Virgin Atlantic Reviews Notebook 1: Web Scraping and Cleaning

The data for this notebook was obtained from Skytrax.com. The robots.txt page indicates that scraping is ok.

In [18]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
```

Test scrape of one page

In [15]:
```
# Test url we will use to try the code
url = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/1/?s
```

In [19]:
```
# Send a GET request to the URL - uncomment code to run
response = requests.get(url)

# Raise an error if the response status code is not 200 (OK)
response.raise_for_status()

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')
```

In [20]:
```
# We need to extract a range of things from the soup object including titles, review te

# titles, reviews and dates
titles = []
review_texts = []
dates = []

# stars and categories info
stars = []
cats_interest = [] # categories of interest
other_cats = [] # other categories
other_cats_values = []

# Iterate over the number of reviews in this first url
for i in range(0,100):

    # Extract the details we want
    dets = soup.find_all('div', class_ = "body")[i]
    review_title = dets.find('h2')
    review_text = dets.find('div', class_='text_content', itemprop='reviewBody')
    review_date = dets.find('time', itemprop='datePublished')
    review_cats = dets.find_all('td', class_="review-rating-header")
    review_stars = dets.find_all('span', class_="star")
    review_values = dets.find_all('td',class_ = 'review-value')
```

```
    # Append to the lists
    titles.append(review_title.get_text(strip= True))
    review_texts.append(review_text.get_text(strip= True))
    dates.append(review_date.get_text(strip= True))
    stars.append(review_stars)
    for c in review_values:
        v = c.get_text(strip = True)
        other_cats_values.append(v)

    # For the categories we only want certain ones with the star ratings attached
    for i in review_cats:
        g = i.get_text(strip = True)
        cats_of_interest = ['Seat Comfort','Cabin Staff Service','Food & Beverages','In
                    'Wifi & Connectivity','Value For Money']
        cats_not_interest = ['Type of Traveller','Seat Type','Route','Date Flown','Airc

        if g in cats_of_interest:
            cats_interest.append(g)
        else:
            other_cats.append(g)

    # Extract the seat class information from the extra details we collected
    seats_classes = ['Economy Class', 'First Class', 'Premium Economy','Business Class'
    seats = [item for item in other_cats_values if any(x in item for x in seats_classes

    # Extract destinations from the details as strings that contain the word 'to' as a
    destinations = [s for s in other_cats_values if re.search(r'\bto\b', s)]

    # Create dataframe of what we have so far
    final_test = pd.DataFrame(zip(dates, titles, review_texts,seats, destinations))

    # Name the columns
    final_test.columns = ['Date','Review Title', 'Review Text','Seat Class','Route']
```

In [21]:

```
final_test.head()
```

Out[21]:

	Date	Review Title	Review Text	Seat Class	Route
0	23rd June 2025	"Screen monitor is old"	✅Trip Verified| Maybe because of the route, ...	Economy Class	London to Mumbai
1	23rd June 2025	"Seat for economy was comfortable"	✅Trip Verified| Food was decent. Seat for ec...	Economy Class	Miami to London
2	11th June 2025	“a great experience”	✅Trip Verified| First time with Virgin in over...	Premium Economy	Manchester to New York JFK
3	23rd May 2025	"most people wanted to just sleep"	✅Trip Verified| Really good experience in Pr...	Premium Economy	New York JFK to London
4	11th May 2025	"Good selection of movies"	✅Trip Verified| Premium economy had its own ...	Premium Economy	London Heathrow to Orlando

In [22]:

```
# Stripping out characters and words we don't want
final_test['Review Text'] = final_test['Review Text'].str.replace('Trip Verified|', '')
final_test['Review Text'] = final_test['Review Text'].str.replace('Not Verified|', '')
final_test['Review Text'] = final_test['Review Text'].str.replace('|', '')
```

```
final_test['Review Text'] = final_test['Review Text'].str.replace('✅', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('"', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('“', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('”', '')
```

```
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:2:
FutureWarning: The default value of regex will change from True to False in a future ver
sion.

C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:3:
FutureWarning: The default value of regex will change from True to False in a future ver
sion.
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:4:
FutureWarning: The default value of regex will change from True to False in a future ver
sion. In addition, single character regular expressions will*not* be treated as literal
strings when regex=True.
  after removing the cwd from sys.path.
```

In [23]:

```
# Split the 'Route' column into two new columns
final_test[['From', 'To']] = final_test['Route'].str.split(' to ', expand=True)

# Drop the original 'Route' column
final_test.drop(columns=['Route'], inplace=True)

# Now split the 'To' column into two new columns which splits out the 'via' details
final_test[['To', 'Via']] = final_test['To'].str.split(' via ', expand=True)

# Drop this 'Via' column
final_test.drop(columns=['Via'], inplace=True)

final_test.head()
```

Out[23]:

	Date	Review Title	Review Text	Seat Class	From	To
0	23rd June 2025	Screen monitor is old	Maybe because of the route, food did not ca...	Economy Class	London	Mumbai
1	23rd June 2025	Seat for economy was comfortable	Food was decent. Seat for economy was comfo...	Economy Class	Miami	London
2	11th June 2025	a great experience	First time with Virgin in over a decade but f...	Premium Economy	Manchester	New York JFK
3	23rd May 2025	most people wanted to just sleep	Really good experience in Premium Economy. ...	Premium Economy	New York JFK	London
4	11th May 2025	Good selection of movies	Premium economy had its own dedicated check...	Premium Economy	London Heathrow	Orlando

Now we need to deal with all the star ratings information, as we would like to append this to the dataframe, such that for each review, we also have the star rating for the categories of seat comfort, food and beverages etc.

The number of categories rated per review is not set, some reviews have seven categories and some only two or three. Added to this each category rated has five circles which are filled to represent the

rating given. We will try to sort this out with a series of steps. We have a list of the star ratings and a list of the categories of interest determined in the code block above. We can use the information in the star ratings to split up the categories per review.

In [60]:
```
# this list will contain a series of numbers representing the number of categories that
chunk_sizes = []

for i in stars: # the stars list is already split according to the number of categories
    l = len(i)/5 # there are five circles per rating so we divide the total by this to g
    chunk_sizes.append(l)

chunk_sizes = [int(item) for item in chunk_sizes] # Convert to integers
chunk_sizes[0:5] # shows the number of categories rated for the first five reviews
```

Out[60]:
```
[5, 5, 6, 6, 6]
```

In [71]:
```
# Now we split the categories list according to the chunk sizes we just found - in effe
# the number of categories per review and the names of those categories (seat comfort e

# Function to split the categories
def split_list(string_list, lengths):
    result = []
    start_index = 0

    for length in lengths:
        # Slice the string_list from start_index to start_index + length
        sublist = string_list[start_index:start_index + length]
        result.append(sublist)
        start_index += length

    return result
```

In [76]:
```
# Run the function and check the first entry in the list has the right number of categor
cats_interest_split = split_list(cats_interest, chunk_sizes)
print(chunk_sizes[0])
cats_interest_split[0]
```

```
5
```

Out[76]:
```
['Seat Comfort',
 'Cabin Staff Service',
 'Food & Beverages',
 'Ground Service',
 'Value For Money']
```

In [77]:
```
# Check another one
print(chunk_sizes[2])
cats_interest_split[2]
```

```
6
```

Out[77]:
```
['Seat Comfort',
 'Cabin Staff Service',
 'Food & Beverages',
 'Inflight Entertainment',
 'Ground Service',
 'Value For Money']
```

We now have the categories split according to the chunks. We now have to deal with the star ratings by converting the words 'star filled' and 'star' to digits

In [79]:
```
# Flatten out the stars ratings give a '1' if the star is filled and '0' if not
stars_flat = [x for xs in stars for x in xs]

star_fills = []

for i in stars_flat:
    r = i['class']
    if "fill" in r:
        star_fills.append(1)
    else:
        star_fills.append(0)
```

In [83]:
```
# Split the star fills list into chunks on every fifth entry for the circles per rating
chunks_stars_split = [star_fills[i:i + 5] for i in range(0, len(star_fills), 5)]
chunks_stars_split[0:5]
```

Out[83]:
```
[[1, 1, 1, 0, 0],
 [1, 1, 1, 0, 0],
 [1, 1, 0, 0, 0],
 [1, 1, 1, 1, 0],
 [1, 1, 1, 1, 0]]
```

This is now a list of lists that shows the stars as numbers, so 1,1,1,0,0 indicates a 3 star for the first category

In [97]:
```
# Get a single value for these lists by summing the ones and zeros
star_totals = []

for i in chunks_stars_split:
        s = sum(i)
        star_totals.append(s)

# List of lists showing the value of the stars given for each category in each review
results = split_list(star_totals, chunk_sizes)

results[0:5]
```

Out[97]:
```
[[3, 3, 2, 4, 4],
 [5, 5, 4, 4, 4],
 [4, 5, 5, 3, 5, 5],
 [5, 5, 5, 5, 2, 5],
 [5, 4, 5, 5, 5, 5]]
```

In [99]:
```
# Create a list of dictionaries with the categories and the star rating values
dict_list = []

for i in range(len(cats_interest_split)):
    dictionary = dict(zip(cats_interest_split[i], results[i]))
    dict_list.append(dictionary)

# Create DataFrame from the list of dictionaries
df_stars = pd.DataFrame(dict_list)
```

```
# Display the DataFrame
df_stars.head()
```

Out[99]:

	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service	Value For Money	Inflight Entertainment	Wifi & Connectivity
0	3.0	3.0	2.0	4.0	4	NaN	NaN
1	5.0	5.0	4.0	4.0	4	NaN	NaN
2	4.0	5.0	5.0	5.0	5	3.0	NaN
3	5.0	5.0	5.0	2.0	5	5.0	NaN
4	5.0	4.0	5.0	5.0	5	5.0	NaN

In [100…

```
# Add the data to our final dataframe
final_test = pd.concat([final_test, df_stars], axis = 1)
final_test.head()
```

Out[100…

	Date	Review Title	Review Text	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service
0	23rd June 2025	Screen monitor is old	Maybe because of the route, food did not ca...	Economy Class	London	Mumbai	3.0	3.0	2.0	4.0
1	23rd June 2025	Seat for economy was comfortable	Food was decent. Seat for economy was comfo...	Economy Class	Miami	London	5.0	5.0	4.0	4.0
2	11th June 2025	a great experience	First time with Virgin in over a decade but f...	Premium Economy	Manchester	New York JFK	4.0	5.0	5.0	5.0
3	23rd May 2025	most people wanted to just sleep	Really good experience in Premium Economy. ...	Premium Economy	New York JFK	London	5.0	5.0	5.0	2.0

	Date	Review Title	Review Text	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service
4	11th May 2025	Good selection of movies	Premium economy had its own dedicated check...	Premium Economy	London Heathrow	Orlando	5.0	4.0	5.0	5.0

In [101…

```
# Select the columns that have nan values
cols = final_test.columns[6:]

# Replace nan values with 0 in the table and set to integers
final_test[cols] = final_test[cols].fillna(value=0)
final_test[cols] = final_test[cols].astype(int)

final_test.head()
```

Out[101…

	Date	Review Title	Review Text	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service
0	23rd June 2025	Screen monitor is old	Maybe because of the route, food did not ca...	Economy Class	London	Mumbai	3	3	2	4
1	23rd June 2025	Seat for economy was comfortable	Food was decent. Seat for economy was comfo...	Economy Class	Miami	London	5	5	4	4
2	11th June 2025	a great experience	First time with Virgin in over a decade but f...	Premium Economy	Manchester	New York JFK	4	5	5	5
3	23rd May 2025	most people wanted to just sleep	Really good experience in Premium Economy. ...	Premium Economy	New York JFK	London	5	5	5	2
4	11th May 2025	Good selection of movies	Premium economy had its own	Premium Economy	London Heathrow	Orlando	5	4	5	5

Date	Review Title	Review Text	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food & Beverages	Ground Service
		dedicated check...							

We now have a process we can use against more pages of the website. We can create a function to run this and apply it as required to each page

---

Scrape of all of the pages we want

In [2]:
```
# Note: we are listing them separately as the code didn't work properly using a for loo
url1 = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/1/?
url2 = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/2/?
url3 = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/3/?
url4 = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/4/?
url5 = 'https://www.airlinequality.com/airline-reviews/virgin-atlantic-airways/page/5/?
```

In [3]:
```
# Function to fetch page details
def fetch_page_details(url):
    try:
        # Send a GET request to the URL
        response = requests.get(url)
        # Raise an error if the response status code is not 200 (OK)
        response.raise_for_status()

        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

    except requests.RequestException as e:
        print(f"Error fetching {url}: {e}")

    return soup
```

In [4]:
```
# Return each soup object (done separately as errors when appended to list)
soup1 = fetch_page_details(url1)
soup2 = fetch_page_details(url2)
soup3 = fetch_page_details(url3)
soup4 = fetch_page_details(url4)
soup5 = fetch_page_details(url5)
```

In [1]:
```
#soup1
```

In [112…
```
def process_soups(soup):

    # set up empty lists for titles, reviews and dates
    titles = []
    review_texts = []
```

```
dates = []

# set up empty lists for stars and categories info
stars = []
cats_interest = [] # categories of interest
other_cats = [] # other categories
other_cats_values = []

# Iterate over the number of reviews in each url
for i in range(0,100):

    # Extract the details we want
    dets = soup.find_all('div', class_ = "body")[i]
    review_title = dets.find('h2')
    review_text = dets.find('div', class_='text_content', itemprop='reviewBody')
    review_date = dets.find('time', itemprop='datePublished')
    review_cats = dets.find_all('td', class_="review-rating-header")
    review_stars = dets.find_all('span', class_="star")
    review_values = dets.find_all('td',class_ = 'review-value')

    # Append the extracted details to the empty lists
    titles.append(review_title.get_text(strip= True))
    review_texts.append(review_text.get_text(strip= True))
    dates.append(review_date.get_text(strip= True))
    stars.append(review_stars)
    for c in review_values:
        v = c.get_text(strip = True)
        other_cats_values.append(v)

    # For the categories we only want certain ones with the star ratings attached, s
    for i in review_cats:
        g = i.get_text(strip = True)
        cats_of_interest = ['Seat Comfort','Cabin Staff Service','Food & Beverages'
                    'Wifi & Connectivity','Value For Money']
        cats_not_interest = ['Type of Traveller','Seat Type','Route','Date Flown','A

        if g in cats_of_interest:
            cats_interest.append(g)
        else:
            other_cats.append(g)

    # Extract the seat class information from the 'other cats' details we collected
    seats_classes = ['Economy Class', 'First Class', 'Premium Economy','Business Cl
    seats = [item for item in other_cats_values if any(x in item for x in seats_cla

    # Extract destinations from the 'other cats' details as strings that contain th
    destinations = [s for s in other_cats_values if re.search(r'\bto\b', s)]

    # Extract date flown
    date_flown = [f for f in other_cats_values if re.search(r'[2][0][0-2][0-9]',f)]

    # Determine the maximum length of the titles column
    max_length = len(titles)

    # Extend the destinations and date flown lists to the maximum length
    destinations.extend([np.nan] * (max_length - len(destinations)))
    date_flown.extend([np.nan] * (max_length - len(date_flown)))

# Create dataframe of what we have so far
final_test = pd.DataFrame(zip(dates, titles, review_texts, date_flown, seats, desti
```

```
# Name the columns
final_test.columns = ['Review Date','Review Title', 'Review Text','Flight Date','Se

# Stripping out characters and words we don't want
final_test['Review Text'] = final_test['Review Text'].str.replace('Trip Verified|',
final_test['Review Text'] = final_test['Review Text'].str.replace('Not Verified|',
final_test['Review Text'] = final_test['Review Text'].str.replace('|', '')
final_test['Review Text'] = final_test['Review Text'].str.replace('✅', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('"', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('“', '')
final_test['Review Title'] = final_test['Review Title'].str.replace('”', '')

# Split the 'Route' column into two new columns
final_test[['From', 'To']] = final_test['Route'].str.split(' to ', expand=True)
final_test.drop(columns=['Route'], inplace=True)
final_test[['To', 'Via']] = final_test['To'].str.split(' via ', expand=True)
final_test.drop(columns=['Via'], inplace=True)

# Now work on the star ratings

# Get the chunk sizes (number of categories rated per review)
chunk_sizes = []
for i in stars:
    l = len(i)/5
    chunk_sizes.append(l)

chunk_sizes = [int(item) for item in chunk_sizes]

# Function to split the scraped categories to match the chunk sizes
def split_list(string_list, lengths):
    result = []
    start_index = 0

    for length in lengths:
    #Slice the string_list from start_index to start_index + length
        sublist = string_list[start_index:start_index + length]
        result.append(sublist)
        start_index += length

    return result

# Function to split the categories
def split_list(string_list, lengths):
    result = []
    start_index = 0

    for length in lengths:
    # Slice the string_list from start_index to start_index + length
        sublist = string_list[start_index:start_index + length]
        result.append(sublist)
        start_index += length

    return result

# Run the function to split categories
cats_interest_split = split_list(cats_interest, chunk_sizes)

# Get the scraped stars information and if the field is filled give it a '1' or else
stars_flat = [x for xs in stars for x in xs]
```

```
    star_fills = []

    for i in stars_flat:
        r = i['class']
        if "fill" in r:
            star_fills.append(1)
        else:
            star_fills.append(0)

    # Split the star fills list into chunks on every fifth entry for the circles per ra
    chunks_stars = [star_fills[i:i + 5] for i in range(0, len(star_fills), 5)]

    # Get a single value for these lists by summing the ones and zeros
    star_totals = []
    for i in chunks_stars:
        s = sum(i)
        star_totals.append(s)

    results = split_list(star_totals, chunk_sizes)

    # Create a list of dictionaries with the categories and the star rating values
    dict_list = []

    for i in range(len(cats_interest_split)):
        dictionary = dict(zip(cats_interest_split[i], results[i]))
        dict_list.append(dictionary)

    # Create DataFrame from the list of dictionaries
    df_stars = pd.DataFrame(dict_list)
    final_test = pd.concat([final_test, df_stars], axis = 1)

    # Fill nan values with zero, set values to integer and dates to datetime
    cols = final_test.columns[7:]
    final_test[cols] = final_test[cols].fillna(value=0)
    final_test[cols] = final_test[cols].astype(int)
    final_test['Review Date'] = pd.to_datetime(final_test['Review Date'])
    final_test['Flight Date'] = pd.to_datetime(final_test['Flight Date'])
    final_test['Flight Date'] = final_test['Flight Date'].fillna(value = 'Not available

    return final_test
```

In [113…

```
# Combine into dataframe
df1 = process_soups(soup1)
df2 = process_soups(soup2)
df3 = process_soups(soup3)
df4 = process_soups(soup4)
df5 = process_soups(soup5)
all_results = pd.concat([df1, df2, df3, df4, df5])
all_results.shape
```

```
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:7
1: FutureWarning: The default value of regex will change from True to False in a future
version.
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:7
2: FutureWarning: The default value of regex will change from True to False in a future
version.
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\ipykernel_launcher.py:7
3: FutureWarning: The default value of regex will change from True to False in a future
```

```
version. In addition, single character regular expressions will*not* be treated as liter
al strings when regex=True.
```

Out[113…

```
(500, 14)
```

In [114…

```
all_results.head()
```

Out[114…

	Review Date	Review Title	Review Text	Flight Date	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food & Beverages
0	2025-06-23	Screen monitor is old	Maybe because of the route, food did not ca...	2025-06-01 00:00:00	Economy Class	London	Mumbai	3	3	2
1	2025-06-23	Seat for economy was comfortable	Food was decent. Seat for economy was comfo...	2025-06-01 00:00:00	Economy Class	Miami	London	5	5	4
2	2025-06-11	a great experience	First time with Virgin in over a decade but f...	2025-06-01 00:00:00	Premium Economy	Manchester	New York JFK	4	5	5
3	2025-05-23	most people wanted to just sleep	Really good experience in Premium Economy. ...	2025-05-01 00:00:00	Premium Economy	New York JFK	London	5	5	5
4	2025-05-11	Good selection of movies	Premium economy had its own dedicated check...	2025-05-01 00:00:00	Premium Economy	London Heathrow	Orlando	5	4	5

Cleaning the route details

In [138…

```
reviews = all_results.copy()
```

In [139…

```
# Groupby the departure location
reviews.groupby('From', as_index = False)['Review Date'].count().sort_values(by = 'Revi
```

Out[139…

	From	Review Date
44	London	74
42	LHR	53
46	London Heathrow	31
52	Manchester	30
25	Gatwick	18

We can see that we have a mix of IATA codes and names for airport departure locations. We have the same occurring for the arrival locations also.

In [140…

```
# Groupby the departure location
reviews.groupby('To', as_index = False)['Review Date'].count().sort_values(by = 'Review
```

Out[140…

	To	Review Date
40	London	58
35	LHR	39
42	London Heathrow	25
58	Orlando	21
29	JFK	17

In [141…

```
reviews['From'].unique()
```

Out[141…

```
array(['London', 'Miami', 'Manchester', 'New York JFK', 'London Heathrow',
       'Boston', 'New York', 'Atlanta', 'Antigua', 'Delhi', 'Orlando',
       'Tampa', 'Washington Dulles', 'Los Angeles', 'Heathrow', 'Mumbai',
       'Shanghai', 'San Francisco', 'Austin', 'Detroit, USA', 'Cape Town',
       'Montego Bay', 'Ben Gurion', 'Las Vegas', 'Washington', 'Honolulu',
       'Tel Aviv', 'Bridgetown', 'Johannesburg', 'London Gatwick',
       'San Fransisco', 'Gatwick', 'Glasgow', 'Hong Kong', 'Cork',
       'Dubai', 'Havana', nan, 'LHR', 'Gatwick London', 'MEL', 'Barbados',
       'Perth', 'LGW', 'ATL', 'JFK', 'JNB', 'CUN', 'SFO', 'DEL', 'MCO',
       'KGW', 'MAN', 'HAV', 'ANU', 'PVG', 'BGI', 'LAX', 'LAS', 'EWR',
       'BNE', 'DXB', 'BGO', 'HKG', 'MIA', 'Cancun', 'UVF', 'Glasgow UK',
       'Dulles', 'Aberdeen', 'ABZ'], dtype=object)
```

In [142…

```
# Replace the nan value with 'Not provided'
reviews['From'] = reviews['From'].fillna('Not Provided')
reviews['To'] = reviews['To'].fillna('Not Provided')
```

In [143…

```
# Convert the 'from' locations to a string
names = ' '.join(list(reviews['From'].unique()))

# Convert the 'to' locations to a string
names_to = ' '.join(list(reviews['To'].unique()))

# Create a list of IATA codes in the locations in each column
names = re.findall(r'\b[A-Z]{3}\b',names)
```

```
names_to = re.findall(r'\b[A-Z]{3}\b',names_to)

# Create a set of all the codes combined
names_set = list(set(names + names_to))

# Create a dataframe of all the codes
names_df = pd.DataFrame(names_set)
```

In [144…

```
# Bring in a file of IATA airport codes which we can reference
codes = pd.read_csv(r'C:\Users\imoge\AllMLProjects\Data\airports.csv')
codes_keep = codes[['IATA', 'Airport name']]

# Remove the repeating words 'Airport' and 'International'
codes_keep.loc[:, 'Airport name'] = codes_keep['Airport name'].str.replace('Airport', '
codes_keep.loc[:, 'Airport name'] = codes_keep['Airport name'].str.replace('Internation
```

```
C:\Users\imoge\Anaconda3\envs\MachineLearning\lib\site-packages\pandas\core\indexing.py:
1843: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[item_labels[indexer[info_axis]]] = value
```

In [145…

```
# Merge the two dataframes to get the airport names matched to the IATA codes
merged_names = names_df.merge(codes_keep, how = 'left', left_on = 0, right_on = 'IATA')
merged_names.drop(columns = 0, axis = 1, inplace = True)
merged_names.head(5)
```

Out[145…

	IATA	Airport name
0	BOS	Logan
1	MIA	Miami
2	LHR	London Heathrow
3	CUN	Aeropuerto Internacional de Cancún
4	BNE	Brisbane

In [146…

```
# convert the dataframe to a dictionary and use this to replace the various IATA codes
replacement_map = merged_names.set_index(['IATA'])['Airport name'].to_dict()

# Replace the values in the original dataframe for each of the 'From' and 'To' columns
reviews['From'] = reviews['From'].replace(replacement_map)
reviews['To'] = reviews['To'].replace(replacement_map)
```

In [147…

```
# Strip out whitespace in the columns
reviews['From'] = reviews['From'].str.strip()
reviews['To'] = reviews['To'].str.strip()
```

In [148…

```
# What unique values do we have in the 'From' column?
reviews['From'].unique()
```

Out[148…

```
array(['London', 'Miami', 'Manchester', 'New York JFK', 'London Heathrow',
       'Boston', 'New York', 'Atlanta', 'Antigua', 'Delhi', 'Orlando',
       'Tampa', 'Washington Dulles', 'Los Angeles', 'Heathrow', 'Mumbai',
       'Shanghai', 'San Francisco', 'Austin', 'Detroit, USA', 'Cape Town',
       'Montego Bay', 'Ben Gurion', 'Las Vegas', 'Washington', 'Honolulu',
       'Tel Aviv', 'Bridgetown', 'Johannesburg', 'London Gatwick',
       'San Fransisco', 'Gatwick', 'Glasgow', 'Hong Kong', 'Cork',
       'Dubai', 'Havana', 'Not Provided', 'Gatwick London', 'Melbourne',
       'Barbados', 'Perth', 'Hartsfield Jackson Atlanta',
       'John F Kennedy', 'O.R. Tambo',
       'Aeropuerto Internacional de Cancún', 'Indira Gandhi', 'Kagi',
       'José Martí', 'V C Bird', 'Shanghai Pudong', 'Grantley Adams',
       'Harry Reid', 'Newark Liberty', 'Brisbane', 'Bergen , Flesland',
       'Cancun', 'Hewanorra', 'Glasgow UK', 'Dulles', 'Aberdeen',
       'Aberdeen Dyce'], dtype=object)
```

In [149…

```
# What unique values do we have in the 'To' column?
reviews['To'].unique()
```

Out[149…

```
array(['Mumbai', 'London', 'New York JFK', 'Orlando', 'London Heathrow',
       'Montego Bay', 'New York', 'Los Angeles', 'Male', 'Delhi return',
       'Manchester', 'Heathrow', 'Johannesburg', 'new York', 'to London',
       'Barbados', 'Austin', 'Washington', 'New Orleans', 'Delhi',
       'Hyderabad', 'London, UK', 'Boston', 'Seattle', 'Islamabad',
       'Tel Aviv', 'Las Vegas', 'Dehli', 'Antigua', 'Miami',
       'San Francisco', 'Amsterdam', 'Tobago', 'Havana', 'Atlanta',
       'Durban', 'Newark', 'Florida', 'Hong Kong', 'St Lucia',
       'San Diego', 'Gatwick', 'Cancun', 'London Gatwick', 'Not Provided',
       'John F Kennedy', 'Orlando Florida', 'Lagos', 'Hewanorra',
       'Harry Reid', 'Aeropuerto Internacional de Cancún',
       'Hartsfield Jackson Atlanta', 'Key West', 'O.R. Tambo',
       'Newark Liberty', 'Murtala Muhammed', 'Dubai', 'Indira Gandhi',
       'Logan', 'Shanghai Pudong', 'Grantley Adams', 'A.N.R. Robinson',
       'Las vegas', 'Orlando, Florida', "Chicago O'Hare", 'Bridgetown',
       'Aberdeen', 'Orlando International'], dtype=object)
```

In [127…

```
# Convert to lists
list_a = list(reviews['From'].unique())
list_b = list(reviews['To'].unique())

# Set to lower case (so that we don't have for example 'Las Vegas and 'Las vegas' as tw
list_a = list(map(str.lower, list_a))
list_b = list(map(str.lower, list_b))

# remove the commas and other punctuation, again to avoid multiple entries for the same
list_a = [re.sub(r'[^\w\s]', '', s) for s in list_a]
list_b = [re.sub(r'[^\w\s]', '', s) for s in list_b]

# Combine into a set of unique locations
combined_list = list(set(list_a + list_b))
```

In [207…

```
# Function to clean up the rest of the locations
def run_clean(col):

    # Set to lower case
    reviews[col] = reviews[col].apply(lambda x: x.lower())

    # single word replacement
    reviews[col].replace({'orlando florida': 'orlando', 'glasgow uk': 'glasgow', 'v c b
```

```
                        'san fransisco':'san francisco','detroit, usa': 'detroit', 'jo
                        'ben gurion':'tel aviv', 'murtala muhammed':'lagos','flesland'
                         'bergen , flesland':'bergen','aberdeen dyce':'aberdeen',
                        'washington dulles':'washington','aeropuerto internacional de
                        'harry reid': 'las vegas','anr robinson':'trinidad and tobago'
                        'shanghai pudong':'shanghai','chicago ohare':'chicago','a.n.r.
                        'montego bay':'jamaica'}, inplace=True)


    # multiple word replacements
    barb = ['grantley adams','bridgetown']
    delh = ['indira gandhi','delhi return','dehli']
    heath = ['heathrow','to london','london uk','london','london','london, Uk'] # we wi
    gat = ['london gatwick','gatwick london','gatwick']
    atlant = ['hartsfield jackson atlanta']
    newyork = ['new york jfk','john f kennedy']
    newark = ['newark','newark liberty']
    flor = ['key west','miami']
    joh = ['o.r. tambo','johannesburg']

    reviews.loc[reviews[col].isin(barb), col] = 'barbados'
    reviews.loc[reviews[col].isin(delh), col] = 'delhi'
    reviews.loc[reviews[col].isin(heath), col] = 'london heathrow'
    reviews.loc[reviews[col].isin(gat), col] = 'london gatwick'
    reviews.loc[reviews[col].isin(atlant), col] = 'atlanta'
    reviews.loc[reviews[col].isin(newyork), col] = 'new york'
    reviews.loc[reviews[col].isin(newark), col] = 'newark'
    reviews.loc[reviews[col].isin(flor), col] = 'miami'
    reviews.loc[reviews[col].isin(joh), col] = 'johannesburg'

    reviews[col] = reviews[col].apply(lambda x: x.title())

    return reviews
```

In [208…

```
# Run the function against the 'From' column
reviews = run_clean('From')
```

In [209…

```
# Clean up the textural entries in the 'To' column by running the run clean function
reviews = run_clean('To')
```

In [210…

```
# Check the result
reviews.head()
```

Out[210…

	Review Date	Review Title	Review Text	Flight Date	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food Beverag
0	2025-06-23	Screen monitor is old	Maybe because of the route, food did not ca...	2025-06-01 00:00:00	Economy Class	London Heathrow	Mumbai	3	3	
1	2025-06-23	Seat for economy	Food was decent.	2025-06-01	Economy Class	Miami	London Heathrow	5	5	

	Review Date	Review Title	Review Text	Flight Date	Seat Class	From	To	Seat Comfort	Cabin Staff Service	Food Beverag
		was comfortable	Seat for economy was comfo...	00:00:00						
2	2025-06-11	a great experience	First time with Virgin in over a decade but f...	2025-06-01 00:00:00	Premium Economy	Manchester	New York	4	5	
3	2025-05-23	most people wanted to just sleep	Really good experience in Premium Economy. ...	2025-05-01 00:00:00	Premium Economy	New York	London Heathrow	5	5	
4	2025-05-11	Good selection of movies	Premium economy had its own dedicated check...	2025-05-01 00:00:00	Premium Economy	London Heathrow	Orlando	5	4	

In [211...
```
reviews['From'].unique()
```

Out[211...
```
array(['London Heathrow', 'Miami', 'Manchester', 'New York', 'Boston',
       'Atlanta', 'Antigua', 'Delhi', 'Orlando', 'Tampa', 'Washington',
       'Los Angeles', 'Mumbai', 'Shanghai', 'San Francisco', 'Austin',
       'Detroit', 'Cape Town', 'Jamaica', 'Tel Aviv', 'Las Vegas',
       'Honolulu', 'Barbados', 'Johannesburg', 'London Gatwick',
       'Glasgow', 'Hong Kong', 'Cork', 'Dubai', 'Havana', 'Not Provided',
       'Melbourne', 'Perth', 'Cancun', 'Kagi', 'Anitigua', 'Newark',
       'Brisbane', 'Bergen', 'St Lucia', 'Dulles', 'Aberdeen'],
      dtype=object)
```

In [212...
```
reviews['To'].unique()
```

Out[212...
```
array(['Mumbai', 'London Heathrow', 'New York', 'Orlando', 'Jamaica',
       'Los Angeles', 'Male', 'Delhi', 'Manchester', 'Johannesburg',
       'Barbados', 'Austin', 'Washington', 'New Orleans', 'Hyderabad',
       'London, Uk', 'Boston', 'Seattle', 'Islamabad', 'Tel Aviv',
       'Las Vegas', 'Antigua', 'Miami', 'San Francisco', 'Amsterdam',
       'Tobago', 'Havana', 'Atlanta', 'Durban', 'Newark', 'Florida',
       'Hong Kong', 'St Lucia', 'San Diego', 'London Gatwick', 'Cancun',
       'Not Provided', 'Lagos', 'Dubai', 'Logan', 'Shanghai',
       'Trinidad And Tobago', 'Orlando, Florida', "Chicago O'Hare",
       'Aberdeen', 'Orlando International'], dtype=object)
```

In [213...
```
# Save our final dataframe out to a file
reviews.to_csv(r'C:\Users\imoge\AllMLProjects\Data\AirlineReviewsScraped.csv')
```

We will analyse the data in notebook 2