CS5052

Practical 1: Apache Spark

REPORT

220026989

Word Count: 2979 words

1. Introduction

This report explains the interpretation and analysis of a large dataset done with the help of Apache Spark using python. The data deals with the various details of absentees over various schools that are categories locally, regionally and nationally over different time periods. The given data set deals with the time period 2006 – 2018. A console based application and a web based application is finally built for the users to interact and understand various features of the data.

1.i. Implementation checklist

- ✓ Part A – Completed
- ✓ Part B – Completed
- ✓ Part C - Web application using streamlit completed

1.ii. Compilation and running instructions

- The program is created with python v3.11.2 and pip v22.3.1
  S
- Dependencies
  Before starting the program it is advised to install all the dependencies required:

  pip install requirements.txt

- Running console-application
  python console.py

- Running web-based application
  streamlit run webapp.py

2. Structure and Implementation

2.i. Structure

The files for the program are split into:

- console.py
- create.py

Part A:

- authorised_absence.py
- enrolls_by_la.py
- top_three_auth_absence.py
- unauth_absence.py

Part B:

- compare_two_la.py
- region_performance_2006_18.py

Part C:

- webapp.py

2.ii. Implementation

2.ii.i. Initial exploration

Initial exploration of data is important when working with huge analysis of data. The basics of data exploration is to check for null values and irrelevant values. Basic exploration of data shows the existence of null values in certain columns. This was due to the way the data is provided: to be more specific – the geographic level which is divided into National, Regional and Local authority are all stacked into a single column, that causes some null values in the corresponding columns that are meant for region name, local authority name. This arrangement clubbed with the arrangement of school-type is what describes the rest of the data. It is crucial to understand that the school type is also divided into similar manner where various school types including the total schools according to the geographic location are stacked into a single column.

Further exploration showed the requirement of data cleaning where one of the columns of the data is partially filled with strings whereas the arrangement of data implies that they are meant to be integers. This column specifically - *sess_auth_ext_holiday*, is filled with irregular values and integers which is required to be cleaned. Simple call on pyspark's distinct() shows that the column is filled with ":" values and integer values. The ":" values are then converted into zero as to avoid future errors while working on the data. Finally the hypothesis of whether the various school types in local add up to regional and regional add up to national is verified. Now as we understand the basic outline of the data, we move on to perform the tasks required.

2.ii.ii. create.py

This python script creates a PySpark session and reads a CSV file into a PySpark DataFrame. The script defines two functions: **create()** and **create_for_streamlit(path)**.

The **create()** function prompts the user to enter the path to the CSV file, reads the CSV file into a PySpark DataFrame using **spark.read.csv()** function, and then cleans the **sess_auth_ext_holiday** column by replacing invalid values (":") with 0 and casting the column to an integer data type. Note that if the file exists in the default location, that is in the location of console.py the path should be kept as null.

The **create_for_streamlit(path)** function takes a path argument as input, reads the CSV file into a PySpark DataFrame using **spark.read.csv()** function, and then cleans the **sess_auth_ext_holiday** column by replacing invalid values (":") with 0 and casting the column to an integer data type.

The create() and create_for_streamlit() does the same function although the first function is for the console app and the latter is specifically for the webapp.

2.ii.iii. console.py

The application initially creates a PySpark dataframe by calling the create() function from create.py. This function reads in a CSV file and returns a PySpark dataframe. The CSV file contains data on pupil enrolments and absences in schools in England from 2006 to 2018.

The application then enters a while loop that prompts the user to input a choice from a menu of options. The options include searching the dataset by local authority or school type, displaying unauthorised absences in a certain year, comparing two local authorities for any session in a given year, and charting the performance of regions in England from 2006 to 2018.

The user's input is matched with a case statement, which calls the appropriate function depending on the user's choice.

For example, if the user chooses option 1 (search the dataset by local authority), the application prompts the user to enter local authorities. It then splits the input into a list of local authorities and passes this list and the PySpark dataframe to the display_enrol_by_la() function from enrolls_by_la.py. This function returns a PySpark dataframe that contains the number of pupil enrolments for the specified local authorities. The resulting dataframe is then displayed to the user.

Similarly, if the user chooses option 2 (search the dataset by school type), the application prompts the user to enter the school type. It then passes this school type and the PySpark dataframe to the auth_leave_schooltype() function from authorised_absence.py. This function returns a PySpark dataframe that contains the total number of pupils who were given authorised absences because of medical appointments or illness in the time period 2017-2018. The resulting dataframe is then displayed to the user.

If the user chooses option 5 (compare two local authorities for any session in a given year), the application enters a nested while loop that allows the user to choose from a menu of options to compare. The user can choose to compare authorized reasons sessions, overall absence rate, authorized absence rate, or unauthorized absence rate. The application then prompts the user to enter local authorities (comma-separated) and the year to compare. It then passes this information, the chosen option, and the PySpark dataframe to the compare_local_auth() function from compare_two_la.py. This function returns a figure object that displays a comparison between the two local authorities for the chosen option in the given year. The resulting figure object is then displayed to the user.

Finally, if the user chooses option 6 (chart/explore the performance of regions in England from 2006-2018), the application passes the PySpark dataframe to the regional_performance_2006_2018() function from region_performance_2006_18.py. This function returns two figure objects that display the performance of regions in England from 2006 to 2018. The resulting figure objects are then displayed to the user.

2.ii.iv. PART A – 1:

Given a list of local authorities, display in a well-formatted fashion the number of pupil enrolments in each local authority by time period (year). – enrol_by_la.py

The **display_enrol_by_la** function takes in two parameters: **la_list** which is a list of local authorities and **df** which is a PySpark dataframe. The function filters the PySpark dataframe to select only the rows where the **geographic_level** is "Local authority" and **school_type** is "Total". This is done because among each geographic level as explained earlier there are different school types. The “Total” school type takes into consideration the overall data of all the different type of schools in that given local authority. Then, it filters the resulting dataframe again to select only the rows where the **la_name** column matches any of the local authorities in the input **la_list**. Finally, it selects the **time_period**, **la_name**, and **enrolments** columns from the filtered dataframe and returns the resulting PySpark dataframe.

```
        4. List the top 3 reasons for authorised absences in each year
        5. Compare two local authorities for any session in a given year
        6. Chart/explore the performance of regions in England from 2006-2018
        7. Quit

Input :  1
Enter local authorities: City of London
+-----------+--------------+----------+
|time_period|       la_name|enrolments|
+-----------+--------------+----------+
|     201819|City of London|       207|
|     201718|City of London|       211|
|     201617|City of London|       179|
|     201516|City of London|       181|
|     201415|City of London|       181|
|     201314|City of London|       179|
|     201213|City of London|       184|
|     201112|City of London|       184|
|     201011|City of London|       182|
|     200910|City of London|       175|
|     200809|City of London|       179|
|     200708|City of London|       184|
|     200607|City of London|       177|
+-----------+--------------+----------+
```

2.ii.v. **PART A – 2:**

search the dataset by school type, showing the total number of pupils who were given authorised absences because of medical appointments or illness in the time period 2017-2018. – authorised_absence.py

This python file defines a function **auth_leave_schooltype()** that takes two arguments: **schoolType** and **df**. The function filters a DataFrame **df** to include only the rows where the value of the **school_type** column is in the list **schoolType**. It then filters the resulting DataFrame to include only the rows where the **geographic_level** column has the value "National" and the **time_period** column has the value "201718".

After that, the function adds a new column to the DataFrame called "Total_auth_absentees", which is the sum of the columns "sess_auth_appointments" and "sess_auth_illness". Finally, the function selects only the columns "time_period", "school_type", and "Total_auth_absentees" and returns the resulting DataFrame.

```
    Enter your choice (1, 2, 3, 4, 5, 6)
        1. search the dataset by the local authority
        2. search the dataset by school type
        3. search for all unauthorised absences in a certain year
        4. List the top 3 reasons for authorised absences in each year
        5. Compare two local authorities for any session in a given year
        6. Chart/explore the performance of regions in England from 2006-2018
        7. Quit

Input :  2
Enter the school type: Total
+-----------+-----------+--------------------+
|time_period|school_type|Total_auth_absentees|
+-----------+-----------+--------------------+
|201718     |Total      |70891284            |
+-----------+-----------+--------------------+
```

2.ii.vi. **PART A – 3:**

search for all unauthorised absences in a certain year, broken down by either region name or local authority name – unauth_absence.py:

This code defines a function **unauth_absentees()** that takes three parameters: **search_year**, **search_region**, and **df** (a dataframe). The purpose of this function is to filter the **df** dataframe to return the number of unauthorised absentees for a given region and year.

The function first applies filters to the dataframe based on the **search_year** and **search_region** parameters, using the **filter()** method of the dataframe. It also adds an additional filter to only consider the **Total** school type (as it contains the details of all the schools in the region and we do not want to split it down further).

Then, the function checks the **search_region** parameter to determine which columns to select in the resulting dataframe. If the **search_region** is **Regional**, it selects the **region_name** and **sess_unauthorised** columns. If the **search_region** is **Local authority**, it selects the **la_name** and **sess_unauthorised** columns.

Finally, the function returns the resulting dataframe, which contains the filtered and selected data.

```
Input :  3
Enter the time-period: 200809
Enter the level to be searched for (Regional or Local authority): Regional
+--------------------+-----------------+
|         region_name|sess_unauthorised|
+--------------------+-----------------+
|          North East|           903232|
|          North West|          2938008|
|Yorkshire and the...|          2459468|
|       East Midlands|          1697444|
|       West Midlands|          2272502|
|     East of England|          2000460|
|          South East|          2831955|
|          South West|          1480523|
|        Inner London|          1273235|
|        Outer London|          1967713|
+--------------------+-----------------+
```

2.ii.vii. **PART A – 4:**

List the top 3 reasons for authorised absences in each year

This is a Python function that takes a DataFrame **df** and a SparkSession **spark** as inputs and returns two outputs: a pretty table and a Spark DataFrame. The purpose of the function is to find the top three reasons for authorised absences in schools at the national level, based on data from the input DataFrame.

The first step of the function is to filter the input DataFrame by selecting only rows where the geographic level is "National" and the school type is "Total". This is taken into consideration as we are only focusing on 3 reasons of unauthorized absence and taking geographic level as National covers all the schools. It then selects the columns that begin with 'sess_auth_', and drops the 'sess_auth_totalreasons' column. Selecting columns beginning with sess_auth covers all the authorized sessions including the total one. Since we want to find the top 3 authorised reasons we drop the total authorized session.

The function then iterates over the rows of the filtered DataFrame, creating a dictionary for each row that contains the time period and the top three reasons for authorised absences in that time period. To find the top three reasons, the function sorts the row dictionary by the values in descending order and selects the top three keys (which correspond to the reasons for authorised absences).

A spark DataFrame is then created with top_3_list obtained which is returned.

Select an option

List the top 3 reasons for authorised absences in each year

	Reason1	Reason2	Reason3	Time Period
0	sess_auth_illness	sess_auth_other	sess_auth_appointments	201,819
1	sess_auth_illness	sess_auth_other	sess_auth_appointments	201,718
2	sess_auth_illness	sess_auth_appointments	sess_auth_other	201,617
3	sess_auth_illness	sess_auth_appointments	sess_auth_other	201,516
4	sess_auth_illness	sess_auth_appointments	sess_auth_other	201,415
5	sess_auth_illness	sess_auth_appointments	sess_auth_other	201,314
6	sess_auth_illness	sess_auth_holiday	sess_auth_appointments	201,213
7	sess_auth_illness	sess_auth_holiday	sess_auth_appointments	201,112
8	sess_auth_illness	sess_auth_holiday	sess_auth_other	201,011
9	sess_auth_illness	sess_auth_other	sess_auth_holiday	200,910

2.ii.viii. PART B – 1:

Allow a user to compare two local authorities of their choosing in a given year. – compare_two_la.py

This code defines a function called **compare_local_auth** that takes four arguments: **year**, **la_list_compare**, **toCompare**, and **df**.

year is an integer representing the academic year for which the data is to be analyzed. **la_list_compare** is a list containing two strings, representing the names of two local authorities that are to be compared. **toCompare** is an integer representing which variable to compare, with 1 indicating the number of authorized reasons sessions, 2 indicating the overall absence rate, 3 indicating the authorized absence rate, and 4 indicating the unauthorized absence rate. **df** is a DataFrame containing the pupil absence data.

The function first assigns **la1** and **la2** from **la_list_compare** and then determines which variable to compare based on the value of **toCompare**. It filters the **df** DataFrame to include only rows with the specified **year**, local authority, and school type. It then selects the **la_name** and specified variable columns and creates a new DataFrame **comparing_df**.

The function then creates a new figure using **plt.subplots** and assigns it to **fig** and **ax**. It extracts the values of the specified variable for **la1** and **la2** from **comparing_df** and removes any null values. It then creates a bar chart with two bars, one for **la1** and one for **la2**, using the extracted values. The title and y-axis label of the plot are set based on the **toCompare** argument. The legend is also created and positioned in the upper right corner of the plot.

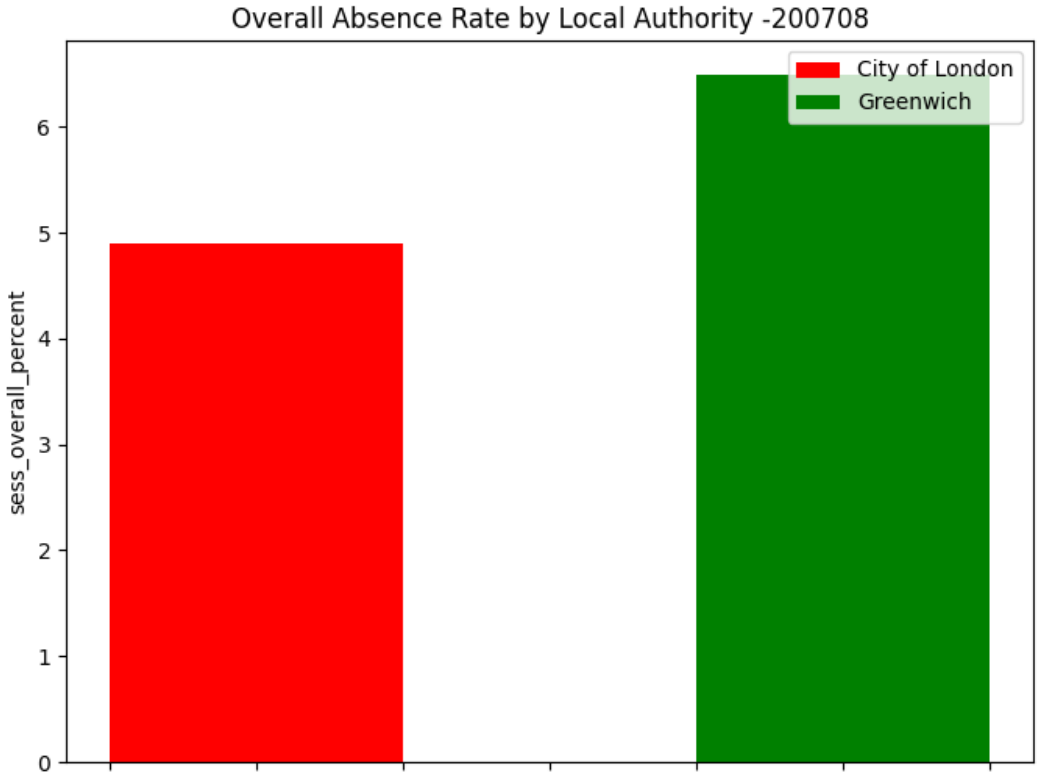
Finally, the function returns the **fig** object.

The variables sess_authorised, sess_overall_percent, sess_authorised_percent, and sess_unauthorised_percent are important because they provide information on different aspects of pupil absence in schools in England.

sess_authorised represents the number of authorised absence sessions, which refers to absences that are approved by the school, such as for medical or family reasons. This on a large scale could give insight to a particular local authorities' health/disease cases (if it comes to that).
sess_overall_percent represents the overall absence rate, which is the percentage of all possible sessions that were missed due to any reason, whether authorised or unauthorised. This provides an insight onto how many absence session the specific local_authority had.
sess_authorised_percent represents the authorised absence rate, which is the percentage of all possible sessions that were missed due to authorised reasons.
sess_unauthorised_percent represents the unauthorised absence rate, which is the percentage of all possible sessions that were missed due to unauthorised reasons, such as skipping school without permission.

Comparing various local authorities on these four bases can give us important information about how different regions are performing in terms of reducing pupil absence. For example, if one local authority has a high overall absence rate but a low unauthorised

absence rate, it may indicate that the schools in that region are doing a good job of managing authorised absences but need to focus on reducing unauthorised absences. On the other hand, if another local authority has a high unauthorised absence rate, it may indicate a need for stronger enforcement of attendance policies or additional support for students who are struggling to attend school regularly. By comparing data across multiple regions, policymakers and educators can identify best practices and areas for improvement to help reduce pupil absence and improve educational outcomes.



2.ii.ix. PART B – 2:

Chart/explore the performance of regions in England from 2006-2018 – region_performance_2006_18.py

The code provided is a Python script that generates two plots for regional attendance and attendance rates for a given dataset.

The **regional_performance_2006_2018** function takes a PySpark DataFrame as an input, filters it to extract data for England region and Total school type, and computes

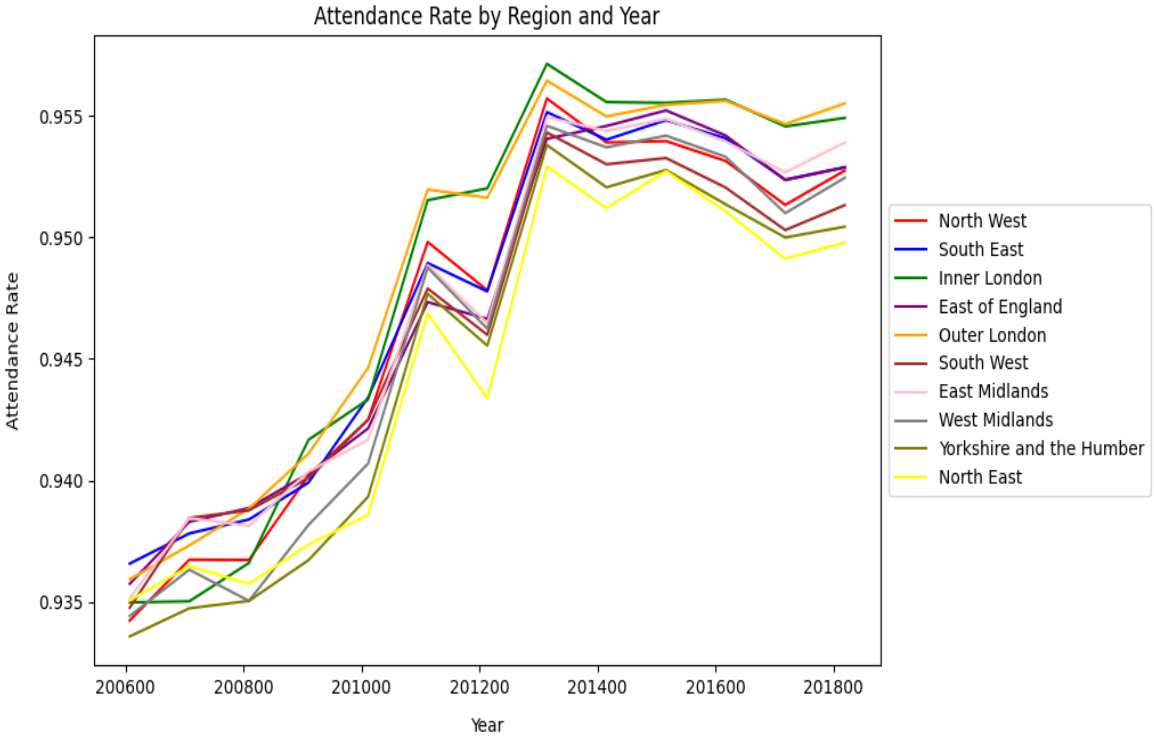
attendance and attendance rates for each region and time period. Then, it calls two plotting functions, **plot_for_attendance** and **plot_for_attendance_rate**, which create two plots respectively.

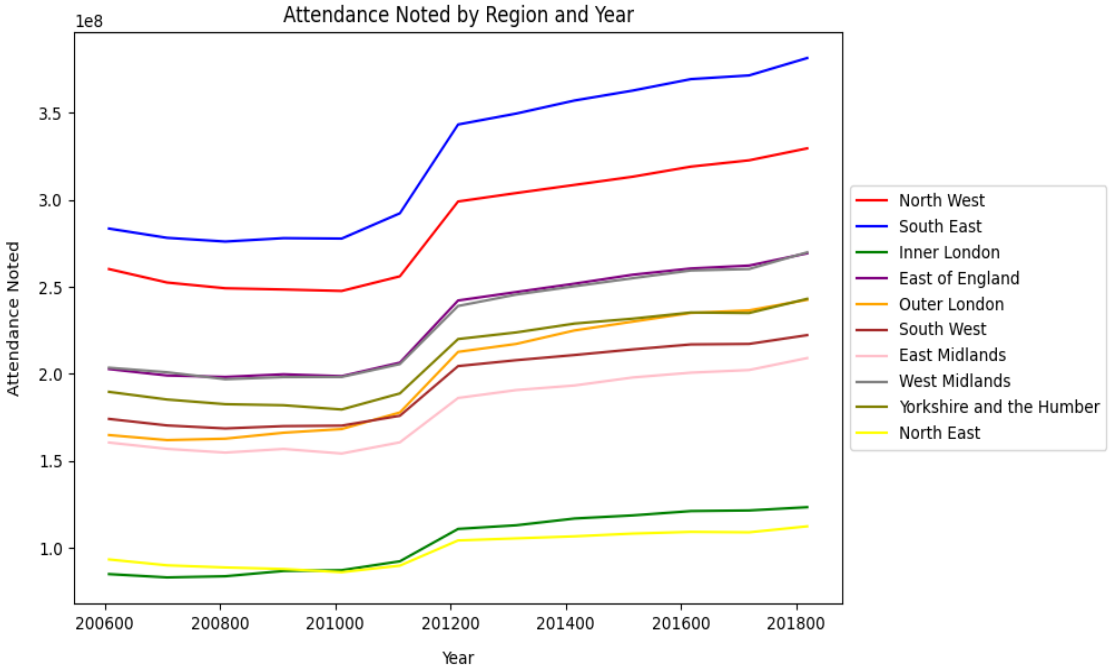
The total attendance is calculated by subtracting overall sessions from sessions possible. While the attendance rate is calculated by dividing the total attendance by sessions possible. This gives a percentage (multiplied by 100) of attendance rate of a given region.

The **plot_for_attendance** function creates a plot of the attendance noted over time for each region. It first creates a figure and axis object using **plt.figure** and **fig1.add_subplot**, respectively. It then iterates over the list of regions passed as input and filters the input DataFrame to extract data for the current region. It then converts the PySpark DataFrame to a Pandas DataFrame using **toPandas**, selects the relevant columns, and plots the attendance noted over time using **ax1.plot**. Finally, it sets the x and y axis labels, title, and legend, and adjusts the legend position using **ax1.set_xlabel**, **ax1.set_ylabel**, **ax1.set_title**, **ax1.legend**, and **plt.subplots_adjust**, respectively.

The **plot_for_attendance_rate** function creates a plot of the attendance rate over time for each region. It follows the same structure as **plot_for_attendance**, but plots the attendance rate over time using **ax2.plot** instead. The **plot_for_attendance** function creates a line plot of attendance data for each region over the years 2006 to 2018. It loops over each region in the **data** list, filters the **filtered_df** DataFrame to only include rows with data for that region, and creates a Pandas DataFrame from the resulting PySpark DataFrame. It then plots the attendance data for that region using a unique color from the **colors** list. Finally, it adds a legend to the plot and returns the resulting figure.

The two plots are then returned as outputs of the **regional_performance_2006_2018** function.


Attendance Rate by Region and Year
Attendance Rate
0.955
0.950
0.945
0.940
0.935
200600
200800
201000
201200
201400
201600
201800
Year
North West
South East
Inner London
East of England
Outer London
South West
East Midlands
West Midlands
Yorkshire and the Humber
North East


Attendance Noted by Region and Year
1e8
Attendance Noted
3.5
3.0
2.5
2.0
1.5
1.0
200600
200800
201000
201200
201400
201600
201800
Year
North West
South East
Inner London
East of England
Outer London
South West
East Midlands
West Midlands
Yorkshire and the Humber
North East

Data Review

Q: Are there any regions that have improved in pupil attendance over the years?

According to the attendance_rate plot we can see that eventhough all the regions have seen an increase in the attendance, inner London and outer London show the maximum increase throughout the years.

Q. Are there any regions that have worsened?

The south west region which was on the top of attendance rate during the starting year has came down to the bottom of the group in case of attendance by 2018.

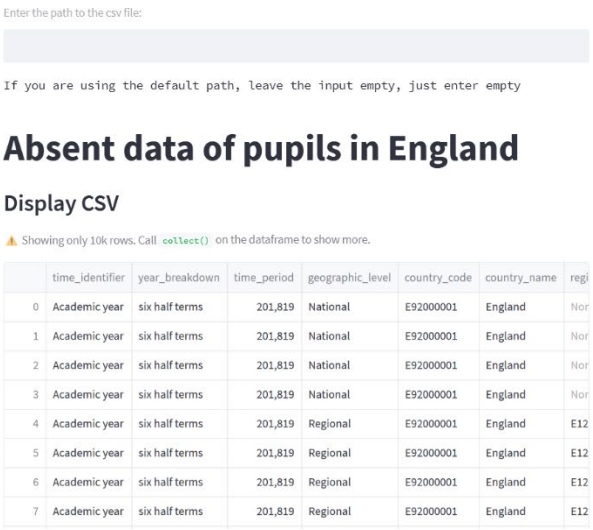
Q. Which is the overall best/worst region for pupil attendance?

The overall worst region for pupil attendance is found to be North East region as they stay on the bottom line among the group eventhough the overall attendance rate of all the regions increase.

The overall best region is found to be Outer London which has consistently stayed on the top.

2.ii.x **PART C – webapp.py**

Web application with streamlit library. Streamlit library is an easy framework which was incorporated into the existing code without no hardships. The python code for the webapp is similar to that of the console app with tiny differences such as the usage of elif instead of match function.



	time_identifier	year_breakdown	time_period	geographic_level	country_code	country_name	regi
0	Academic year	six half terms	201,819	National	E92000001	England	Nor
1	Academic year	six half terms	201,819	National	E92000001	England	Nor
2	Academic year	six half terms	201,819	National	E92000001	England	Nor
3	Academic year	six half terms	201,819	National	E92000001	England	Nor
4	Academic year	six half terms	201,819	Regional	E92000001	England	E12
5	Academic year	six half terms	201,819	Regional	E92000001	England	E12
6	Academic year	six half terms	201,819	Regional	E92000001	England	E12
7	Academic year	six half terms	201,819	Regional	E92000001	England	E12

3.Conclusion

The PySpark project was successful in achieving its goals through the use of PySpark's libraries and APIs for efficient data cleaning, transformation, and analysis. The project's output included visualizations that provided valuable insights for data-driven decision making. It also provided an opportunity to gain practical experience in using PySpark, a valuable skill in the data-driven world. Overall, the project was a great learning experience that demonstrated the potential of PySpark in handling large datasets.