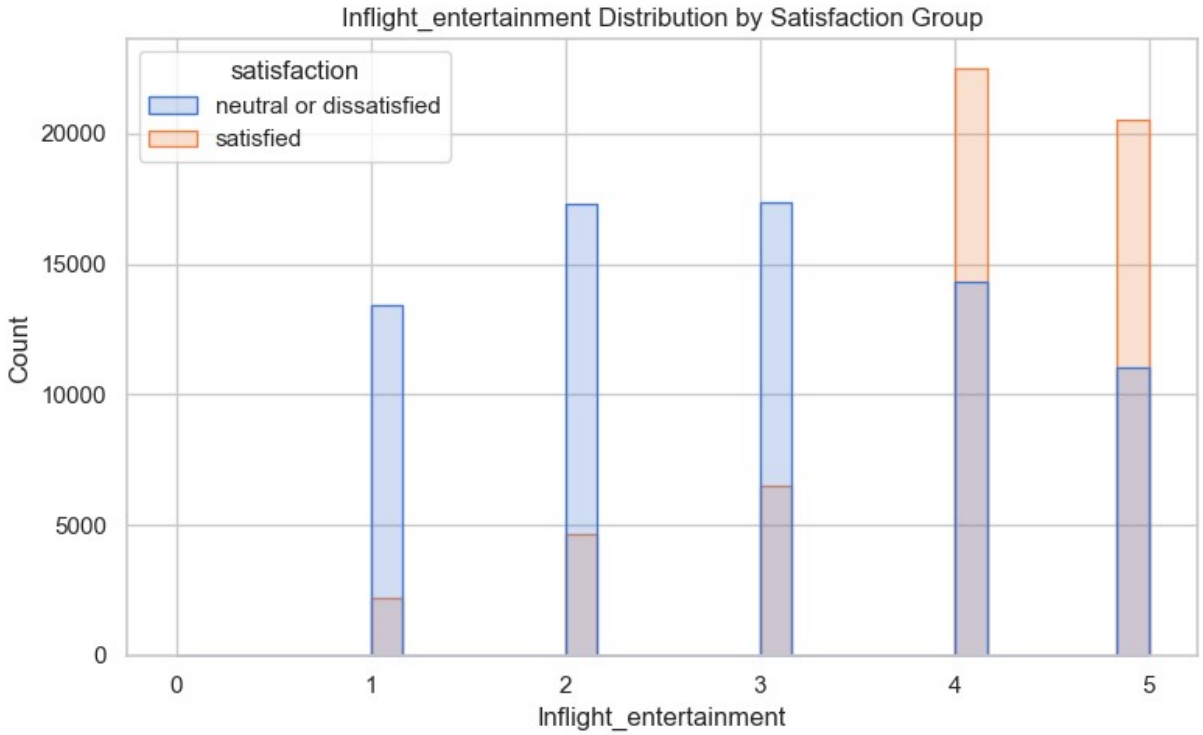

Overall Distribution of Online_boarding
Count
Online_boarding
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

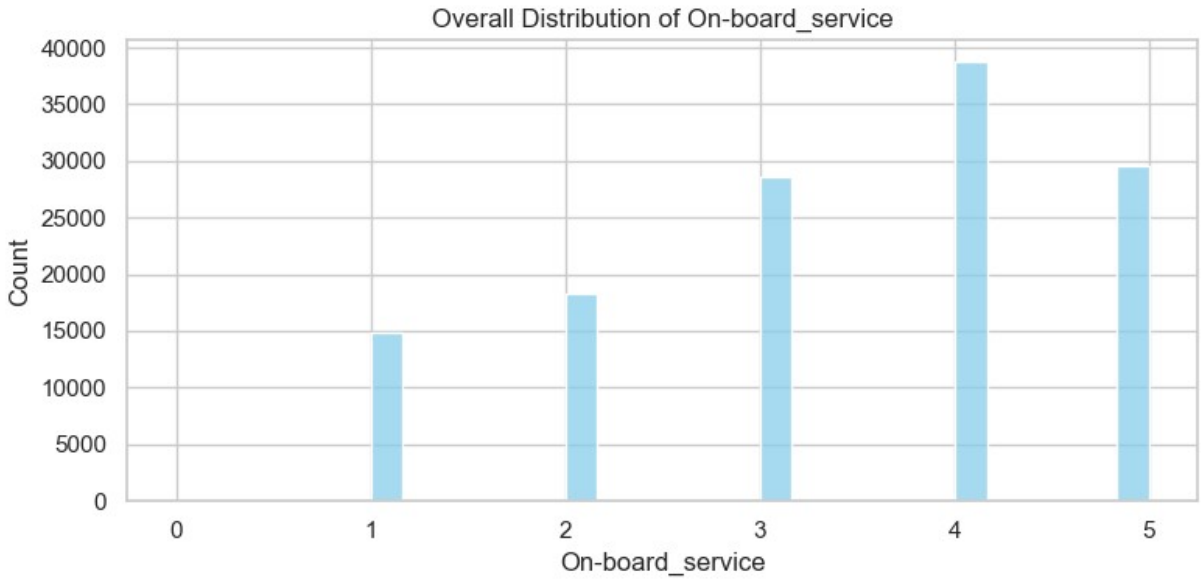

Online_boarding Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Online_boarding
0
5000
10000
15000
20000
25000
0
1
2
3
4
5

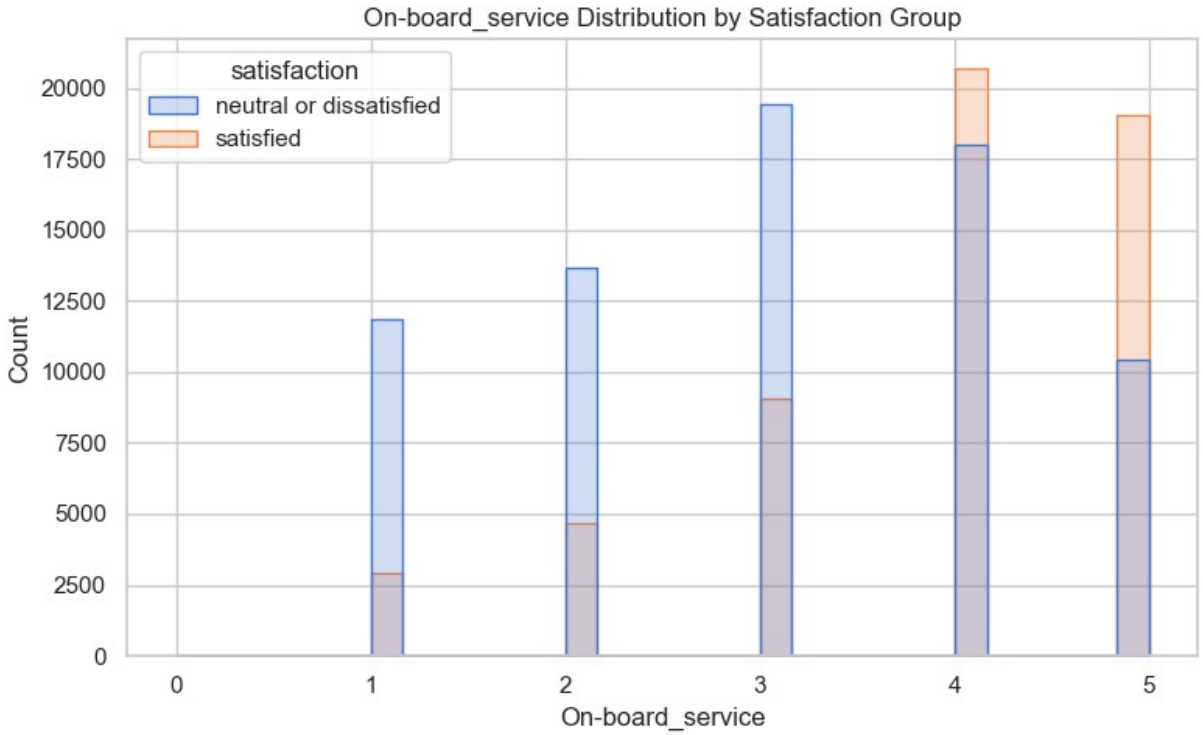

Overall Distribution of Seat_comfort
Count
Seat_comfort
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

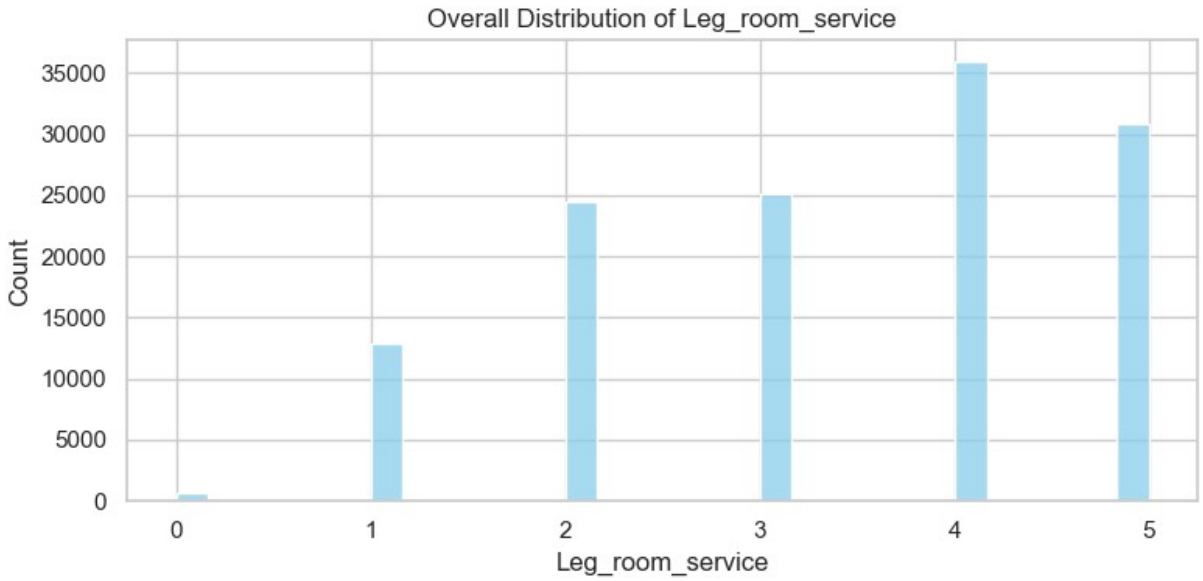

Seat_comfort Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
Seat_comfort
0
5000
10000
15000
20000
0
1
2
3
4
5

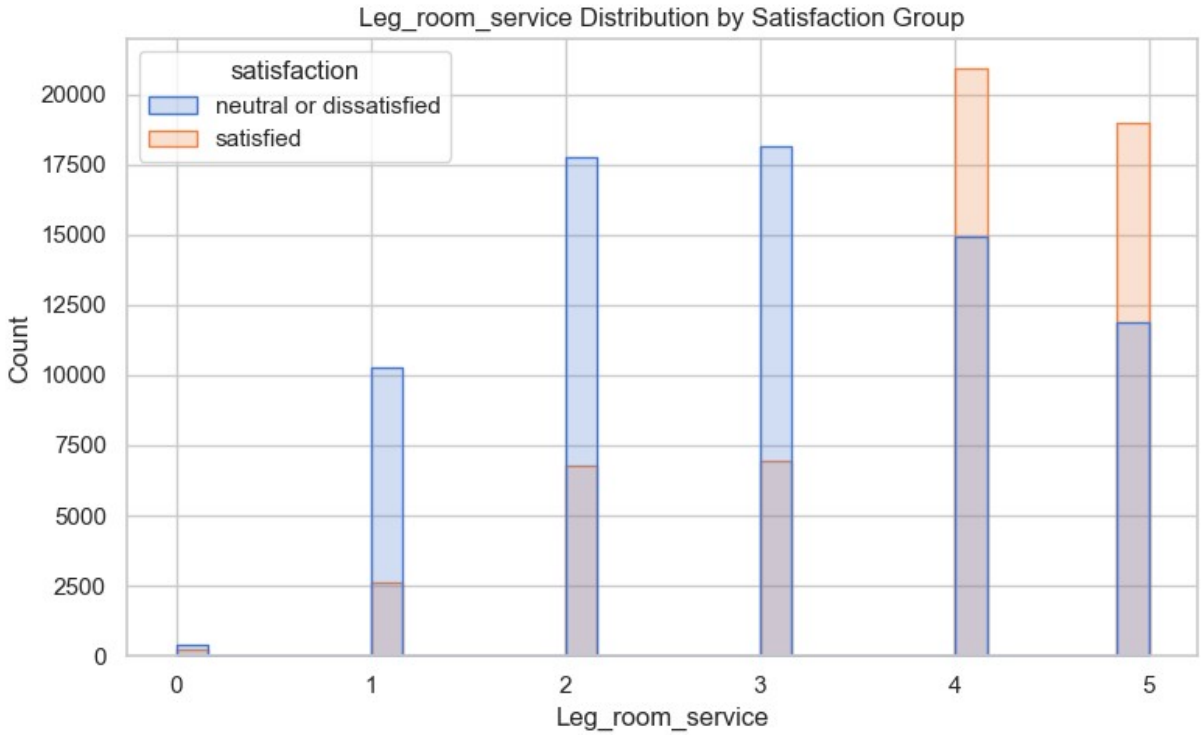

Overall Distribution of Inflight_entertainment
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_entertainment

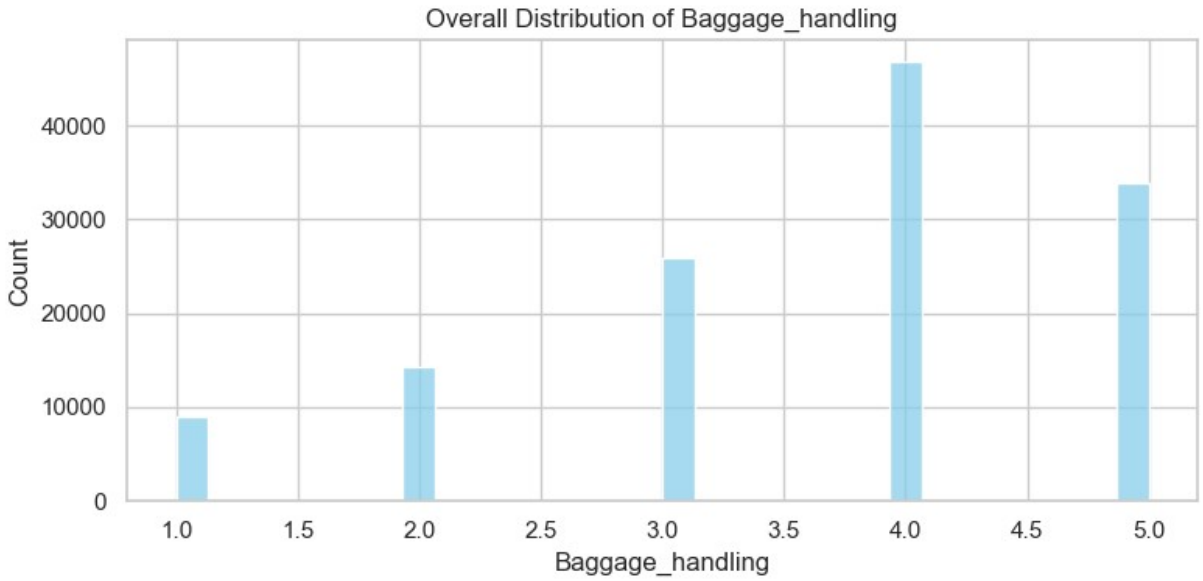

Inflight_entertainment Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_entertainment

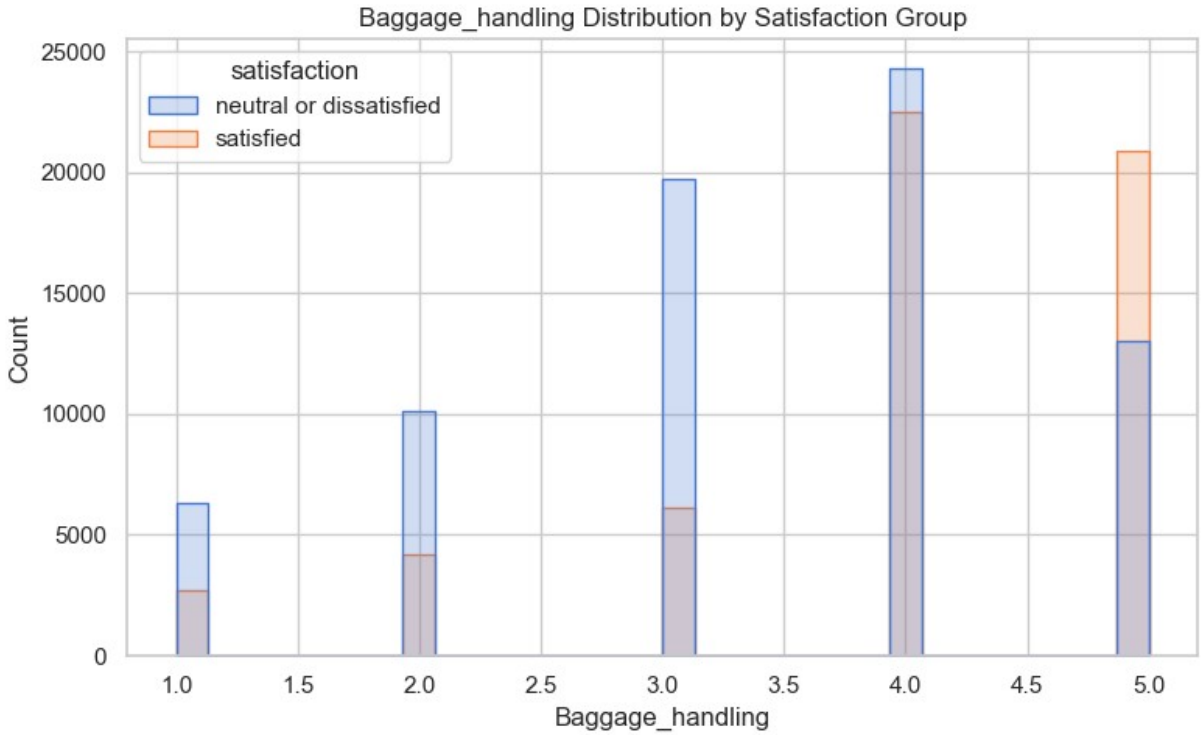

Overall Distribution of On-board_service
Count
On-board_service
0
5000
10000
15000
20000
25000
30000
35000
40000
0
1
2
3
4
5

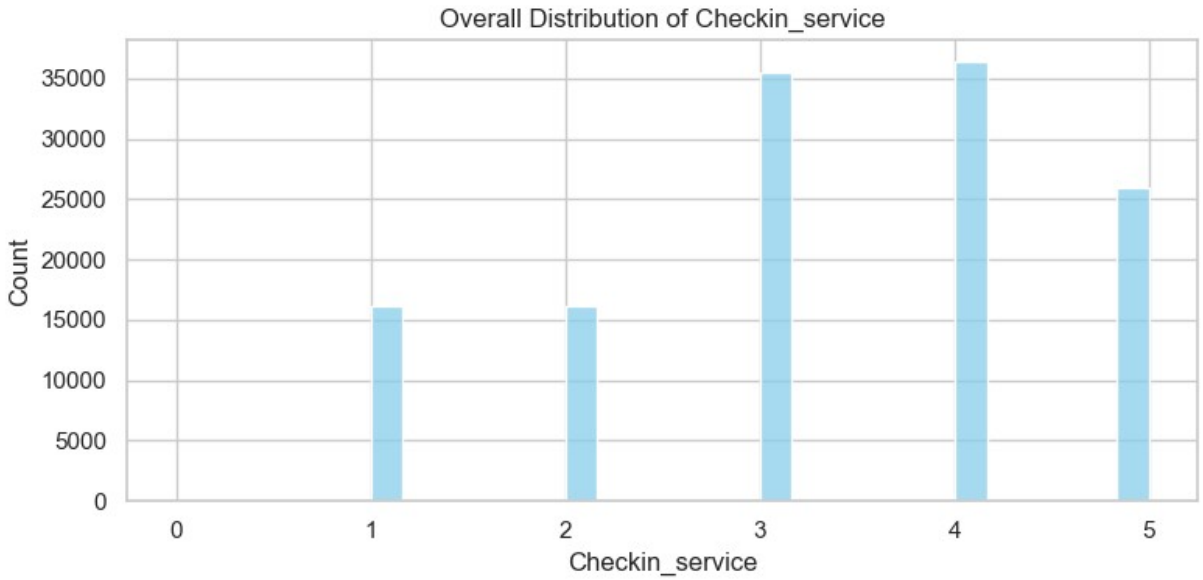

On-board_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
On-board_service
0
2500
5000
7500
10000
12500
15000
17500
20000
0
1
2
3
4
5

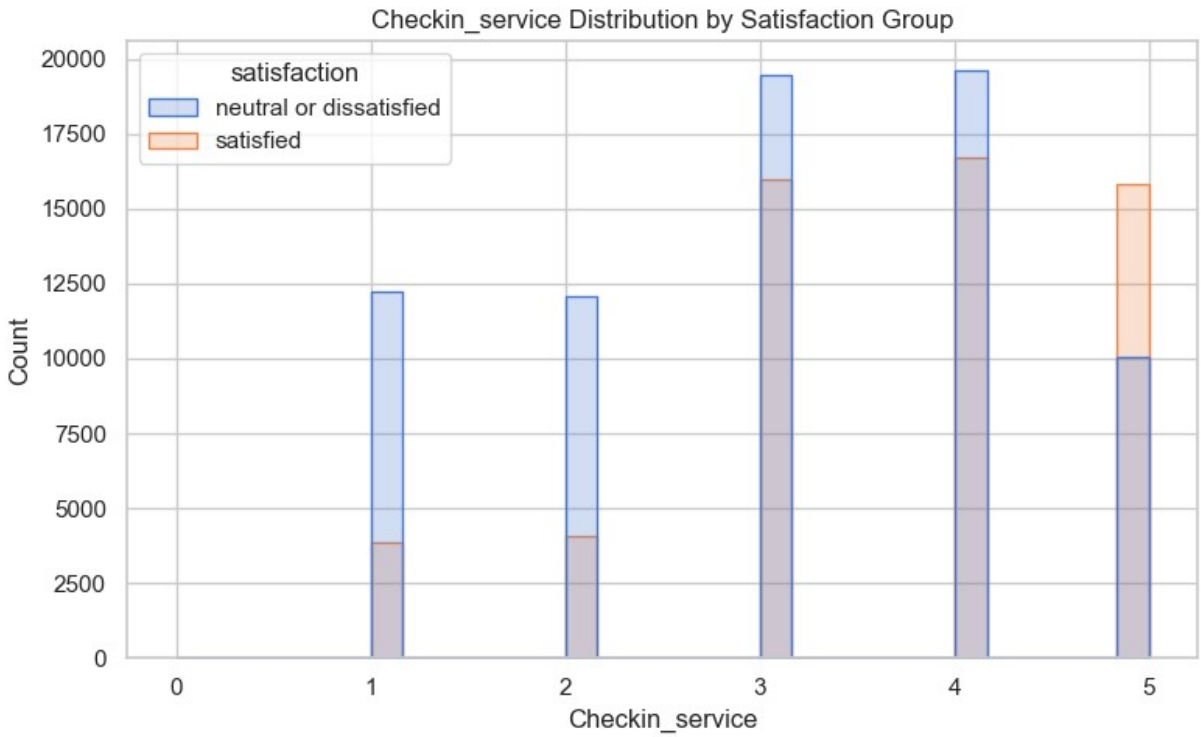

Overall Distribution of Leg_room_service
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Leg_room_service

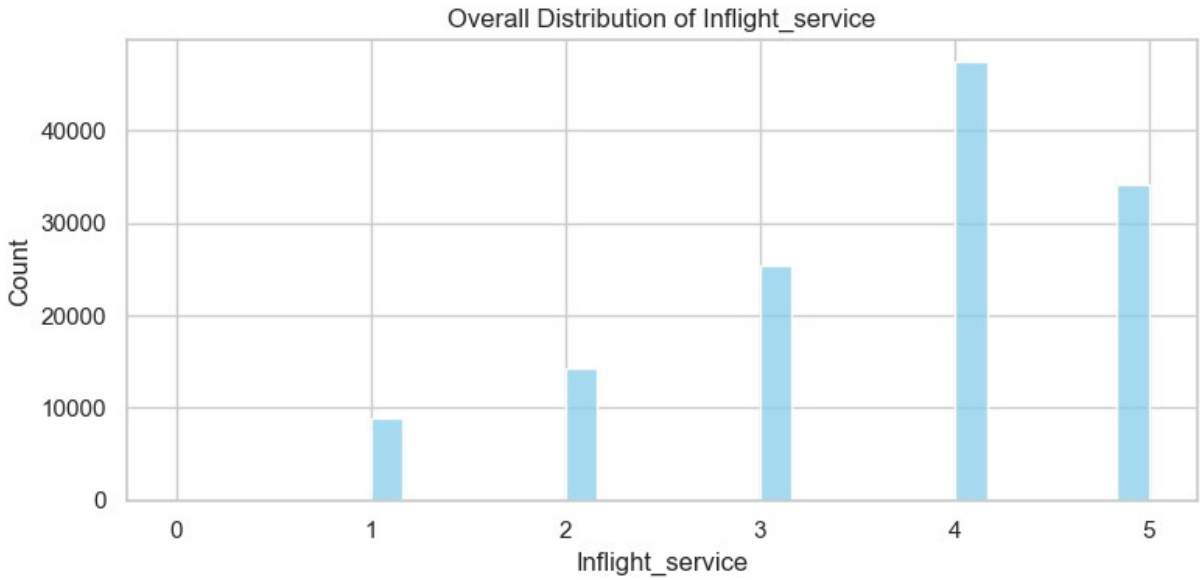

Leg_room_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Leg_room_service

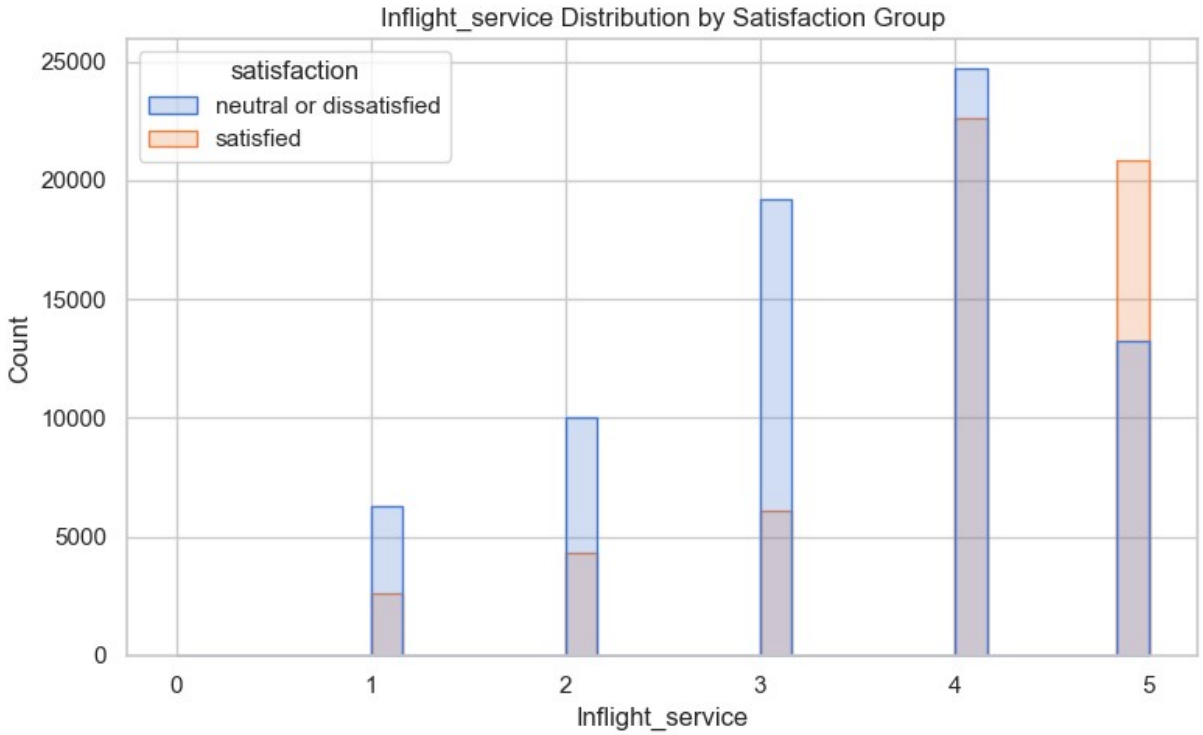

Overall Distribution of Baggage_handling
Count
0
10000
20000
30000
40000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Baggage_handling

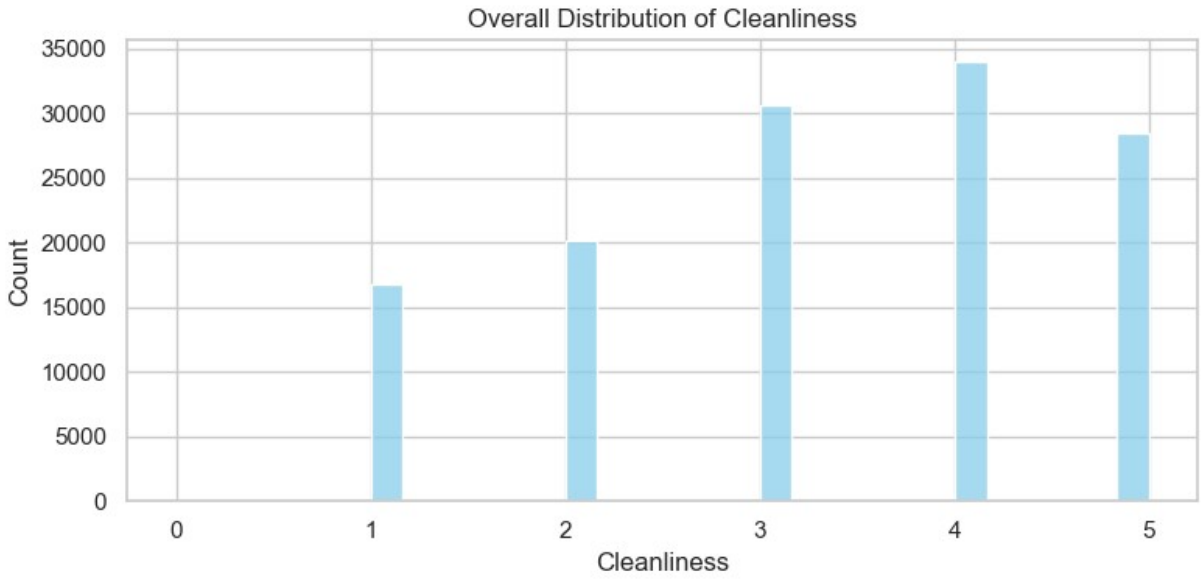

Baggage_handling Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
0
5000
10000
15000
20000
25000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Baggage_handling

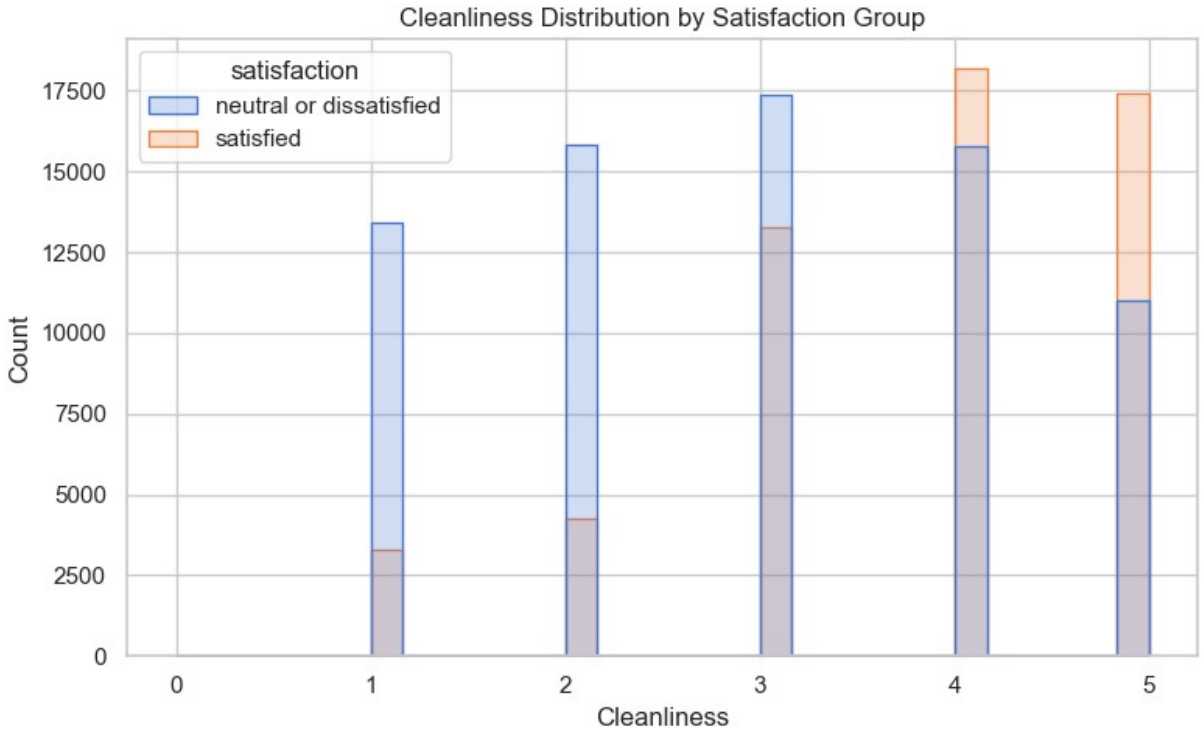

Overall Distribution of Checkin_service
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Checkin_service

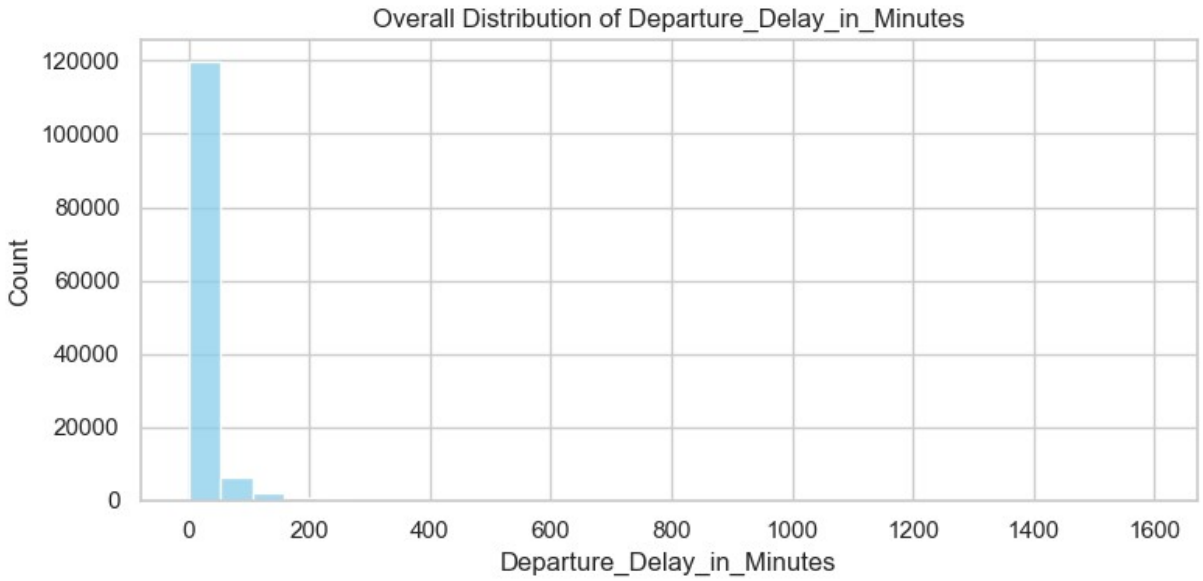

Checkin_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Checkin_service

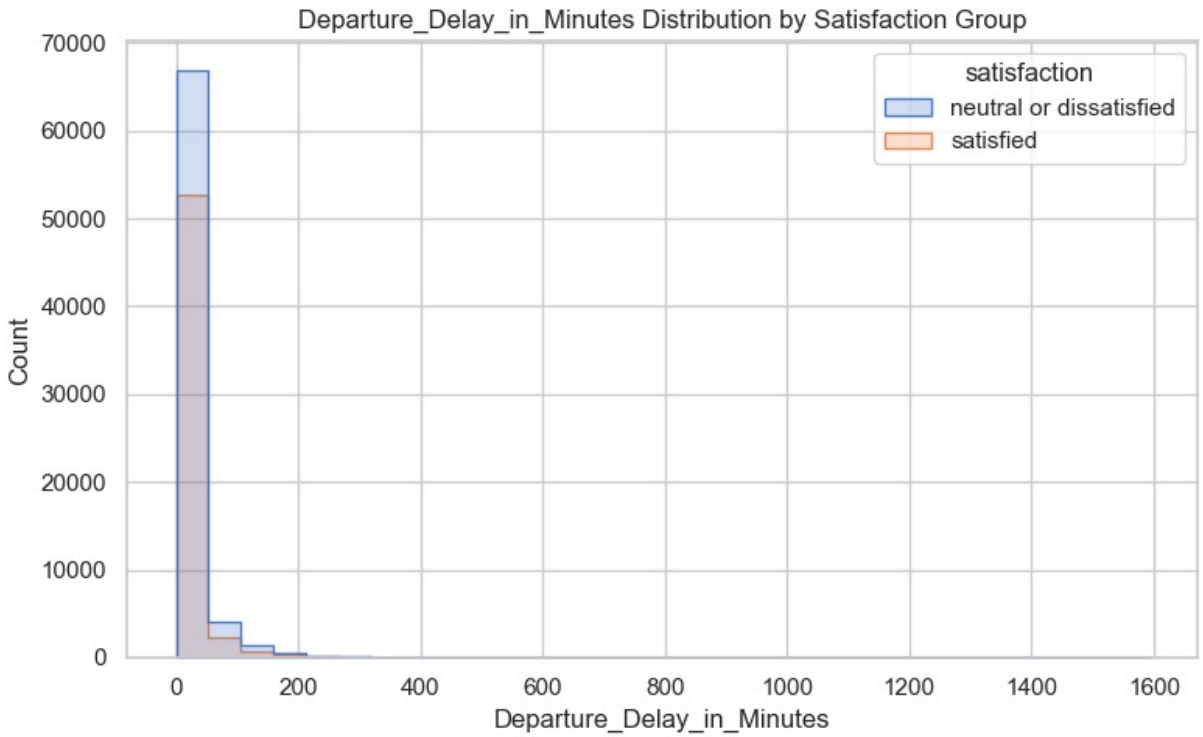

Overall Distribution of Inflight_service
Count
40000
30000
20000
10000
0
0
1
2
3
4
5
Inflight_service

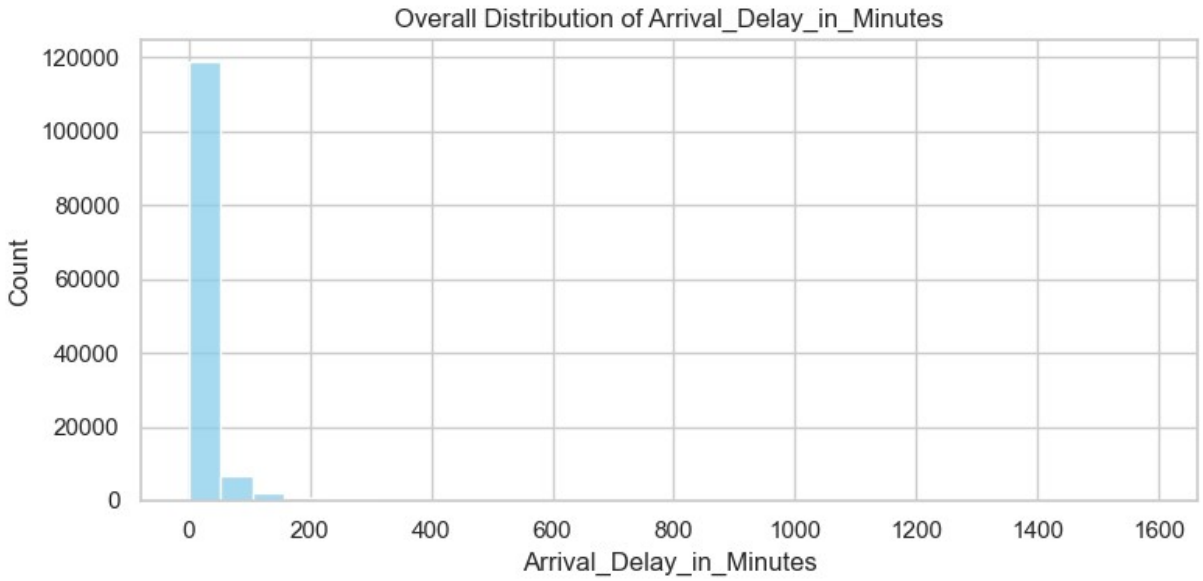

Inflight_service Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Inflight_service

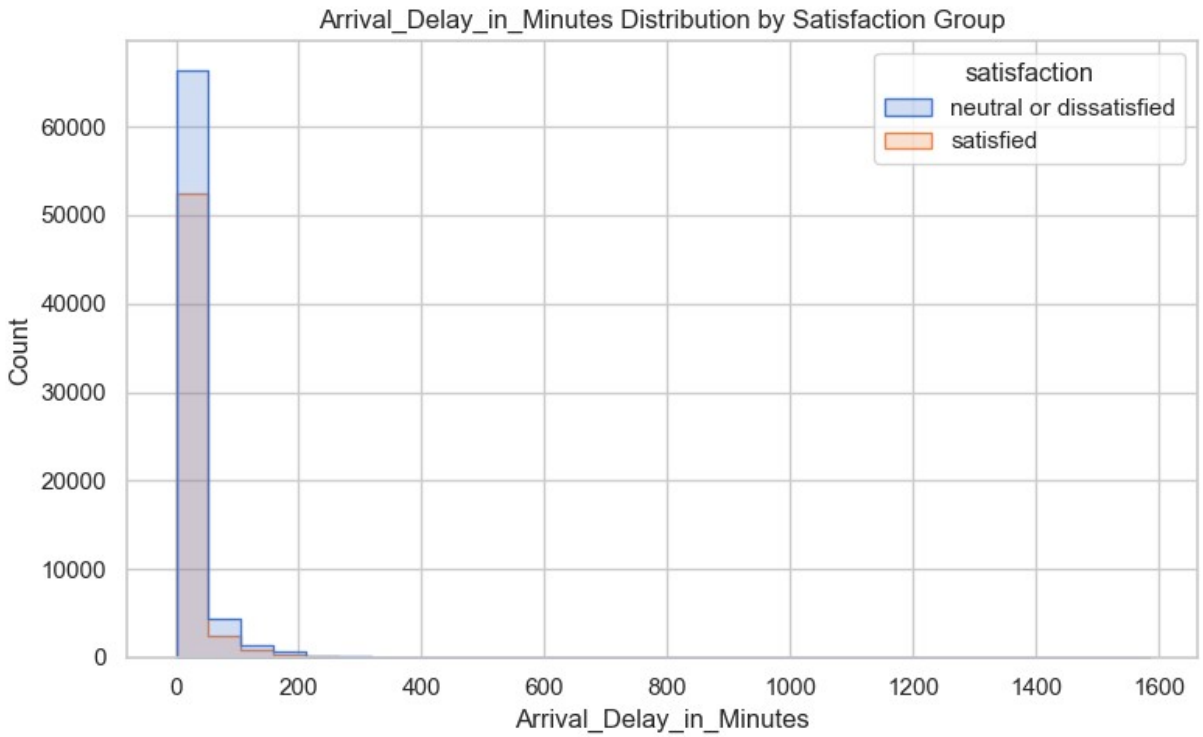

Overall Distribution of Cleanliness
Count
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
Cleanliness

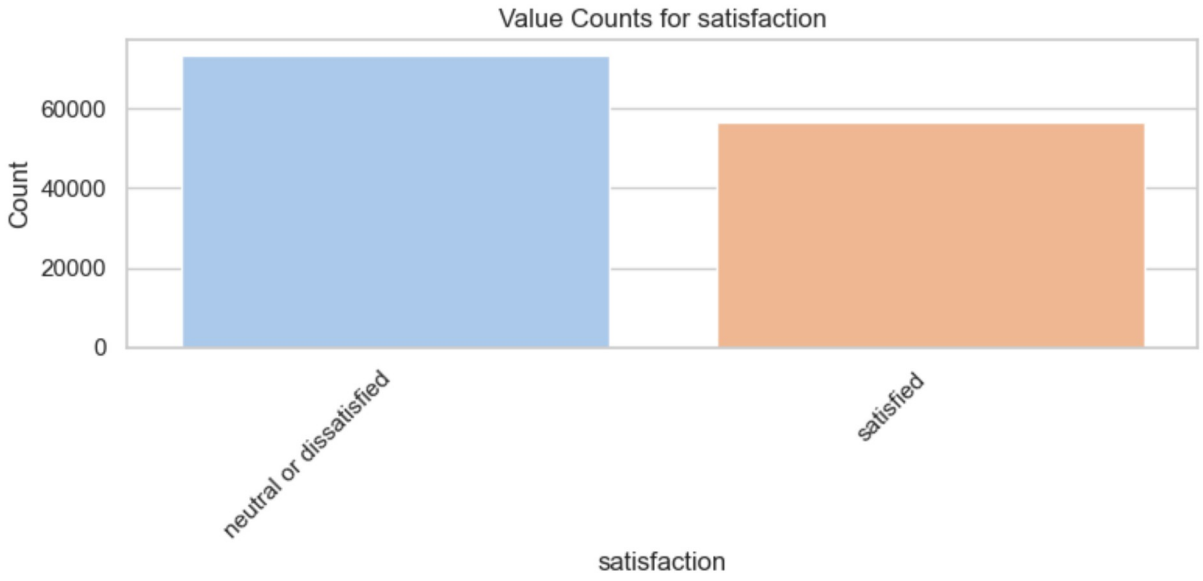

Cleanliness Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
Cleanliness

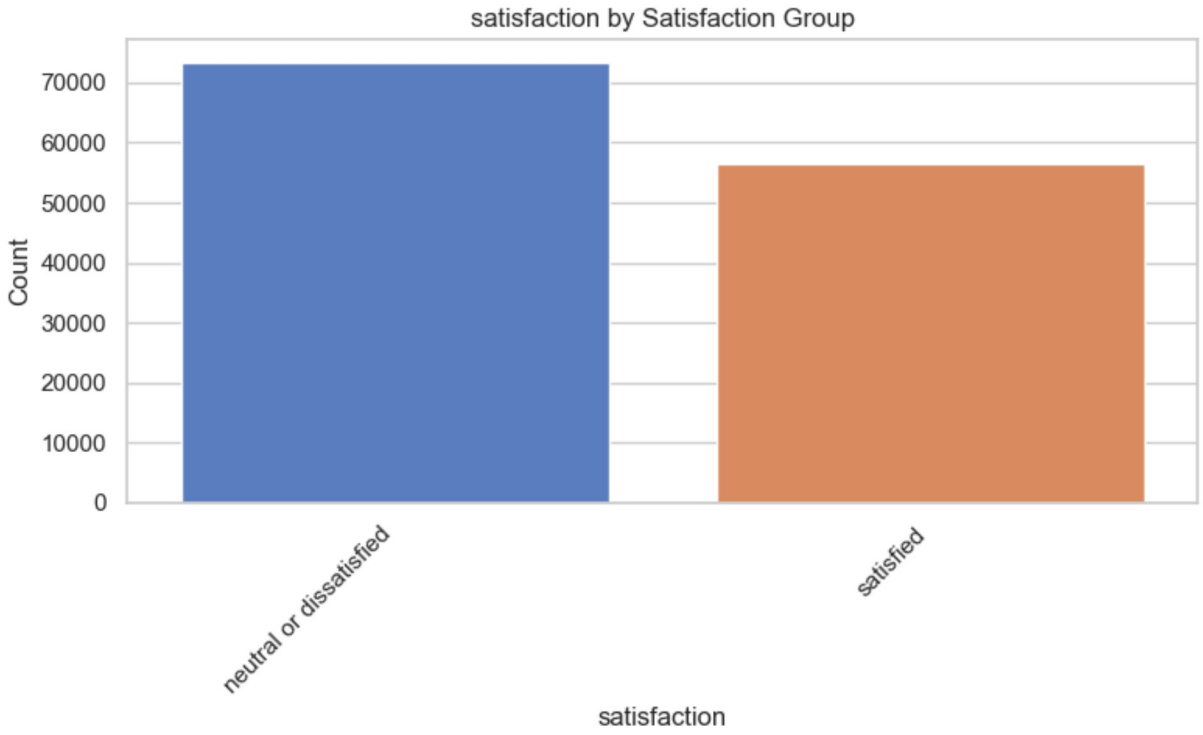

Overall Distribution of Departure_Delay_in_Minutes
Count
120000
100000
80000
60000
40000
20000
0
0
200
400
600
800
1000
1200
1400
1600
Departure_Delay_in_Minutes


Departure_Delay_in_Minutes Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
70000
60000
50000
40000
30000
20000
10000
0
0
200
400
600
800
1000
1200
1400
1600
Departure_Delay_in_Minutes


Overall Distribution of Arrival_Delay_in_Minutes
Count
120000
100000
80000
60000
40000
20000
0
0
200
400
600
800
1000
1200
1400
1600
Arrival_Delay_in_Minutes


Arrival_Delay_in_Minutes Distribution by Satisfaction Group
satisfaction
neutral or dissatisfied
satisfied
Count
60000
50000
40000
30000
20000
10000
0
0
200
400
600
800
1000
1200
1400
1600
Arrival_Delay_in_Minutes





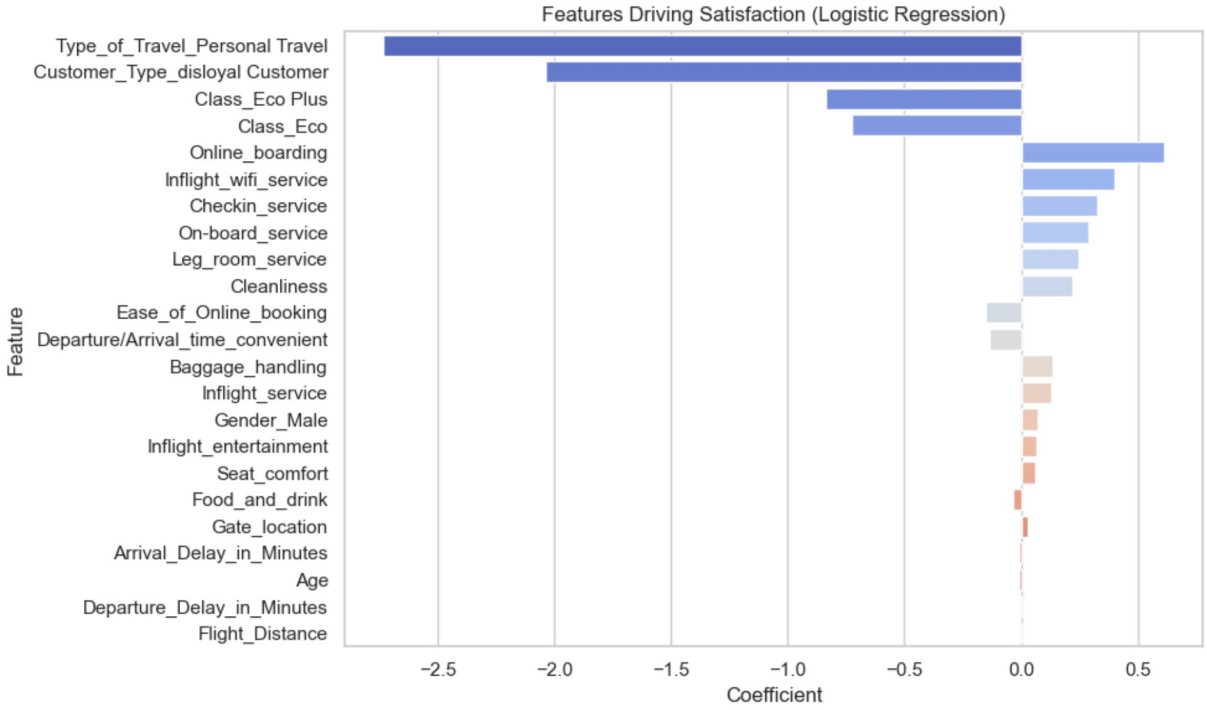
In [68]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv("AirlineSatisfaction_Transformed.csv")

#Drop rows with missing values (will only drop due to Arrival Delay in Minutes, for
```

```
df = df.dropna()

#Encode target variable for satisfaction; 1 for satisfied and 0 if not
df["satisfaction_binary"] = df["satisfaction"].apply(lambda x: 1 if x.strip().lower

#Build logistic regression model
X = df.drop(columns=["satisfaction", "satisfaction_binary"])
y = df["satisfaction_binary"]
categorical_cols = X.select_dtypes(include="object").columns.tolist()
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_c
        ('num', 'passthrough', numeric_cols)
    ]
)

logreg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=10000))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2

logreg_pipeline.fit(X_train, y_train)

#Also adding a section to output the actual statistics about the fitted model so we
#But looks like we can't get it from sklearn so instead have to use statsmodels for
import statsmodels.api as sm
X_train_encoded = preprocessor.fit_transform(X_train)

#Build DataFrame for statsmodels with column names
X_train_df = pd.DataFrame(
    X_train_encoded.toarray() if hasattr(X_train_encoded, "toarray") else X_train_e
    columns=all_feature_names
)

#Add intercept manually
X_train_df = sm.add_constant(X_train_df)
#Fit statsmodels logistic regression
sm_model = sm.Logit(y_train.values, X_train_df)
sm_result = sm_model.fit()

#Print detailed model statistical summary (coefficients and p values visible etc...
print(sm_result.summary())




#Get variable names again post encoding
ohe = logreg_pipeline.named_steps["preprocessor"].named_transformers_["cat"]
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([ohe_feature_names, numeric_cols])

#grab coefficients
```

```
coefficients = logreg_pipeline.named_steps["classifier"].coef_[0]
coef_df = pd.DataFrame({
    "Feature": all_feature_names,
    "Coefficient": coefficients
}).sort_values(by="Coefficient", key=np.abs, ascending=False)

#Plot the top 100 variables/features (so should just show all of them unless we exc
plt.figure(figsize=(10, 6))
sns.barplot(data=coef_df.head(100), x="Coefficient", y="Feature", palette="coolwarm
plt.title("Features Driving Satisfaction (Logistic Regression)")
plt.tight_layout()
plt.show()


#Now the only message left is a future warning so nbd
```

```
Optimization terminated successfully.
         Current function value: 0.334868
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:               103589
Model:                          Logit   Df Residuals:                   103565
Method:                           MLE   Df Model:                           23
Date:                Mon, 19 May 2025   Pseudo R-squ.:                  0.5108
Time:                        00:58:16   Log-Likelihood:                -34689.
converged:                       True   LL-Null:                       -70911.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
=================
                                       coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                                -5.7439      0.075    -76.552      0.000
-5.891      -5.597
Gender_Male                           0.0712      0.019      3.660      0.000
0.033       0.109
Customer_Type_disloyal Customer      -2.0405      0.030    -68.549      0.000
-2.099      -1.982
Type_of_Travel_Personal Travel       -2.7382      0.031    -87.034      0.000
-2.800      -2.676
Class_Eco                            -0.7223      0.026    -28.180      0.000
-0.773      -0.672
Class_Eco Plus                       -0.8364      0.041    -20.174      0.000
-0.918      -0.755
Age                                  -0.0087      0.001    -12.277      0.000
-0.010      -0.007
Flight_Distance                   -2.652e-05   1.13e-05     -2.343      0.019   -4.8
7e-05   -4.34e-06
Inflight_wifi_service                 0.4000      0.011     34.907      0.000
0.378       0.422
Departure/Arrival_time_convenient    -0.1349      0.008    -16.546      0.000
-0.151      -0.119
Ease_of_Online_booking               -0.1503      0.011    -13.299      0.000
-0.172      -0.128
Gate_location                         0.0277      0.009      3.030      0.002
0.010       0.046
Food_and_drink                       -0.0317      0.011     -2.954      0.003
-0.053      -0.011
Online_boarding                       0.6126      0.010     59.816      0.000
0.593       0.633
Seat_comfort                          0.0623      0.011      5.573      0.000
0.040       0.084
Inflight_entertainment                0.0680      0.014      4.758      0.000
0.040       0.096
On-board_service                      0.2916      0.010     28.665      0.000
0.272       0.312
Leg_room_service                      0.2473      0.009     29.079      0.000
0.231       0.264
Baggage_handling                      0.1354      0.011     11.840      0.000
0.113       0.158
```

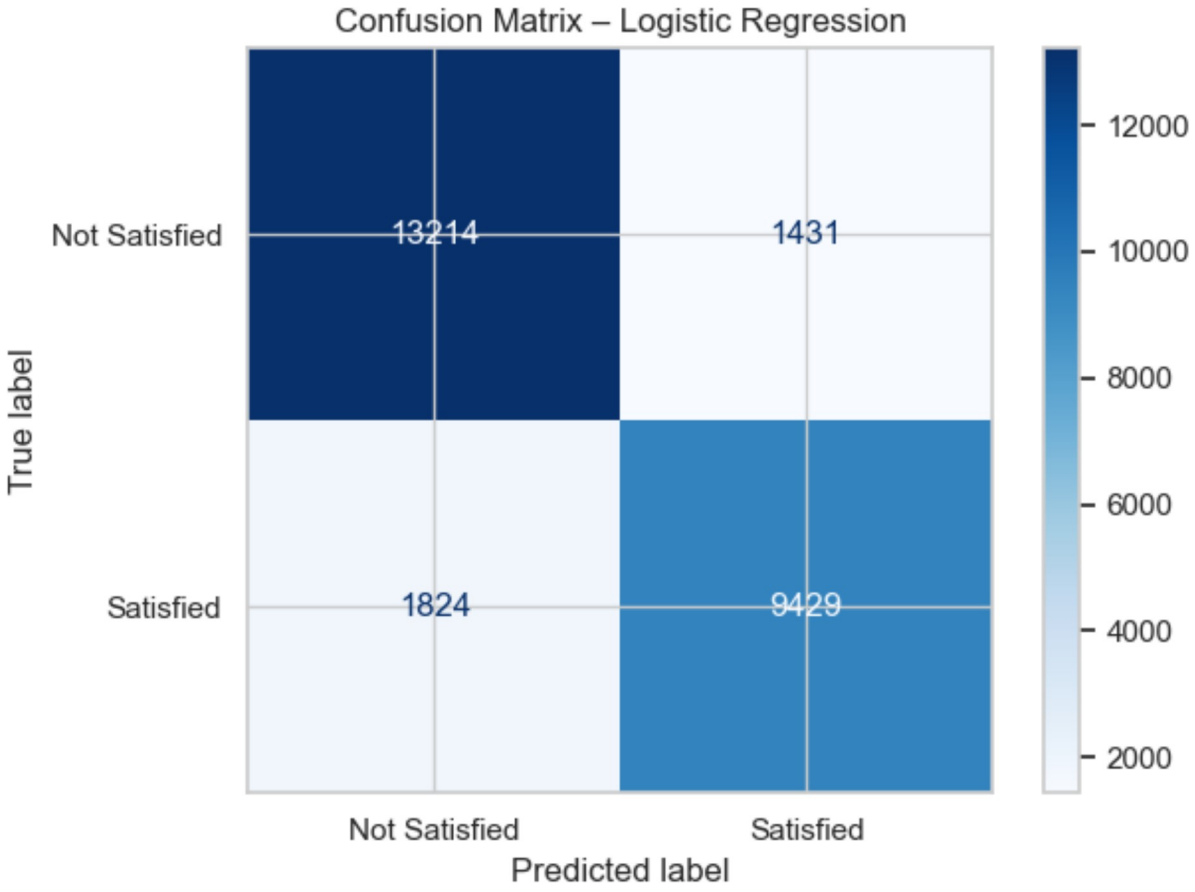
```
Checkin_service                      0.3304      0.009     38.645      0.000
0.314       0.347
Inflight_service                     0.1290      0.012     10.728      0.000
0.105       0.153
Cleanliness                          0.2191      0.012     18.040      0.000
0.195       0.243
Departure_Delay_in_Minutes           0.0044      0.001      4.422      0.000
0.002       0.006
Arrival_Delay_in_Minutes            -0.0095      0.001     -9.641      0.000
-0.011      -0.008
================================================================================
=================
C:\Users\guy74\AppData\Local\Temp\ipykernel_16280\2854533236.py:80: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=coef_df.head(100), x="Coefficient", y="Feature", palette="coolwar
m")
```



In [72]:
```python
#Finally test out how the "big model" is doing but we will likely want to trim this
#(to avoid overfitting but still try to retain as much explanatory power as possibl
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_scor

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

#Build out a confusion matrix and print the accuracy, precision, and recall
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Not Satisfied",

plt.figure(figsize=(6, 6))
disp.plot(cmap="Blues", values_format='d')
plt.title("Confusion Matrix – Logistic Regression")
```

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

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
```

```
<Figure size 600x600 with 0 Axes>
```




```
Accuracy: 0.8743146188894896
Precision: 0.868232044198895
Recall: 0.8379098906958145
```

In [ ]:
```
#As seen above, good performance on the unseen testing data still, but likely more
#More to come in future iterations
```

In [ ]:
```

```