In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Loading CSV file
df = pd.read_csv('raw dataset.csv')

# Basic exploration
print("Dataset shape:", df.shape)
```

```
Dataset shape: (103904, 25)
```

In [4]:
```
print("\nFirst 5 rows:")
df.head()
```

```
First 5 rows:
```

Out[4]:

	serial_number	id	gender	customer_type	age	type_of_travel	class	flight_
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	
1	1	5047	Male	disloyal Customer	25	Business travel	Business	
2	2	110028	Female	Loyal Customer	26	Business travel	Business	
3	3	24026	Female	Loyal Customer	25	Business travel	Business	
4	4	119299	Male	Loyal Customer	61	Business travel	Business	

5 rows × 25 columns

In [7]:
```
print("\nMissing values:")
df.isnull().sum()
```

```
Missing values:
```

Out[7]:
```
serial_number                          0
id                                     0
gender                                 0
customer_type                          0
age                                    0
type_of_travel                         0
class                                  0
flight_distance                        0
inflight_wifi_service                  0
departure_arrival_time_convenience     0
ease_of_online_booking                 0
gate_location                          0
food_and_drink                         0
online_boarding                        0
seat_comfort                           0
inflight_entertainment                 0
on_board_service                       0
leg_room_service                       0
baggage_handling                       0
checkin_service                        0
inflight_service                       0
cleanliness                            0
departure_delay_in_minutes             0
arrival_delay_in_minutes             310
satisfaction                           0
dtype: int64
```

In [6]:
```
print("\nData types:")
df.dtypes
```

```
Data types:
```

Out[6]:
```
serial_number                          int64
id                                     int64
gender                                object
customer_type                         object
age                                    int64
type_of_travel                        object
class                                 object
flight_distance                        int64
inflight_wifi_service                  int64
departure_arrival_time_convenience     int64
ease_of_online_booking                 int64
gate_location                          int64
food_and_drink                         int64
online_boarding                        int64
seat_comfort                           int64
inflight_entertainment                 int64
on_board_service                       int64
leg_room_service                       int64
baggage_handling                       int64
checkin_service                        int64
inflight_service                       int64
cleanliness                            int64
departure_delay_in_minutes             int64
arrival_delay_in_minutes             float64
satisfaction                          object
dtype: object
```

In [10]:
```
print("\nBasic statistics:")
```

```
df.describe()
```

Basic statistics:

Out[10]:

	serial_number	id	age	flight_distance	inflight_wifi_service
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000
mean	51951.500000	64924.210502	39.379706	1189.448375	2.729683
std	29994.645522	37463.812252	15.114964	997.147281	1.327829
min	0.000000	1.000000	7.000000	31.000000	0.000000
25%	25975.750000	32533.750000	27.000000	414.000000	2.000000
50%	51951.500000	64856.500000	40.000000	843.000000	3.000000
75%	77927.250000	97368.250000	51.000000	1743.000000	4.000000
max	103903.000000	129880.000000	85.000000	4983.000000	5.000000

In [11]:
```
df = df.drop('serial_number', axis=1)
```

In [12]:
```
df.describe()
```

Out[12]:

	id	age	flight_distance	inflight_wifi_service	departure_arri
count	103904.000000	103904.000000	103904.000000	103904.000000	
mean	64924.210502	39.379706	1189.448375	2.729683	
std	37463.812252	15.114964	997.147281	1.327829	
min	1.000000	7.000000	31.000000	0.000000	
25%	32533.750000	27.000000	414.000000	2.000000	
50%	64856.500000	40.000000	843.000000	3.000000	
75%	97368.250000	51.000000	1743.000000	4.000000	
max	129880.000000	85.000000	4983.000000	5.000000	

In [13]:
```
df['satisfaction'] = df['satisfaction'].astype(str)
```

In [15]:
```
df.dtypes
```

Out[15]:
```
id                                      int64
gender                                 object
customer_type                          object
age                                     int64
type_of_travel                         object
class                                  object
flight_distance                         int64
inflight_wifi_service                   int64
departure_arrival_time_convenience      int64
ease_of_online_booking                  int64
gate_location                           int64
food_and_drink                          int64
online_boarding                         int64
seat_comfort                            int64
inflight_entertainment                  int64
on_board_service                        int64
leg_room_service                        int64
baggage_handling                        int64
checkin_service                         int64
inflight_service                        int64
cleanliness                             int64
departure_delay_in_minutes              int64
arrival_delay_in_minutes              float64
satisfaction                           object
dtype: object
```

In [18]:
```
df['arrival_delay_in_minutes'] = pd.to_numeric(df['arrival_delay_in_minutes'], e
```

In [22]:
```
df.head()
```

Out[22]:

	id	gender	customer_type	age	type_of_travel	class	flight_distance	inflig
0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	
1	5047	Male	disloyal Customer	25	Business travel	Business	235	
2	110028	Female	Loyal Customer	26	Business travel	Business	1142	
3	24026	Female	Loyal Customer	25	Business travel	Business	562	
4	119299	Male	Loyal Customer	61	Business travel	Business	214	

5 rows × 24 columns

In [23]:
```
df['gender'] = df['gender'].str[0].str.upper()
df['customer_type'] = df['customer_type'].str.lower().str.extract(r'(loyal|dislo
df['class'] = df['class'].str.lower()
```

In [25]:
```
df.head()
```

Out[25]:

	id	gender	customer_type	age	type_of_travel	class	flight_distance	inflig
0	70172	M	loyal	13	Personal Travel	eco plus	460	
1	5047	M	disloyal	25	Business travel	business	235	
2	110028	F	loyal	26	Business travel	business	1142	
3	24026	F	loyal	25	Business travel	business	562	
4	119299	M	loyal	61	Business travel	business	214	

5 rows × 24 columns

In [26]:
```
df['type_of_travel'] = df['type_of_travel'].str.lower().str.replace(' travel', '
```

In [27]:
```
df.head()
```

Out[27]:

	id	gender	customer_type	age	type_of_travel	class	flight_distance	inflig
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 24 columns

In [28]:
```
df = df.rename(columns={'id': 'customer_id'})
```

In [29]:
```
df.head()
```

Out[29]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 24 columns

In [30]:
```
initial_count = len(df)
df = df.drop_duplicates(subset='customer_id')
removed_count = initial_count - len(df)
print(f"Removed {removed_count} duplicate records")
```

```
Removed 0 duplicate records
```

In [31]:
```
df.describe()
```

Out[31]:

	customer_id	age	flight_distance	inflight_wifi_service	departure_arri
count	103904.000000	103904.000000	103904.000000	103904.000000	
mean	64924.210502	39.379706	1189.448375	2.729683	
std	37463.812252	15.114964	997.147281	1.327829	
min	1.000000	7.000000	31.000000	0.000000	
25%	32533.750000	27.000000	414.000000	2.000000	
50%	64856.500000	40.000000	843.000000	3.000000	
75%	97368.250000	51.000000	1743.000000	4.000000	
max	129880.000000	85.000000	4983.000000	5.000000	

In [33]:
```
## code to replace all product or service 0 ratings with NaN
service_columns = [
    'inflight_wifi_service', 'departure_arrival_time_convenience',
    'ease_of_online_booking', 'gate_location', 'food_and_drink',
    'online_boarding', 'seat_comfort', 'inflight_entertainment',
    'on_board_service', 'leg_room_service', 'baggage_handling',
    'checkin_service', 'inflight_service', 'cleanliness'
]
df[service_columns] = df[service_columns].replace(0, np.nan)
```

Out[33]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 24 columns

In [35]:
```
df.describe()
```

Out[35]:

	customer_id	age	flight_distance	inflight_wifi_service	departure_arri
count	103904.000000	103904.000000	103904.000000	100801.000000	
mean	64924.210502	39.379706	1189.448375	2.813712	
std	37463.812252	15.114964	997.147281	1.257367	
min	1.000000	7.000000	31.000000	1.000000	
25%	32533.750000	27.000000	414.000000	2.000000	
50%	64856.500000	40.000000	843.000000	3.000000	
75%	97368.250000	51.000000	1743.000000	4.000000	
max	129880.000000	85.000000	4983.000000	5.000000	

In [36]:
```
## code to remove all rows with any one cell blank or null (other than NaN) and

# Remove rows with any empty/blank/null values
df_blank_issues = df[df.isnull().any(axis=1) | (df == '').any(axis=1)]
df_clean = df.drop(df_blank_issues.index)

# Save problematic rows to separate file
df_blank_issues.to_csv('airline_data_blank_data_for_review.csv', index=False)
print(f"Found {len(df_blank_issues)} rows with blank/null values")
```

```
Found 8200 rows with blank/null values
```

In [ ]:
```
## analysis of blank rows and how it can effect business and why it is necessary
```

In [37]:
```
## code to remove all impractical flight distance but there are none
df.describe()
```

Out[37]:

	customer_id	age	flight_distance	inflight_wifi_service	departure_arri
count	103904.000000	103904.000000	103904.000000	100801.000000	
mean	64924.210502	39.379706	1189.448375	2.813712	
std	37463.812252	15.114964	997.147281	1.257367	
min	1.000000	7.000000	31.000000	1.000000	
25%	32533.750000	27.000000	414.000000	2.000000	
50%	64856.500000	40.000000	843.000000	3.000000	
75%	97368.250000	51.000000	1743.000000	4.000000	
max	129880.000000	85.000000	4983.000000	5.000000	

In [38]:
```
## code to remove all impractical departure delay in minutes
initial_rows = len(df)

# Identify impractical departure delays (>6 hours = 360 minutes)
impractical_mask = df['departure_delay_in_minutes'] > 360
df_delay_issues = df[impractical_mask].copy()
df = df[~impractical_mask].copy()

current_rows = len(df)

print(f"Rows before removal: {initial_rows}")
print(f"Rows after removal: {current_rows}")
print(f"Removed {len(df_delay_issues)} rows with impractical departure delays (>

# Save problematic rows
df_delay_issues.to_csv('airline_impractical_departure_delay_issues_for_review.cs
```

```
Rows before removal: 103904
Rows after removal: 103785
Removed 119 rows with impractical departure delays (>6 hours)
```

In [38]:
```
## analysis of all removed impractical arrival delay in minutes
```

```
Rows before removal: 103904
Rows after removal: 103785
Removed 119 rows with impractical departure delays (>6 hours)
```

In [39]:
```
## code to remove all impractical arrival delay in minutes

initial_arrival_rows = len(df)

arrival_impractical_mask = df['arrival_delay_in_minutes'] > 30
df_arrival_issues = df[arrival_impractical_mask].copy()
df = df[~arrival_impractical_mask].copy()

current_arrival_rows = len(df)

print(f"Rows before arrival delay removal: {initial_arrival_rows}")
print(f"Rows after arrival delay removal: {current_arrival_rows}")
print(f"Removed {len(df_arrival_issues)} rows with impractical arrival delays (>

df_arrival_issues.to_csv('airline_impractical_arrival_delay_issues_for_review.cs
```

```
Rows before arrival delay removal: 103785
Rows after arrival delay removal: 89240
Removed 14545 rows with impractical arrival delays (>30 minutes)
```

In [ ]:
```
## analysis of all removed impractical arrival delay in minutes
```

In [40]:
```
## code to create a new column Time_Recovered = departure_delay_in_minutes - arr
df['Time_Recovered'] = df['departure_delay_in_minutes'] - df['arrival_delay_in_m
```

In [41]:
```
df.head()
```

Out[41]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 25 columns

In [42]:
```
# Move Time_Recovered column before satisfaction
cols = df.columns.tolist()
time_idx = cols.index('Time_Recovered')
sat_idx = cols.index('satisfaction')

cols.insert(sat_idx, cols.pop(time_idx))
df = df[cols]
```

In [43]:
```
df.head()
```

Out[43]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 25 columns

In [44]:
```
## Time recovery status
df['time_recovery_status'] = df['Time_Recovered'].apply(
```

```
    lambda x: 'Recovered' if x > 0 else ('Same' if x == 0 else 'Worsened')
)
```

In [45]:
```
df.head()
```

Out[45]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 26 columns

In [48]:
```
cols = df.columns.tolist()
recovery_idx = cols.index('time_recovery_status')
sat_idx = cols.index('satisfaction')

cols.insert(sat_idx, cols.pop(recovery_idx))
df = df[cols]
```

In [49]:
```
df.head()
```

Out[49]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 26 columns

In [50]:
```
df = df.rename(columns={'Time_Recovered': 'time_recovered'})
```

In [51]:
```
df['time_recovery_status'] = df['time_recovery_status'].str.lower()
```

In [52]:
```
df.head()
```

Out[52]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 26 columns

In [53]:
```
df = df.rename(columns={'time_recovered': 'time_recovered_in_minutes'})
```

In [54]:
```
df.head()
```

Out[54]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 26 columns

In [55]:
```
## code to remove all impractical time_recovered values

initial_rows = len(df)

# Identify time_recovered values >60 or <-60
time_issue_mask = df['time_recovered_in_minutes'].abs() > 60
df_time_issues = df[time_issue_mask].copy()
df = df[~time_issue_mask].copy()

current_rows = len(df)

print(f"Rows before removal: {initial_rows}")
print(f"Rows after removal: {current_rows}")
print(f"Removed {len(df_time_issues)} rows with [time_recovered_in_minutes] > 60

df_time_issues.to_csv('airline_impractical_time_recovered_in_minutes_issues_for_
```

```
Rows before removal: 89240
Rows after removal: 89174
Removed 66 rows with |time_recovered| > 60 minutes
```

In [56]:
```
df.head()
```

Out[56]:

	customer_id	gender	customer_type	age	type_of_travel	class	flight_distance	i
0	70172	M	loyal	13	personal	eco plus	460	
1	5047	M	disloyal	25	business	business	235	
2	110028	F	loyal	26	business	business	1142	
3	24026	F	loyal	25	business	business	562	
4	119299	M	loyal	61	business	business	214	

5 rows × 26 columns

In [57]:
```
df.to_csv('refined_dataset.csv', index=False)
```

In [ ]:
```
## code to divide ratings into product and services
```