Abstract

This report presents an analysis of trends and patterns in UK travel data sourced from the Office for National Statistics (ONS) website. This specific data looks at UK travel, focusing on both inbound and outbound travel - which countries visit the UK, their spend, when they arrive, and both seasonal and non-seasonal adjustment data are available. The same is then repeated for UK residents who travel abroad - which countries they visit, why, when they travel, and seasonally adjusted data.

Through the analysis of seasonally adjusted data, insights are drawn into the preferences and behaviours of UK residents when travelling abroad.

The report concludes by discussing how the findings are important for policymakers, businesses, and analysts. By using both seasonally adjusted and raw data, this study provides a complete picture of international travel trends involving the UK. This information helps decision-makers in the travel industry and beyond make better-informed choices.

Introduction and Literature Review

In order to gain a broader understanding of the topic prior to the investigation, there were multiple resources available surrounding travel data relating to the COVID-19 pandemic specifically. "On the economic front, albeit temporary, the pandemic's negative impact on trade was quite significant" (Yepez and Leimgruber, (2024)), which is particularly validated by this research in terms of spending habits both in the UK and globally.

The literature also proved incredibly useful in finding reasons as to why different groups may travel. "... Factors that motivate older people to participate in international retirement migration include: the destination (e.g., climate and amenities), the people (e.g., social networks), the cost (e.g., health insurance and living costs), and the movement (e.g., ease of travel)" (Tate, Snyder and Crooks (2024)). It has allowed for many more factors to be taken into account during the analysis. Despite these articles, there were limitations - such as a lack of literature available relating to time series analysis of COVID-19 data, which highlights the significance of the below work.

```
# Importing relevant packages for analysis.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
import seaborn as sns
!pip install pyspark
sns.set()
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
```

```
from google.colab import drive

# Mounting the Google Drive to allow access to my files.
drive.mount('/content/drive')


# Defining the path to my folder in Google Drive.
excel_file = '/content/drive/MyDrive/TravelDataset.xlsx'

xls = pd.ExcelFile(excel_file)
print(xls.sheet_names)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
['Table1', 'Table2', 'Table3', 'Table4', 'Table5']
```

Data Contents & Pre-Processing

To pre-process the data, I ensured that all of the data was in a format that was readable by Google Colab. To do this, I exported all of the data to a new spreadsheet, removing all of the additional information such as contents and the cover sheet.

Next, I joined these tables together on their common primary key, "Period", using left joins as to avoid any rows being omitted or deleted.

The data contains:

- *Table 1*: Visits to the UK by month from other countries, and country of origin
- *Table 2*: Visits to the UK by month from other countries, and reason for visit
- *Table 3*: Visits from the UK by month to other countries, and country visited
- *Table 4*: Visits from the UK by month to other countries, and reason for visit
- *Table 5*: Spending in the UK, and UK residents spending abroad (in £ millions, by month)

All of this data had to be checked for unavailable or null values. There were multiple nulls discovered in each dataset, which were subsequently replaced by proper NULL values as to make the visualisations easier. If these nulls were not formatted properly, the column would be unable to become an integer value, therefore making visualisation impossible.

```
# Load the sheets into separate DataFrames
df_sheet1 = pd.read_excel(excel_file, sheet_name='Table1')
df_sheet2 = pd.read_excel(excel_file, sheet_name='Table2')
df_sheet3 = pd.read_excel(excel_file, sheet_name='Table3')
df_sheet4 = pd.read_excel(excel_file, sheet_name='Table4')
df_sheet5 = pd.read_excel(excel_file, sheet_name='Table5')


# Creating a pyspark.sql instance to make joining datasets together much easier.
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()


# Examining the ends of the dataset to ensure no extra rows have been added as a result of loading the datasets.
result = df_sheet1.tail()

print(result)
```

```
             Period  North_America   Europe       EU     EU15 Other_EU  \
55      2023 August         659000  2274000  2119000  1773000   345000
56   2023 September         513000  1968000  1799000  1474000   324000
57     2023 October         567000  2453000  2225000  1815000   406000
58    2023 November         372000  1946000  1716000  1360000   347000
59    2023 December         307000  2107000  1924000  1669000   269000

    Other_Countries  World_Total Seasonally_Adjusted_World_Total
55           877000      3810000                   Not available
56           624000      3106000                   Not available
57           695000      3715000                   Not available
58           476000      2795000                   Not available
59           516000      2931000                   Not available
```

```
# Turning the datasets into Pyspark dataframes for the purpose of pre-processing the data.
spark_df_sheet1 = spark.createDataFrame(df_sheet1)
spark_df_sheet2 = spark.createDataFrame(df_sheet2)
spark_df_sheet3 = spark.createDataFrame(df_sheet3)
spark_df_sheet4 = spark.createDataFrame(df_sheet4)
spark_df_sheet5 = spark.createDataFrame(df_sheet5)

FinancialData = spark_df_sheet5


# Imports necessary functions.
from pyspark.sql.functions import col
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Performs a left join on the "Period" column.
VisitsToUK = spark_df_sheet1.join(spark_df_sheet2, on="Period", how="left")

# Shows the result.
VisitsToUK.show()
```

```
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+-----------------------
|        Period|North_America| Europe|           EU|         EU15|     Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_Wor
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+-----------------------
|    2019 April|       319000|2417000|      2192000|      1854000|       345000|         462000|    3199000|
|   2019 August|       593000|2707000|      2499000|      2083000|       418000|        1118000|    4418000|
| 2019 December|       374000|2469000|      2235000|      1814000|       421000|         602000|    3445000|
| 2019 February|       213000|1742000|      1607000|      1272000|       332000|         418000|    2372000|
|  2019 January|       330000|1936000|      1803000|      1483000|       324000|         563000|    2830000|
|     2019 July|       696000|2453000|      2188000|      1802000|       386000|        1006000|    4155000|
|     2019 June|       648000|2243000|      2041000|      1703000|       341000|         836000|    3727000|
|    2019 March|       356000|2365000|      2133000|      1701000|       430000|         407000|    3129000|
|      2019 May|       570000|2219000|      2039000|      1630000|       399000|         650000|    3438000|
| 2019 November|       360000|2209000|      1978000|      1572000|       405000|         552000|    3121000|
|  2019 October|       457000|2537000|      2289000|      1910000|       380000|         737000|    3731000|
|2019 September|       456000|1996000|      1825000|      1505000|       319000|         840000|    3292000|
|    2020 April|         7000|  74000|Not available|Not available|Not available|          13000|      95000|                  Not av
|   2020 August|        77000| 761000|Not available|Not available|Not available|         155000|     993000|                  Not av
| 2020 February|       249000|1869000|      1697000|      1316000|       372000|         394000|    2512000|
|  2020 January|       337000|2032000|      1827000|      1461000|       364000|         667000|    3036000|
|     2020 July|        67000| 463000|Not available|Not available|Not available|         102000|     633000|                  Not av
|     2020 June|        22000| 126000|Not available|Not available|Not available|          29000|     176000|                  Not av
|    2020 March|       208000|1025000|       948000|       826000|       132000|         213000|    1446000|
|      2020 May|        11000|  98000|Not available|Not available|Not available|          17000|     127000|                  Not av
```

```
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+------------------------
only showing top 20 rows
```

```
# Imports necessary functions.
from pyspark.sql.functions import col
import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Performs a left join on the "Period" column.
VisitsFromUK = spark_df_sheet3.join(spark_df_sheet4, on="Period", how="left")

# Shows the result.
VisitsFromUK.show()
```

```
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+------------------------
|        Period|North_America| Europe|           EU|         EU15|     Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_Worl
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+------------------------
|    2019 April|       493000|6252000|      5674000|      4791000|       883000|        1662000|    8406000|
|   2019 August|       685000|9346000|      8597000|      7254000|      1344000|        1597000|   11628000|
| 2019 December|       376000|3666000|      3395000|      2876000|       516000|        1008000|    5050000|
| 2019 February|       231000|4210000|      3816000|      3230000|       585000|        1096000|    5538000|
|  2019 January|       435000|4207000|      3851000|      2999000|       855000|        1507000|    6149000|
|     2019 July|       491000|7039000|      6568000|      5492000|      1075000|        1117000|    8647000|
|     2019 June|       558000|7536000|      7099000|      6015000|      1084000|        1032000|    9125000|
|    2019 March|       204000|5096000|      4699000|      4019000|       678000|        1172000|    6473000|
|      2019 May|       565000|6510000|      6142000|      5192000|       950000|        1153000|    8228000|
| 2019 November|       345000|4216000|      3912000|      3169000|       749000|        1122000|    5683000|
|  2019 October|       532000|6685000|      6056000|      5143000|       909000|        1218000|    8434000|
|2019 September|       649000|7708000|      7147000|      6055000|      1093000|        1368000|    9725000|
|    2020 April|        12000| 145000|Not available|Not available|Not available|          62000|     219000|                  Not av
|   2020 August|        36000|2579000|Not available|Not available|Not available|         144000|    2759000|                  Not av
| 2020 February|       300000|3760000|      3466000|      2884000|       582000|        1172000|    5232000|
|  2020 January|       340000|3993000|      3697000|      2907000|       792000|        1086000|    5419000|
|     2020 July|        17000|1287000|Not available|Not available|Not available|          56000|    1360000|                  Not av
|     2020 June|        18000| 379000|Not available|Not available|Not available|          48000|     445000|                  Not av
|    2020 March|       204000|2212000|      1957000|      1672000|       284000|         824000|    3239000|
|      2020 May|        13000| 220000|Not available|Not available|Not available|          42000|     275000|                  Not av
+--------------+-------------+-------+-------------+-------------+-------------+---------------+-----------+------------------------
only showing top 20 rows
```

Prior to being able to visualise this data, it was important to ensure the 'Period' column was transformed into a datetime type column. This will ensure the ability to perform a time series analysis on the data.

```
from pyspark.sql.functions import to_date
from pyspark.sql.types import DateType

VisitsToUK = VisitsToUK.withColumn("Period", to_date(VisitsToUK["Period"], "yyyy MMMM").cast(DateType()))
VisitsFromUK = VisitsFromUK.withColumn("Period", to_date(VisitsFromUK["Period"], "yyyy MMMM").cast(DateType()))
FinancialData = FinancialData.withColumn("Period", to_date(FinancialData["Period"], "yyyy MMMM").cast(DateType()))
```

```
# Checking and validating that the datatype had been converted correctly to date.
column_types = VisitsToUK.dtypes
for column, data_type in column_types:
    print(f"Column: {column}, Data Type: {data_type}")
```

```
Column: Period, Data Type: date
Column: North_America, Data Type: bigint
Column: Europe, Data Type: bigint
Column: EU, Data Type: string
Column: EU15, Data Type: string
Column: Other_EU, Data Type: string
Column: Other_Countries, Data Type: bigint
Column: World_Total, Data Type: bigint
Column: Seasonally_Adjusted_World_Total, Data Type: string
Column: Holiday, Data Type: bigint
Column: Business, Data Type: bigint
Column: Visiting_friends_or_relatives, Data Type: bigint
Column: Miscellaneous, Data Type: bigint
Column: Total, Data Type: bigint
Column: Seasonally_Adjusted_Total, Data Type: string
```

```
from pyspark.sql.functions import when

# List of columns in the DataFrame.
columns_to_convert = VisitsToUK.columns

# Converting "Not available" values to NA in all columns.
for column in columns_to_convert:
    VisitsToUK = VisitsToUK.withColumn(column,
                                       when(VisitsToUK[column] == "Not available", None).otherwise(VisitsToUK[column]))

# Verifying the changes.
VisitsToUK.show()
```

```
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|    Period|North_America| Europe|     EU|   EU15|Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Busin
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|2019-04-01|       319000|2417000|2192000|1854000|  345000|         462000|    3199000|                        3200000|1403000|  604
|2019-08-01|       593000|2707000|2499000|2083000|  418000|        1118000|    4418000|                        3530000|2217000|  569
|2019-12-01|       374000|2469000|2235000|1814000|  421000|         602000|    3445000|                        3790000|1401000|  549
|2019-02-01|       213000|1742000|1607000|1272000|  332000|         418000|    2372000|                        3150000| 844000|  621
|2019-01-01|       330000|1936000|1803000|1483000|  324000|         563000|    2830000|                        3260000| 973000|  652
|2019-07-01|       696000|2453000|2188000|1802000|  386000|        1006000|    4155000|                        3370000|1866000|  722
|2019-06-01|       648000|2243000|2041000|1703000|  341000|         836000|    3727000|                        3470000|1718000|  832
|2019-03-01|       356000|2365000|2133000|1701000|  430000|         407000|    3129000|                        3570000|1147000|  856
|2019-05-01|       570000|2219000|2039000|1630000|  399000|         650000|    3438000|                        3210000|1388000|  754
|2019-11-01|       360000|2209000|1978000|1572000|  405000|         552000|    3121000|                        3360000|1187000|  801
|2019-10-01|       457000|2537000|2289000|1910000|  380000|         737000|    3731000|                        3530000|1480000|  856
|2019-09-01|       456000|1996000|1825000|1505000|  319000|         840000|    3292000|                        3360000|1282000|  866
|2020-04-01|         7000|  74000|   NULL|   NULL|    NULL|          13000|      95000|                           NULL|  42000|   19
|2020-08-01|        77000| 761000|   NULL|   NULL|    NULL|         155000|     993000|                           NULL| 495000|  127
|2020-02-01|       249000|1869000|1697000|1316000|  372000|         394000|    2512000|                        3310000| 891000|  751
|2020-01-01|       337000|2032000|1827000|1461000|  364000|         667000|    3036000|                        3480000|1125000|  616
|2020-07-01|        67000| 463000|   NULL|   NULL|    NULL|         102000|     633000|                           NULL| 296000|  116
|2020-06-01|        22000| 126000|   NULL|   NULL|    NULL|          29000|     176000|                           NULL|  79000|   38
|2020-03-01|       208000|1025000| 948000| 826000|  132000|         213000|    1446000|                        1720000| 584000|  299
|2020-05-01|        11000|  98000|   NULL|   NULL|    NULL|          17000|     127000|                           NULL|  57000|   27
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
only showing top 20 rows
```

```
from pyspark.sql.functions import when

# List of columns in the DataFrame.
columns_to_convert = VisitsFromUK.columns

# Converting "Not available" values to NA in all columns.
for column in columns_to_convert:
    VisitsFromUK = VisitsFromUK.withColumn(column,
                                       when(VisitsFromUK[column] == "Not available", None).otherwise(VisitsFromUK[column]))

# Verifying the changes.
VisitsFromUK.show()
```

```
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|    Period|North_America| Europe|     EU|   EU15|Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Busin
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|2019-04-01|       493000|6252000|5674000|4791000|  883000|        1662000|    8406000|                        8010000|5041000|  836
|2019-08-01|       685000|9346000|8597000|7254000| 1344000|        1597000|   11628000|                        7780000|8026000|  597
|2019-12-01|       376000|3666000|3395000|2876000|  516000|        1008000|    5050000|                        7370000|2839000|  556
|2019-02-01|       231000|4210000|3816000|3230000|  585000|        1096000|    5538000|                        7680000|3155000|  886
|2019-01-01|       435000|4207000|3851000|2999000|  855000|        1507000|    6149000|                        7830000|2706000|  597
|2019-07-01|       491000|7039000|6568000|5492000| 1075000|        1117000|    8647000|                        7690000|6146000|  605
|2019-06-01|       558000|7536000|7099000|6015000| 1084000|        1032000|    9125000|                        7760000|6219000|  917
|2019-03-01|       204000|5096000|4699000|4019000|  678000|        1172000|    6473000|                        8300000|3879000|  749
|2019-05-01|       565000|6510000|6142000|5192000|  950000|        1153000|    8228000|                        8120000|5483000|  781
|2019-11-01|       345000|4216000|3912000|3169000|  749000|        1122000|    5683000|                        7440000|3211000|  798
|2019-10-01|       532000|6685000|6056000|5143000|  909000|        1218000|    8434000|                        7520000|5516000|  857
|2019-09-01|       649000|7708000|7147000|6055000| 1093000|        1368000|    9725000|                        7620000|6452000|  801
|2020-04-01|        12000| 145000|   NULL|   NULL|    NULL|          62000|     219000|                           NULL| 132000|   21
|2020-08-01|        36000|2579000|   NULL|   NULL|    NULL|         144000|    2759000|                           NULL|1918000|  121
|2020-02-01|       300000|3760000|3466000|2884000|  582000|        1172000|    5232000|                        7090000|3015000|  624
|2020-01-01|       340000|3993000|3697000|2907000|  792000|        1086000|    5419000|                        7050000|2458000|  684
|2020-07-01|        17000|1287000|   NULL|   NULL|    NULL|          56000|    1360000|                           NULL| 951000|   96
|2020-06-01|        18000| 379000|   NULL|   NULL|    NULL|          48000|     445000|                           NULL| 306000|   40
|2020-03-01|       204000|2212000|1957000|1672000|  284000|         824000|    3239000|                        5240000|2187000|  243
|2020-05-01|        13000| 220000|   NULL|   NULL|    NULL|          42000|     275000|                           NULL| 180000|   26
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
only showing top 20 rows
```

```
from pyspark.sql.functions import when

# List of columns in the DataFrame.
columns_to_convert = FinancialData.columns
```

```
# Converting "Not available" values to NA in all columns.
for column in columns_to_convert:
    FinancialData = FinancialData.withColumn(column,
                                   when(FinancialData[column] == "Not Available", None).otherwise(FinancialData[column]))

# Verifying the changes.
FinancialData.show()
```

```
+----------+--------------------------------+-----------------------------------------------------+----------------------------+--
|    Period|Overseas_Residents_Spending_in_UK|Seasonally_Adjusted_Overseas_Residents_Spending_in_UK|UK_Residents_Spending_Abroad|Sea
+----------+--------------------------------+-----------------------------------------------------+----------------------------+--
|2019-01-01|                            1646|                                                 2140|                        3994|
|2019-02-01|                            1318|                                                 2100|                        3356|
|2019-03-01|                            1840|                                                 2350|                        4106|
|2019-04-01|                            1824|                                                 2070|                        5018|
|2019-05-01|                            2347|                                                 2290|                        5011|
|2019-06-01|                            2725|                                                 2430|                        6128|
|2019-07-01|                            3090|                                                 2310|                        5950|
|2019-08-01|                            3510|                                                 2360|                        8687|
|2019-09-01|                            2593|                                                 2320|                        7235|
|2019-10-01|                            2645|                                                 2560|                        5827|
|2019-11-01|                            2201|                                                 2460|                        3906|
|2019-12-01|                            2708|                                                 2980|                        3108|
|2020-01-01|                            2026|                                                 2620|                        3340|
|2020-02-01|                            1538|                                                 2330|                        3454|
|2020-03-01|                             780|                                                 1020|                        2421|
|2020-04-01|                              82|                                                 NULL|                         151|
|2020-05-01|                              75|                                                 NULL|                         139|
|2020-06-01|                              61|                                                 NULL|                         113|
|2020-07-01|                              69|                                                 NULL|                        1361|
|2020-08-01|                             519|                                                 NULL|                         846|
+----------+--------------------------------+-----------------------------------------------------+----------------------------+--
only showing top 20 rows
```

```
# Casting all the columns to integer types for analysis, except 'Period', which follows the datetime format.

VisitsToUK = VisitsToUK.withColumn("North_America", col("North_America").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Europe", col("Europe").cast("int"))
VisitsToUK = VisitsToUK.withColumn("EU", col("EU").cast("int"))
VisitsToUK = VisitsToUK.withColumn("EU15", col("EU15").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Other_EU", col("Other_EU").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Other_Countries", col("Other_Countries").cast("int"))
VisitsToUK = VisitsToUK.withColumn("World_Total", col("World_Total").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Seasonally_Adjusted_World_Total", col("Seasonally_Adjusted_World_Total").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Other_Countries", col("Other_Countries").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Holiday", col("Holiday").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Business", col("Business").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Visiting_friends_or_relatives", col("Visiting_friends_or_relatives").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Miscellaneous", col("Miscellaneous").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Total", col("Total").cast("int"))
VisitsToUK = VisitsToUK.withColumn("Seasonally_Adjusted_Total", col("Seasonally_Adjusted_Total").cast("int"))


# Casting all the columns to integer types for analysis, except 'Period', which follows the datetime format.
VisitsFromUK = VisitsFromUK.withColumn("North_America", col("North_America").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Europe", col("Europe").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("EU", col("EU").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("EU15", col("EU15").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Other_EU", col("Other_EU").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Other_Countries", col("Other_Countries").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("World_Total", col("World_Total").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Seasonally_Adjusted_World_Total", col("Seasonally_Adjusted_World_Total").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Other_Countries", col("Other_Countries").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Holiday", col("Holiday").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Business", col("Business").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Visiting_friends_or_relatives", col("Visiting_friends_or_relatives").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Miscellaneous", col("Miscellaneous").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Total", col("Total").cast("int"))
VisitsFromUK = VisitsFromUK.withColumn("Seasonally_Adjusted_Total", col("Seasonally_Adjusted_Total").cast("int"))


# Casting all the columns to integer types for analysis, except 'Period', which follows the datetime format.
FinancialData = FinancialData.withColumn("Overseas_Residents_Spending_in_UK", col("Overseas_Residents_Spending_in_UK").cast("int"))
FinancialData = FinancialData.withColumn("Seasonally_Adjusted_Overseas_Residents_Spending_in_UK", col("Seasonally_Adjusted_Overseas_Resid
FinancialData = FinancialData.withColumn("UK_Residents_Spending_Abroad", col("UK_Residents_Spending_Abroad").cast("int"))
FinancialData = FinancialData.withColumn("Seasonally_Adjusted_UK_Residents_Spending_Abroad", col("Seasonally_Adjusted_UK_Residents_Spendi
FinancialData = FinancialData.withColumn("Balance", col("Balance").cast("int"))
FinancialData = FinancialData.withColumn("Seasonally_Adjusted_Balance", col("Seasonally_Adjusted_Balance").cast("int"))


# Checking and validating that the datatype had been converted correctly to date.
column_types_touk = VisitsToUK.dtypes
for column, data_type in column_types_touk:
```

```
for column, data_type in column_types_touk:
    print(f"Column: {column}, Data Type: {data_type}")
```

```
Column: Period, Data Type: date
Column: North_America, Data Type: int
Column: Europe, Data Type: int
Column: EU, Data Type: int
Column: EU15, Data Type: int
Column: Other_EU, Data Type: int
Column: Other_Countries, Data Type: int
Column: World_Total, Data Type: bigint
Column: Seasonally_Adjusted_World_Total, Data Type: string
Column: Holiday, Data Type: bigint
Column: Business, Data Type: bigint
Column: Visiting_friends_or_relatives, Data Type: bigint
Column: Miscellaneous, Data Type: bigint
Column: Total, Data Type: bigint
Column: Seasonally_Adjusted_Total, Data Type: string
```

```
# Checking and validating that the datatype had been converted correctly to date.
column_types_fromuk = VisitsFromUK.dtypes
for column, data_type in column_types_fromuk:
    print(f"Column: {column}, Data Type: {data_type}")
```

```
Column: Period, Data Type: date
Column: North_America, Data Type: int
Column: Europe, Data Type: int
Column: EU, Data Type: int
Column: EU15, Data Type: int
Column: Other_EU, Data Type: int
Column: Other_Countries, Data Type: int
Column: World_Total, Data Type: bigint
Column: Seasonally_Adjusted_World_Total, Data Type: string
Column: Holiday, Data Type: bigint
Column: Business, Data Type: bigint
Column: Visiting_friends_or_relatives, Data Type: bigint
Column: Miscellaneous, Data Type: bigint
Column: Total, Data Type: bigint
Column: Seasonally_Adjusted_Total, Data Type: string
```

```
# Checking and validating that the datatype had been converted correctly to date.
column_types_financialdata = FinancialData.dtypes
for column, data_type in column_types_financialdata:
    print(f"Column: {column}, Data Type: {data_type}")
```

```
Column: Period, Data Type: date
Column: Overseas_Residents_Spending_in_UK, Data Type: int
Column: Seasonally_Adjusted_Overseas_Residents_Spending_in_UK, Data Type: int
Column: UK_Residents_Spending_Abroad, Data Type: int
Column: Seasonally_Adjusted_UK_Residents_Spending_Abroad, Data Type: int
Column: Balance, Data Type: int
Column: Seasonally_Adjusted_Balance, Data Type: int
```

```
from pyspark.sql.functions import col

# Orders the DataFrame by the 'Period' column.
ordered_VisitsToUK = VisitsToUK.orderBy(col("Period"))

# Shows the ordered DataFrame.
ordered_VisitsToUK.show()
```

```
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|    Period|North_America| Europe|     EU|   EU15|Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Busin
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|2019-01-01|       330000|1936000|1803000|1483000|  324000|         563000|    2830000|                        3260000| 973000|  652
|2019-02-01|       213000|1742000|1607000|1272000|  332000|         418000|    2372000|                        3150000| 844000|  621
|2019-03-01|       356000|2365000|2133000|1701000|  430000|         407000|    3129000|                        3570000|1147000|  850
|2019-04-01|       319000|2417000|2192000|1854000|  345000|         462000|    3199000|                        3200000|1403000|  604
|2019-05-01|       570000|2219000|2039000|1630000|  399000|         650000|    3438000|                        3210000|1388000|  754
|2019-06-01|       648000|2243000|2041000|1703000|  341000|         836000|    3727000|                        3470000|1718000|  832
|2019-07-01|       696000|2453000|2188000|1802000|  386000|        1006000|    4155000|                        3370000|1866000|  722
|2019-08-01|       593000|2707000|2499000|2083000|  418000|        1118000|    4418000|                        3530000|2217000|  569
|2019-09-01|       456000|1996000|1825000|1505000|  319000|         840000|    3292000|                        3360000|1282000|  866
|2019-10-01|       457000|2537000|2289000|1910000|  380000|         737000|    3731000|                        3530000|1480000|  850
|2019-11-01|       360000|2209000|1978000|1572000|  405000|         552000|    3121000|                        3360000|1187000|  801
|2019-12-01|       374000|2469000|2235000|1814000|  421000|         602000|    3445000|                        3790000|1401000|  549
|2020-01-01|       337000|2032000|1827000|1461000|  364000|         667000|    3036000|                        3480000|1125000|  616
|2020-02-01|       249000|1869000|1697000|1316000|  372000|         394000|    2512000|                        3310000| 891000|  751
|2020-03-01|       208000|1025000| 948000| 826000|  132000|         213000|    1446000|                        1720000| 584000|  299
|2020-04-01|         7000|  74000|   NULL|   NULL|    NULL|          13000|      95000|                           NULL|  42000|   19
|2020-05-01|        11000|  98000|   NULL|   NULL|    NULL|          17000|     127000|                           NULL|  57000|   27
|2020-06-01|        22000| 126000|   NULL|   NULL|    NULL|          29000|     176000|                           NULL|  79000|   38
|2020-07-01|        67000| 463000|   NULL|   NULL|    NULL|         102000|     633000|                           NULL| 296000|  116
|2020-08-01|        77000| 761000|   NULL|   NULL|    NULL|         155000|     993000|                           NULL| 495000|  127
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
only showing top 20 rows
```

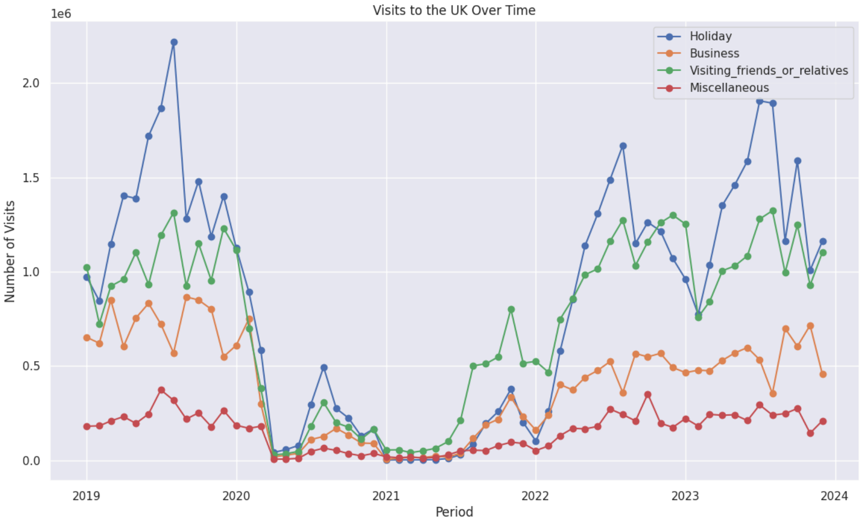
```
# Orders the DataFrame by the 'Period' column.
ordered_VisitsFromUK = VisitsFromUK.orderBy(col("Period"))


# Shows the ordered DataFrame.
ordered_VisitsFromUK.show()
```

```
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|    Period|North_America| Europe|     EU|   EU15|Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Busin
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|2019-01-01|       435000|4207000|3851000|2999000|  855000|        1507000|    6149000|                        7830000|2706000|   597
|2019-02-01|       231000|4210000|3816000|3230000|  585000|        1096000|    5538000|                        7680000|3155000|   886
|2019-03-01|       204000|5096000|4699000|4019000|  678000|        1172000|    6473000|                        8300000|3879000|   749
|2019-04-01|       493000|6252000|5674000|4791000|  883000|        1662000|    8406000|                        8010000|5041000|   836
|2019-05-01|       565000|6510000|6142000|5192000|  950000|        1153000|    8228000|                        8120000|5483000|   781
|2019-06-01|       558000|7536000|7099000|6015000| 1084000|        1032000|    9125000|                        7760000|6219000|   917
|2019-07-01|       491000|7039000|6568000|5492000| 1075000|        1117000|    8647000|                        7690000|6146000|   605
|2019-08-01|       685000|9346000|8597000|7254000| 1344000|        1597000|   11628000|                        7780000|8026000|   597
|2019-09-01|       649000|7708000|7147000|6055000| 1093000|        1368000|    9725000|                        7620000|6452000|   801
|2019-10-01|       532000|6685000|6056000|5143000|  909000|        1218000|    8434000|                        7520000|5516000|   857
|2019-11-01|       345000|4216000|3912000|3169000|  749000|        1122000|    5683000|                        7440000|3211000|   798
|2019-12-01|       376000|3666000|3395000|2876000|  516000|        1008000|    5050000|                        7370000|2839000|   556
|2020-01-01|       340000|3993000|3697000|2907000|  792000|        1086000|    5419000|                        7050000|2458000|   684
|2020-02-01|       300000|3760000|3466000|2884000|  582000|        1172000|    5232000|                        7090000|3015000|   624
|2020-03-01|       204000|2212000|1957000|1672000|  284000|         824000|    3239000|                        5240000|2187000|   243
|2020-04-01|        12000| 145000|   NULL|   NULL|    NULL|          62000|     219000|                           NULL| 132000|    21
|2020-05-01|        13000| 220000|   NULL|   NULL|    NULL|          42000|     275000|                           NULL| 180000|    26
|2020-06-01|        18000| 379000|   NULL|   NULL|    NULL|          48000|     445000|                           NULL| 306000|    46
|2020-07-01|        17000|1287000|   NULL|   NULL|    NULL|          56000|    1360000|                           NULL| 951000|    96
|2020-08-01|        36000|2579000|   NULL|   NULL|    NULL|         144000|    2759000|                           NULL|1918000|   121
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
only showing top 20 rows
```

```
from pyspark.sql.functions import col

# Orders the DataFrame by the 'Period' column.
ordered_FinancialData = FinancialData.orderBy(col("Period"))

# Shows the ordered DataFrame.
ordered_FinancialData.show()
```

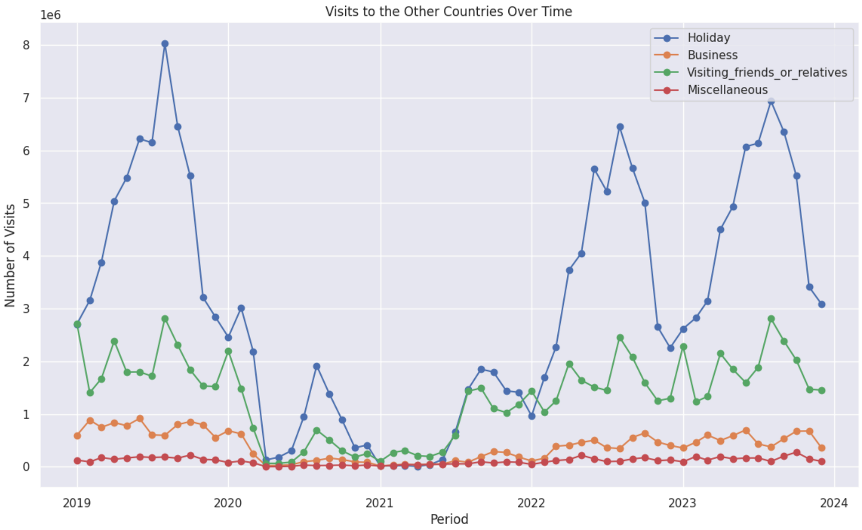
```
+----------+---------------------------------+-----------------------------------------------------+---------------------------+--
|    Period|Overseas_Residents_Spending_in_UK|Seasonally_Adjusted_Overseas_Residents_Spending_in_UK|UK_Residents_Spending_Abroad|Sea
+----------+---------------------------------+-----------------------------------------------------+---------------------------+--
|2019-01-01|                             1646|                                                 2140|                       3994|
|2019-02-01|                             1318|                                                 2100|                       3356|
|2019-03-01|                             1840|                                                 2350|                       4106|
|2019-04-01|                             1824|                                                 2070|                       5018|
|2019-05-01|                             2347|                                                 2290|                       5011|
|2019-06-01|                             2725|                                                 2430|                       6128|
|2019-07-01|                             3090|                                                 2310|                       5950|
|2019-08-01|                             3510|                                                 2360|                       8687|
|2019-09-01|                             2593|                                                 2320|                       7235|
|2019-10-01|                             2645|                                                 2560|                       5827|
|2019-11-01|                             2201|                                                 2460|                       3906|
|2019-12-01|                             2708|                                                 2980|                       3108|
|2020-01-01|                             2026|                                                 2620|                       3340|
|2020-02-01|                             1538|                                                 2330|                       3454|
|2020-03-01|                              780|                                                 1020|                       2421|
|2020-04-01|                               82|                                                 NULL|                        151|
|2020-05-01|                               75|                                                 NULL|                        139|
|2020-06-01|                               61|                                                 NULL|                        113|
|2020-07-01|                               69|                                                 NULL|                       1361|
|2020-08-01|                              519|                                                 NULL|                        846|
+----------+---------------------------------+-----------------------------------------------------+---------------------------+--
only showing top 20 rows
```

```
# What are most popular reasons for countries entering UK?
import matplotlib.pyplot as plt

# Plotting multiple columns.
plt.figure(figsize=(14, 8))

plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Holiday'], label='Holiday', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Business'], label='Business', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Visiting_friends_or_relatives'], label='Visiting_friends_or_relatives', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Miscellaneous'], label='Miscellaneous', marker='o')

# Customising the plot.
# Customising the plot.
plt.xlabel('Period')
plt.ylabel('Number of Visits')
plt.title('Visits to the UK Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



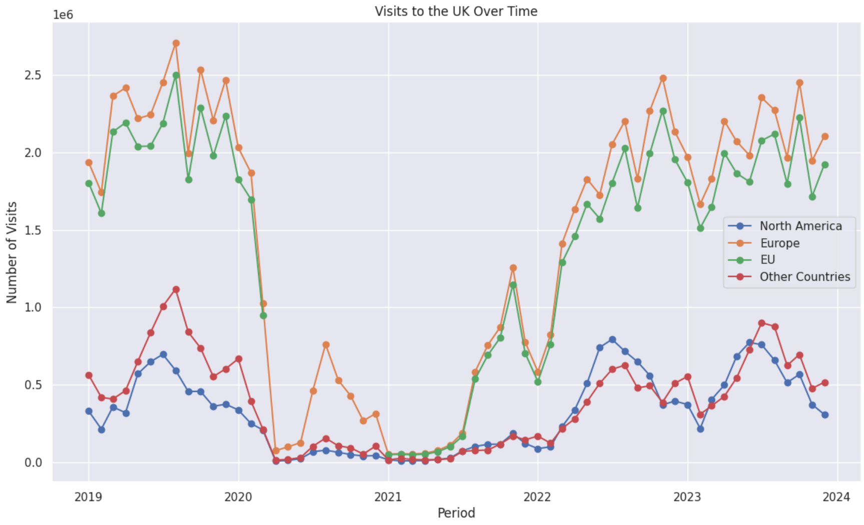
Typically, most people will visit the UK for holiday reasons, however there was a noticeable uptick in people visiting friends or relatives near the end of 2022, perhaps as a result of the COVID-19 pandemic meaning people were much more keen to see their distant family members. The amount of people visiting for business related reasons is relevantly consistent, but yet again dips and almost reaches a level of 0 during the pandemic due to flight restrictions or outright bans.

```
# What are most popular reasons for countries leaving UK?
import matplotlib.pyplot as plt

# Plotting multiple columns.
plt.figure(figsize=(14, 8))

plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Holiday'], label='Holiday', marker='o')
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Business'], label='Business', marker='o')
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Visiting_friends_or_relatives'], label='Visiting_friends_or_relatives', marker=
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Miscellaneous'], label='Miscellaneous', marker='o')

# Customising the plot.
plt.xlabel('Period')
plt.ylabel('Number of Visits')
plt.title('Visits to Other Countries Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



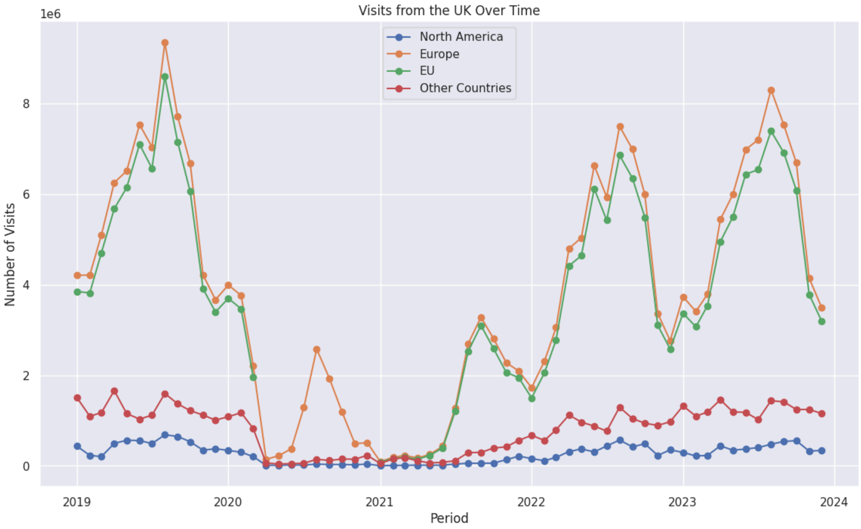
Most commonly, UK residents will visit other countries for purposes of holiday - but during the pandemic, due to non-essential travel bans, the rate of holiday travel almost reached a full standstill.

```
# Which countries do entrants come from?

import matplotlib.pyplot as plt

# Plotting multiple columns.
plt.figure(figsize=(14, 8))

plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['North_America'], label='North America', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Europe'], label='Europe', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['EU'], label='EU', marker='o')
plt.plot(pandas_VisitsToUK.index, pandas_VisitsToUK['Other_Countries'], label='Other Countries', marker='o')

# Customising the plot.
plt.xlabel('Period')
plt.ylabel('Number of Visits')
plt.title('Visits to the UK Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



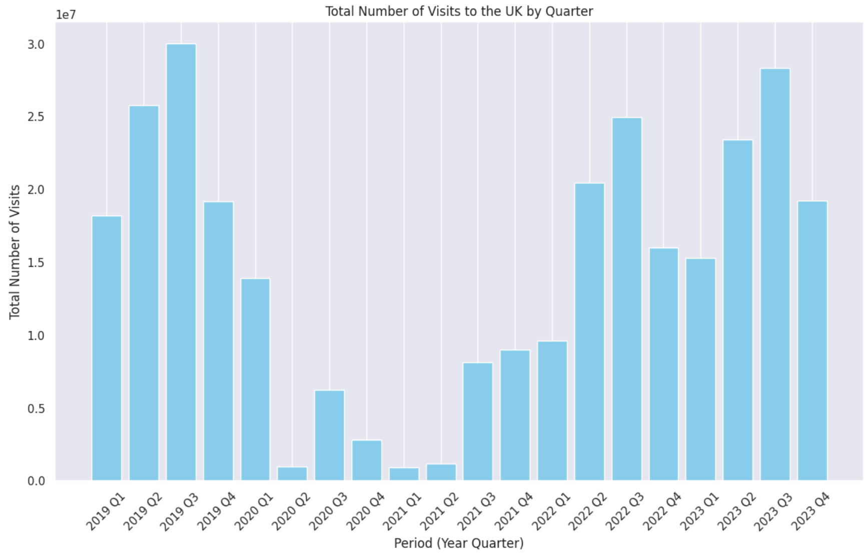
When reviewing the data, the most common area for visitors to the UK is Europe and the countries that make up the European Union. This is likely due to its closer proximity and therefore lower travel costs, with accessibility being a huge feature. Other countries are far less likely, including North America, which is likely due to the high travel times and the greatly increased costs. The time period values are also interesting, as there appears to be less travel restrictions from the rest of Europe compared to any other category, which could be due to policy.

```
# Which countries do UK residents go to?
import matplotlib.pyplot as plt

# Plotting multiple columns.
plt.figure(figsize=(14, 8))

plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['North_America'], label='North America', marker='o')
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Europe'], label='Europe', marker='o')
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['EU'], label='EU', marker='o')
plt.plot(pandas_VisitsFromUK.index, pandas_VisitsFromUK['Other_Countries'], label='Other Countries', marker='o')

# Customising the plot.
plt.xlabel('Period')
plt.ylabel('Number of Visits')
plt.title('Visits from the UK Over Time')
plt.legend()
plt.grid(True)
plt.show()
```

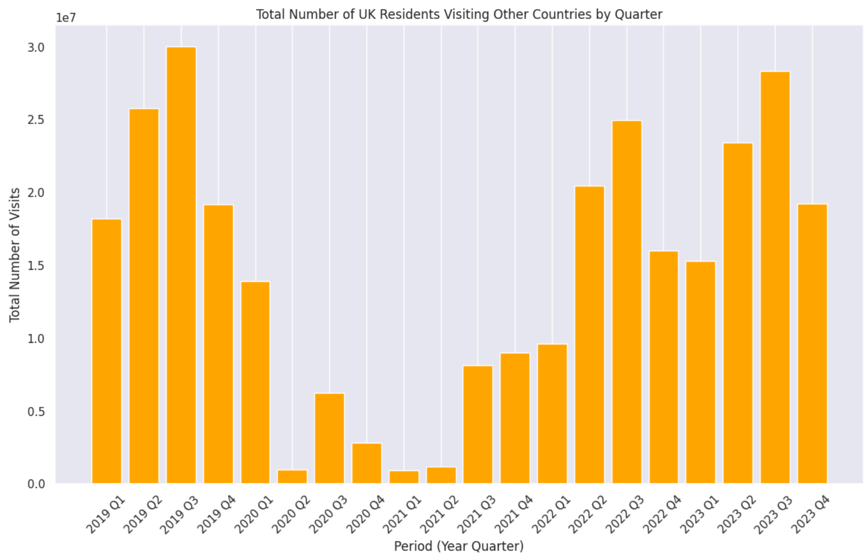


UK residents seem to visit Europe the most out of any group, even during the pandemic period - which is again likely due to its much closer proximity. The figures between Europe and the European Union countries are similar due to the huge overlap in countries which are included in both - however the European Union contains some further away countries which may have impacted its result (further distance likely indicates higher prices, and the higher travel time is likely to disinterest travellers). The results may also be impacted due to some countries in Europe not being themselves part of the European Union, which decreases its numbers.

```
# What peak months do people enter?
import matplotlib.pyplot as plt
from pyspark.sql.functions import col, quarter, year, sum as spark_sum

VisitsToUK = VisitsToUK.withColumn("Year", year(col("Period")))
VisitsToUK = VisitsToUK.withColumn("Quarter", quarter(col("Period")))


# Groups by Year and Quarter, and sums the totals.
quarterly_visits = VisitsToUK.groupBy("Year", "Quarter").agg(spark_sum("Total").alias("Total_Visits"))

# Converting to Pandas DataFrame.
pandas_df = quarterly_visits.toPandas()

# Combines Year and Quarter into a single column.
pandas_df['Year_Quarter'] = pandas_df['Year'].astype(str) + ' Q' + pandas_df['Quarter'].astype(str)


pandas_df.sort_values(by='Year_Quarter', inplace=True)

# Plotting data.
plt.figure(figsize=(14, 8))

# Creates bar chart.
plt.bar(pandas_df['Year_Quarter'], pandas_df['Total_Visits'], color='skyblue')

# Customising the plot.
plt.xlabel('Period (Year Quarter)')
plt.ylabel('Total Number of Visits')
plt.title('Total Number of Visits to the UK by Quarter')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Shows plot.
plt.show()
```



```
# What peak months do people leave?
import matplotlib.pyplot as plt
from pyspark.sql.functions import col, quarter, year, sum as spark_sum

VisitsFromUK = VisitsFromUK.withColumn("Year", year(col("Period")))
VisitsFromUK = VisitsFromUK.withColumn("Quarter", quarter(col("Period")))


# Groups by Year and Quarter, and sums the totals.
quarterly_visits = VisitsFromUK.groupBy("Year", "Quarter").agg(spark_sum("Total").alias("Total_Visits"))

# Converts to Pandas DataFrame.
pandas_df = quarterly_visits.toPandas()

# Combines Year and Quarter into a single column.
pandas_df['Year_Quarter'] = pandas_df['Year'].astype(str) + ' Q' + pandas_df['Quarter'].astype(str)


pandas_df.sort_values(by='Year_Quarter', inplace=True)

# Plotting data.
plt.figure(figsize=(14, 8))

# Creates bar chart.
plt.bar(pandas_df['Year_Quarter'], pandas_df['Total_Visits'], color='orange')

# Customising the plot.
plt.xlabel('Period (Year Quarter)')
plt.ylabel('Total Number of Visits')
plt.title('Total Number of UK Residents Visiting Other Countries by Quarter')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Shows the plot.
plt.show()
```

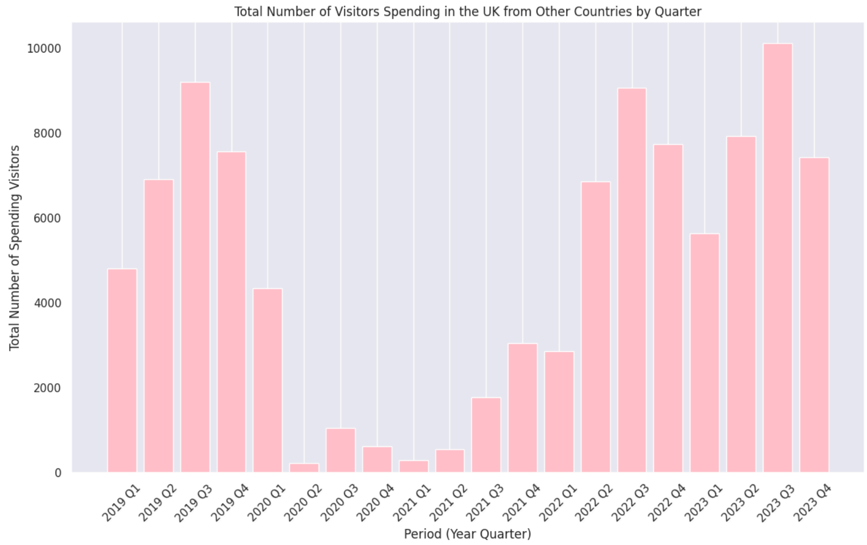


These graphs clearly indicate an overall trend which is highly indicative of COVID-19 being not only a UK resident impacting issue, but a global travel issue due to restrictions. This highlights their effectiveness, of which it is easy to hypothesise that the travel spend will be impacted also as a result.

```
# Performs a left join on the "Period" column.
VisitsToUKFinancial = VisitsToUK.join(FinancialData, on="Period", how="left")

# Shows the result.
VisitsToUKFinancial.show()
```

```
------+---------------+-----------+-------------------------------+-------+--------+-----------------------------+-------------+----
her_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Business|Visiting_friends_or_relatives|Miscellaneous|  To
------+---------------+-----------+-------------------------------+-------+--------+-----------------------------+-------------+----
345000|         462000|    3199000|                        3200000|1403000|  604000|                       960000|       232000|3199
380000|         737000|    3731000|                        3530000|1480000|  850000|                      1149000|       252000|3731
364000|         667000|    3036000|                        3480000|1125000|  610000|                      1115000|       186000|3036
132000|         213000|    1446000|                        1720000| 584000|  299000|                       382000|       181000|1446
  NULL|         103000|     458000|                           NULL| 167000|   90000|                       165000|        37000| 458
399000|         650000|    3438000|                        3210000|1388000|  754000|                      1101000|       196000|3438
  NULL|          13000|      95000|                           NULL|  42000|   19000|                        29000|         6000|  95
341000|         836000|    3727000|                        3470000|1718000|  832000|                       932000|       246000|3727
 15000|          24000|      86000|                           NULL|   3000|   12000|                        56000|        14000|  86
332000|         418000|    2372000|                        3150000| 844000|  621000|                       724000|       183000|2372
 13000|          17000|      78000|                           NULL|   3000|   16000|                        41000|        18000|  78
 16000|          16000|     107000|                           NULL|   4000|   23000|                        64000|        16000| 107
  NULL|         102000|     633000|                           NULL| 296000|  110000|                       180000|        47000| 633
386000|        1006000|    4155000|                        3370000|1866000|  722000|                      1192000|       374000|4155
  NULL|         155000|     993000|                           NULL| 495000|  127000|                       307000|        65000| 993
430000|         407000|    3129000|                        3570000|1147000|  850000|                       923000|       209000|3129
  NULL|          17000|     127000|                           NULL|  57000|   27000|                        35000|         7000| 127
324000|         563000|    2830000|                        3260000| 973000|  652000|                      1025000|       181000|2830
319000|         840000|    3292000|                        3360000|1282000|  866000|                       925000|       219000|3292
421000|         602000|    3445000|                        3790000|1401000|  549000|                      1230000|       265000|3445
------+---------------+-----------+-------------------------------+-------+--------+-----------------------------+-------------+----
```

```
# Groups by Year and Quarter, and sums the totals.
quarterly_finance_in_uk = VisitsToUKFinancial.groupBy("Year", "Quarter").agg(spark_sum("Overseas_Residents_Spending_in_UK").alias("Tota

# Converts to Pandas DataFrame.
pandas_VisitsToUKFinancial = quarterly_finance_in_uk.toPandas()

# Combines Year and Quarter into a single column.
pandas_VisitsToUKFinancial['Year_Quarter'] = pandas_VisitsToUKFinancial['Year'].astype(str) + ' Q' + pandas_VisitsToUKFinancial['Quarte


# Converts PySpark DataFrame to Pandas DataFrame.

pandas_VisitsToUKFinancial.sort_values(by='Year_Quarter', inplace=True)

# Plotting data.
plt.figure(figsize=(14, 8))

# Creates bar chart.
plt.bar(pandas_VisitsToUKFinancial['Year_Quarter'], pandas_VisitsToUKFinancial['Total_Spending_in_UK'], color='pink')

# Customising the plot.
plt.xlabel('Period (Year Quarter)')
plt.ylabel('Total Number of Spending Visitors')
plt.title('Total Number of Visitors Spending in the UK from Other Countries by Quarter')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Shows plot.
plt.show()
```

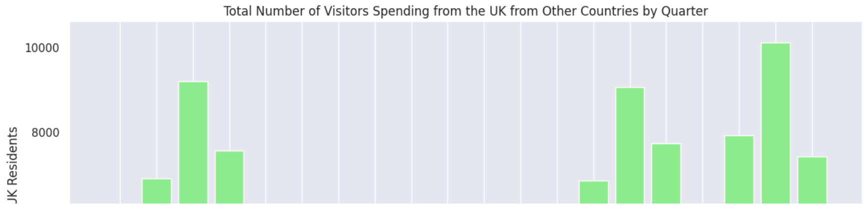


```
# Performs a left join on the "Period" column.
VisitsFromUKFinancial = VisitsFromUK.join(FinancialData, on="Period", how="left")

# Shows the result.
VisitsFromUKFinancial.show()
```

```
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|    Period|North_America| Europe|     EU|   EU15|Other_EU|Other_Countries|World_Total|Seasonally_Adjusted_World_Total|Holiday|Busi
+----------+-------------+-------+-------+-------+--------+---------------+-----------+-------------------------------+-------+----
|2019-02-01|       231000|4210000|3816000|3230000|  585000|        1096000|    5538000|                        7680000|3155000|   88
|2019-10-01|       532000|6685000|6056000|5143000|  909000|        1218000|    8434000|                        7520000|5516000|   85
|2019-09-01|       649000|7708000|7147000|6055000| 1093000|        1368000|    9725000|                        7620000|6452000|   80
```

```
     |2021-02-01|         9000| 186000| 161000|  88000|   68000|         151000|     346000|                          NULL|  17000|   32
     |2019-07-01|       491000|7039000|6568000|5492000| 1075000|        1117000|    8647000|                       7690000|6146000|   605
     |2020-08-01|        36000|2579000|   NULL|   NULL|    NULL|         144000|    2759000|                          NULL|1918000|   121
     |2020-05-01|        13000| 220000|   NULL|   NULL|    NULL|          42000|     275000|                          NULL| 180000|   26
     |2020-12-01|        38000| 510000|   NULL|   NULL|    NULL|         231000|     779000|                          NULL| 411000|   85
     |2020-03-01|       204000|2212000|1957000|1672000|  284000|         824000|    3239000|                       5240000|2187000|   243
     |2019-04-01|       493000|6252000|5674000|4791000|  883000|        1662000|    8406000|                       8010000|5041000|   836
     |2020-04-01|        12000| 145000|   NULL|   NULL|    NULL|          62000|     219000|                          NULL| 132000|   21
     |2021-03-01|        11000| 232000| 193000| 131000|   65000|         181000|     423000|                          NULL|  23000|   56
     |2020-07-01|        17000|1287000|   NULL|   NULL|    NULL|          56000|    1360000|                          NULL| 951000|   96
     |2019-12-01|       376000|3666000|3395000|2876000|  516000|        1008000|    5050000|                       7370000|2839000|   556
     |2019-01-01|       435000|4207000|3851000|2999000|  855000|        1507000|    6149000|                       7830000|2706000|   597
     |2019-05-01|       565000|6510000|6142000|5192000|  950000|        1153000|    8228000|                       8120000|5483000|   781
     |2020-01-01|       340000|3993000|3697000|2907000|  792000|        1086000|    5419000|                       7050000|2458000|   684
     |2019-03-01|       204000|5096000|4699000|4019000|  678000|        1172000|    6473000|                       8300000|3879000|   749
     |2021-04-01|        15000| 172000| 144000|  82000|   59000|         109000|     296000|                          NULL|   8000|   36
     |2021-05-01|        13000| 254000| 231000| 152000|   78000|          64000|     331000|                          NULL|  39000|   54
     +----------+-------------+-------+-------+-------+--------+---------------+-----------+------------------------------+-------+----
     only showing top 20 rows
```

```
# Groups by Year and Quarter, and sums the totals.
quarterly_finance_from_uk = VisitsFromUKFinancial.groupBy("Year", "Quarter").agg(spark_sum("Overseas_Residents_Spending_in_UK").alias("

# Converts to Pandas DataFrame.
pandas_VisitsFromUKFinancial = quarterly_finance_from_uk.toPandas()

# Combines Year and Quarter into a single column.
pandas_VisitsFromUKFinancial['Year_Quarter'] = pandas_VisitsFromUKFinancial['Year'].astype(str) + ' Q' + pandas_VisitsFromUKFinancial['


# Converts PySpark DataFrame to Pandas DataFrame.

pandas_VisitsFromUKFinancial.sort_values(by='Year_Quarter', inplace=True)

# Plotting data.
plt.figure(figsize=(14, 8))

# Creates bar chart.
plt.bar(pandas_VisitsFromUKFinancial['Year_Quarter'], pandas_VisitsFromUKFinancial['Total_Spending_from_UK'], color='lightgreen')

# Customising the plot.
plt.xlabel('Period (Year Quarter)')
plt.ylabel('Total Number of Spending UK Residents')
plt.title('Total Number of Visitors Spending from the UK from Other Countries by Quarter')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Shows plot.
plt.show()
```



Both graphs clearly indicate the impact of the COVID-19 pandemic not just for residents of the UK, but worldwide. There is a clear decrease in 2020 Q2 due to travel restrictions - however, interestingly, it is not a complete flat amount for either graph. This could be due to people staying in the UK for extended periods prior to the pandemic - for reasons of work, or visiting friends and family for long term, and thus being either stuck in the UK or stuck abroad, meaning they are still spending money there.

Discussion

The data shows how drastically the COVID-19 pandemic affected travel. There was a significant drop in travel during the lockdowns in 2020, highlighting the industry's vulnerability to such global events. During the pandemic, there was an increase in people visiting friends and family, especially towards the end of 2022. This shift suggests that people prioritized personal connections over holidays or business trips during these times.

Despite the decline in travel, spending did not completely stop. This indicates some resilience in the travel industry, likely due to factors like flexible bookings and continued travel for essential reasons. Travel to and from Europe remained high, probably due to its proximity and lower travel costs. This shows how economic and political factors influence travel choices.

These findings are important for policymakers who need to understand travel trends to make informed decisions. The data can help shape policies to boost travel demand and support the industry’s recovery. Businesses in the travel sector can use these insights to adjust their strategies. Understanding travel trends and spending behaviors helps businesses meet new demands and mitigate risks.

The analysis has limitations, such as potential biases in the data. Future research should look at long-term trends, compare different regions, and include qualitative data to get a fuller picture of travel behaviors.

In summary, this critical discussion highlights the complexity of travel trends and their implications for the industry. By understanding these results, stakeholders can better navigate the challenges and opportunities in the travel sector.

Conclusion

In conclusion, the analysis of UK travel data has provided valuable insights into travel trends, especially during the COVID-19 pandemic. There are visible significant changes in travel behaviour, with fewer people going on holidays or traveling for business.

European destinations remain popular among travelers, likely due to their proximity and ease of access. However, travel patterns vary based on factors like regulations, economic conditions, and geopolitical factors.

The graphs depicting travel trends serve as visual representations of the severe impact of COVID-19 on global travel, highlighting periods of significant disruption. Despite these challenges, the resilience of the travel industry and the adaptability of travelers are clear, suggesting potential opportunities for recovery and growth in the post-pandemic world.

Overall, this analysis helps us understand how travel is changing and informs decisions on policies and industry recovery efforts. By using this