Exploratory data analysis and Data visualisation

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
```

Public Health England – COVID-19 data --> ¶

File contains counts across England of (a) new COVID-19 cases, (b) tests performed, (c) new hospital admissions and (d) individuals vaccinated (with a second dose).

General description of the dataset, date-range, total number of observations and the number of missing data per variable

Loading the data

Public Health England – COVID-19 data has 683 observation points (rows) and 8 variables (columns). These include areaCode, areaName, areaType, date, newAdmissions, newCasesByPublishDate, newPeopleVaccinatedBySecondDoseByPublishDate, newTestsByPublishDate

In [2]:
```python
nhs_dataset = pd.read_csv('nation_2021-12-13.csv') #Dataset is loaded using read_csv function of pandas library
```

In [3]:
```python
nhs_dataset
```

Out[3]:

	areaCode	areaName	areaType	date	newAdmissions	newCasesByPublishDate	newPeopleVaccinatedSecondDoseByPublishDate	newTestsByPublishDate
0	E92000001	England	nation	2021-12-13	NaN	44931	NaN	NaN
1	E92000001	England	nation	2021-12-12	NaN	40713	24900.0	1140279.0
2	E92000001	England	nation	2021-12-11	696.0	48540	34061.0	927390.0
3	E92000001	England	nation	2021-12-10	707.0	48908	28294.0	1082643.0
4	E92000001	England	nation	2021-12-09	799.0	43550	29840.0	1190280.0
...	...	...	...	...	...	...	...	...
678	E92000001	England	nation	2020-02-04	NaN	0	NaN	NaN
679	E92000001	England	nation	2020-02-03	NaN	0	NaN	NaN
680	E92000001	England	nation	2020-02-02	NaN	0	NaN	NaN
681	E92000001	England	nation	2020-02-01	NaN	0	NaN	NaN
682	E92000001	England	nation	2020-01-31	NaN	2	NaN	NaN

683 rows × 8 columns

In [4]:
```python
nhs_dataset.shape
```

Out[4]:
```
(683, 8)
```

General description of dataset

Below is the description of the dataset showing the minimum value, maximum value, mean, standard deviation, median and quartile values for numeric variables

In [5]:
```python
nhs_dataset.describe() #using describe function of pandas
```

Out[5]:

	newAdmissions	newCasesByPublishDate	newPeopleVaccinatedSecondDoseByPublishDate	newTestsByPublishDate
count	633.000000	683.000000	336.000000	5.180000e+02
mean	820.966825	13547.235725	115650.023810	6.059462e+05
std	825.255043	14398.783009	120832.328111	3.521440e+05
min	25.000000	0.000000	0.000000	7.430000e+04
25%	188.000000	1349.500000	20274.500000	2.587025e+05
50%	648.000000	5632.000000	74276.000000	6.185520e+05
75%	1055.000000	24848.000000	165817.000000	8.787355e+05
max	4134.000000	61757.000000	508013.000000	1.846349e+06

Date range

The date range for the dataset is from 31st January 2020 to 13th December 2021

In [6]:
```python
startDate = min(nhs_dataset['date'])
endDate = max(nhs_dataset['date'])
print('Date range : ', startDate, 'to', endDate)

#using min and max function on a column to calculate the lowest and highest values in a pandas dataset
```

```
Date range :  2020-01-31 to 2021-12-13
```

Total number of observations

In [7]:
```python
numObservations = len(nhs_dataset)  #len function of pandas
print(numObservations)
```

```
683
```

Number of missing data per variable

As calculated below, there are **no null values** for columns: **areaCode**, **areaName**, **areaType**, **date**, **newCasesByPublishDate**

However, 'newAdmissions', 'newPeopleVaccinatedSecondDoseByPublishDate', 'newTestsByPublishDate' have 50, 347 and 165 null values respectively

In [8]:
```
nhs_dataset.isnull().sum() #using .isnull.sum() function of pandas
```

Out[8]:
```
areaCode                                      0
areaName                                      0
areaType                                      0
date                                          0
newAdmissions                                50
newCasesByPublishDate                         0
newPeopleVaccinatedSecondDoseByPublishDate  347
newTestsByPublishDate                       165
dtype: int64
```

Converting date from Dtype 'object' to 'datetime'

The dataset information below gives the Dtype (datatype) of columns and we can see that Dtype of column 'date' is 'object'.

A new column 'newDate' which is datetime version of column 'date' is created to get the dates in datetime format

In [9]:
```
nhs_dataset['newDate'] = pd.to_datetime(nhs_dataset['date']) #using pd.to_datetime function of pandas
nhs_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683 entries, 0 to 682
Data columns (total 9 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   areaCode                                    683 non-null    object
 1   areaName                                    683 non-null    object
 2   areaType                                    683 non-null    object
 3   date                                        683 non-null    object
 4   newAdmissions                               633 non-null    float64
 5   newCasesByPublishDate                       683 non-null    int64
 6   newPeopleVaccinatedSecondDoseByPublishDate  336 non-null    float64
 7   newTestsByPublishDate                       518 non-null    float64
 8   newDate                                     683 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 48.1+ KB
```

Now, the dataset is sorted in ascending order of 'newDate' as it will produce more meaningful plots for the timeline trends of the variables.

In [10]:
```
nhs_dateSorted = nhs_dataset.sort_values('newDate') #using sort_values function of pandas
nhs_dateSorted.head()
```

Out[10]:

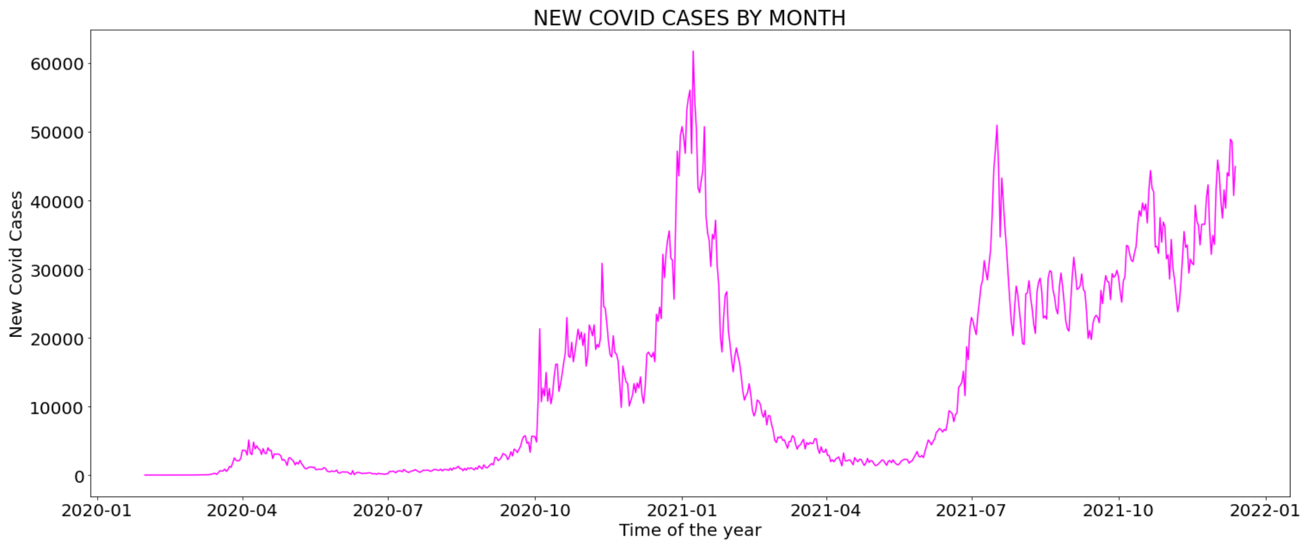
	areaCode	areaName	areaType	date	newAdmissions	newCasesByPublishDate	newPeopleVaccinatedSecondDoseByPublishDate	newTestsByPublishDate	newDate
682	E92000001	England	nation	2020-01-31	NaN	2	NaN	NaN	2020-01-31
681	E92000001	England	nation	2020-02-01	NaN	0	NaN	NaN	2020-02-01
680	E92000001	England	nation	2020-02-02	NaN	0	NaN	NaN	2020-02-02
679	E92000001	England	nation	2020-02-03	NaN	0	NaN	NaN	2020-02-03
678	E92000001	England	nation	2020-02-04	NaN	0	NaN	NaN	2020-02-04
...	...	...	...	...	...	...	...	...	...
4	E92000001	England	nation	2021-12-09	799.0	43550	29840.0	1190280.0	2021-12-09
3	E92000001	England	nation	2021-12-10	707.0	48908	28294.0	1082643.0	2021-12-10
2	E92000001	England	nation	2021-12-11	696.0	48540	34061.0	927390.0	2021-12-11
1	E92000001	England	nation	2021-12-12	NaN	40713	24900.0	1140279.0	2021-12-12
0	E92000001	England	nation	2021-12-13	NaN	44931	NaN	NaN	2021-12-13

683 rows × 9 columns

Plots for the timeline showing all four covid variables

New Covid Cases By Month

In [11]:
```python
plt.rcParams['figure.figsize'] = (25, 10)
plt.rcParams.update({'font.size': 20})
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newCasesByPublishDate'], color = '#ff00ff')
plt.title('NEW COVID CASES BY MONTH')
plt.xlabel('Time of the year')
plt.ylabel('New Covid Cases')
plt.show()
```



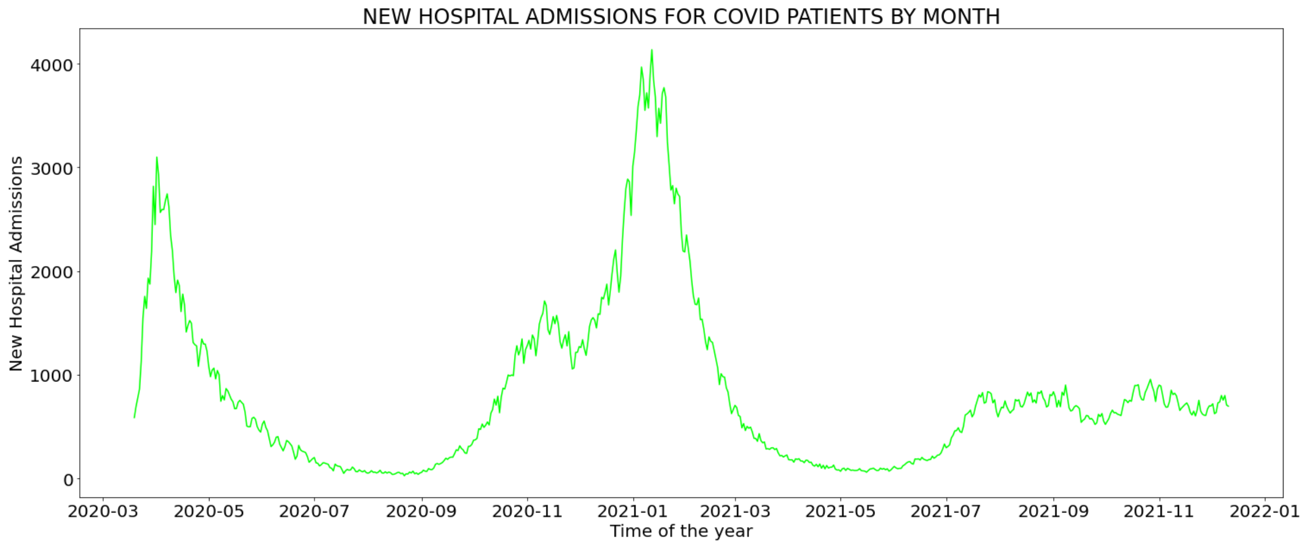
Discussion - Graph 1

The line graph for Public Health England Covid -19 data for the given dataset covers new covid cases between Jan 2020 to Dec 2021.

- New covid cases were detected starting at the end of Jan 2020 and increased slightly with approx 5000 new covid cases detected aorund Apr 2020. However, this number then decreased a little and remained mostly stable until Oct 2020.
- This number saw first ever notable increase after Oct 2020. New covid cases peaked during the winter months (Jan 2021).
- The number of new cases then began decreasing after Jan 2021, with fewer cases seen around April 2021. The number remained mostly stable until Jun 2021.
- The second covid wave hit in and the numbers spiked again in Jul 2021. New covid cases have remained quite high until the end month of our reporting period.

New Hospital Admissions By Month

In [12]:
```python
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newAdmissions'], color = '#00FF00')
plt.title('NEW HOSPITAL ADMISSIONS FOR COVID PATIENTS BY MONTH')
plt.xlabel('Time of the year')
plt.ylabel('New Hospital Admissions')
plt.show()
```



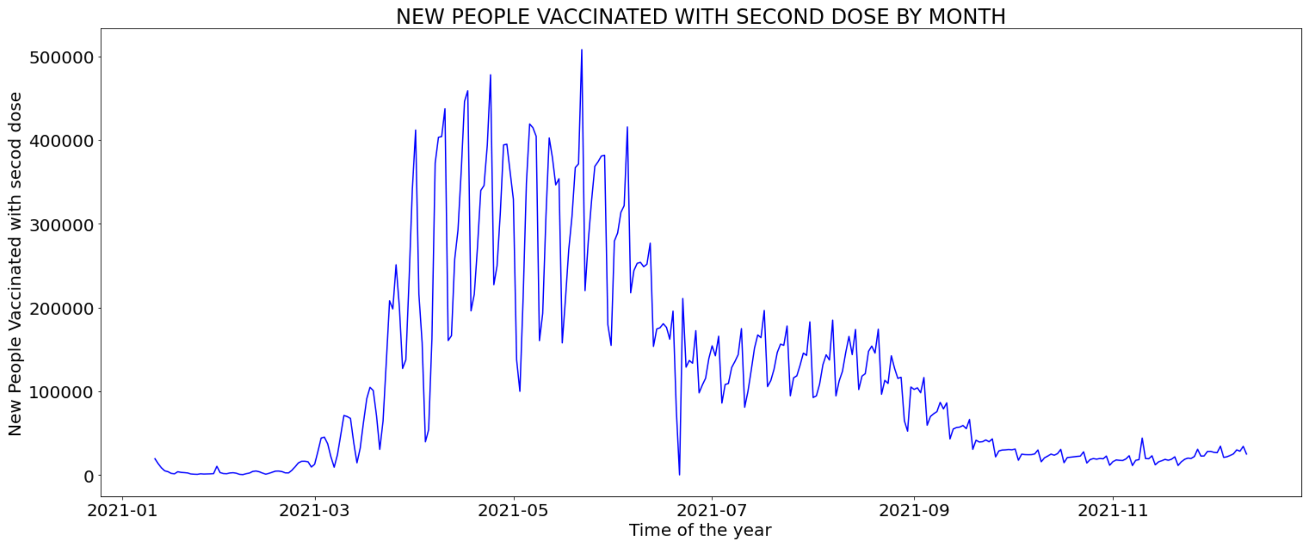
Discussion - Graph 2

The line graph for Public Health England Covid -19 data for the given dataset covers new hospital admissions between Jan 2020 to Dec 2021. (The reason for graph timeline starting from Mar 2020 is because there were possibly no covid related hospital admsions before that time as the pandemic had just started in England.)

- New hospital admsions saw a sharp increase in Mar/Apr 2020 when the covid -19 pandemic hit in. This number went down slowly to very few admissions near Sep 2020.
- After Sep 2020, there was a greatest rise in hospital admissions for the overall time period in the dataset, with a jump of approximately 4000 new admissions in the month of Jan 2021. This was expected because number of new covid cases also peaked during the same time as seen in the previous graph.
- The number of admissions came down considerably during the first quarter of 2021 (Jan to Mar) and then remained stable until Jul 2021.
- There was a slight increase in hospital admissions after Jul 2021 eventually levelling off until Dec 2021. This is also consistent with a slight increase in new covid cases around Jul 2021.

New People Vaccinated With Second Dose By Month

In [13]:
```python
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newPeopleVaccinatedSecondDoseByPublishDate'], color = '#0000FF')
plt.title('NEW PEOPLE VACCINATED WITH SECOND DOSE BY MONTH')
plt.xlabel('Time of the year')
plt.ylabel('New People Vaccinated with secod dose')
plt.show()
```



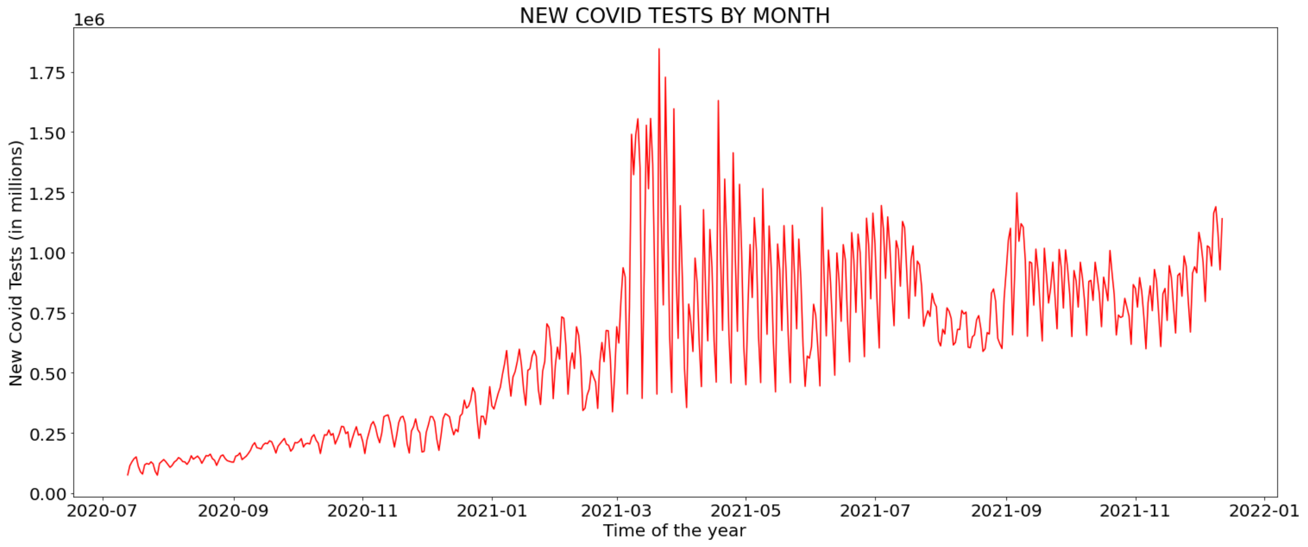
Discussion - Graph 3

The line graph for Public Health England Covid -19 data for the given dataset covers new people vaccinated with second dose between Jan 2021 to Dec 2021.

- Second dose vaccinations did not start until Jan 2021. Hence, we do not have data for the variable 'newPeopleVaccinatedSecondDoseByPublishDate' before that time.
- The number of new second dose vaccinations were quite low during the first quarter of 2021. However, this number rose rapidly after Mar 2021 and remained elevated for the whole second quarter of 2021(until Jun 2021). This also justifies that why the number of new hospital admissions were less during the second wave. Most people were already vaccinated with their first doses and got their second doses before second wave hit in Jul 2021.
- Overall, the number of new second dose vaccinations was lower in third quarter of 2021 than second quarter. The number further decreased a little and then remained stable until Dec 2021
- This decrease is reasonable as most of the England population is now doubly vaccinated and have started looking forward to the booster doses.

New Covid Tests By Month

In [14]:
```python
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newTestsByPublishDate'], color = '#FF0000')
plt.title('NEW COVID TESTS BY MONTH')
plt.xlabel('Time of the year')
plt.ylabel('New Covid Tests (in millions)')
plt.show()
```



Discussion - Graph 4

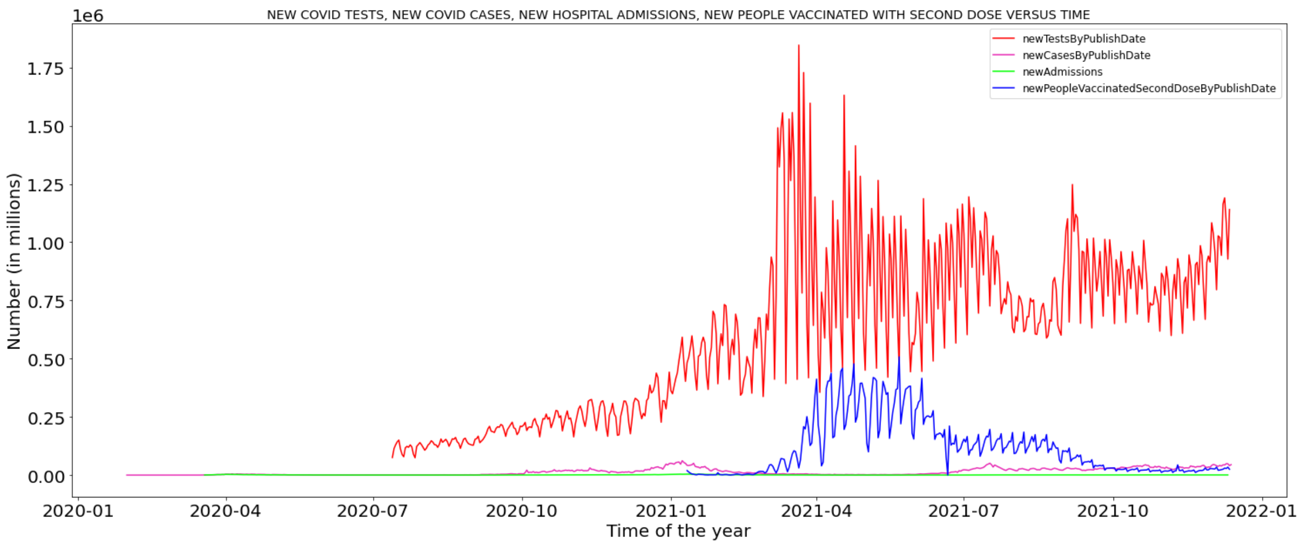
The line graph for Public Health England Covid -19 data for the given dataset covers new covid tests between Jan 2020 to Dec 2021.

- The graph does not show the data for months before Jul 2020 because the number of covid tests conducted in months before that time were possibly less than than 0.1 million. Hence, those numbers could not be plotted with the number units of length 'le6'.
- There is an overall increasing trend in the new covid tests conducted in England each month during the covid pandemic.
- The new covid tests conducted during the second wave was much higher than the number of tests conducted during first wave.
- The data shows an increased testing capacity as the country fight the pandemic.

Combined graph for all four variables

In [15]:
```python
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newTestsByPublishDate'], color = '#FF0000', label = 'newTestsByPublishDate')
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newCasesByPublishDate'], color = '#e82cb5', label = 'newCasesByPublishDate')
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newAdmissions'], color = '#00FF00', label = 'newAdmissions')
plt.plot(nhs_dateSorted['newDate'], nhs_dateSorted['newPeopleVaccinatedSecondDoseByPublishDate'], color = '#0000FF',
         label = 'newPeopleVaccinatedSecondDoseByPublishDate')
plt.xlabel('Time of the year')
plt.ylabel('Number (in millions)')
plt.rcParams['figure.figsize'] = (18, 10)
plt.rcParams.update({'font.size': 12})
plt.title('NEW COVID TESTS, NEW COVID CASES, NEW HOSPITAL ADMISSIONS, NEW PEOPLE VACCINATED WITH SECOND DOSE VERSUS TIME')
plt.legend()
plt.show()

```



NOTE:

The above combined graph is less intelligible because y axis is in millions and not all variables have numbers in millions.

For example: Number of new covid cases and number of new hospital admissions are in thousands rather than millions

Historical releases of transmission rate data for a specific NHS region can be loaded as a csv file directly from URLs using the Public Health England API. In the exemplar URL given below the date is given as release = 2021-09-01 and the NHS Region is defined as areaCode = E40000005.

https://api.coronavirus.data.gov.uk/v2/data?areaType=nhsRegion&areaCode=E40000005&metric=transmissionRateMax&metric=transmissionRateMin&format=csv&release=2021-09-01 (https://api.coronavirus.data.gov.uk/v2/data?areaType=nhsRegion&areaCode=E40000005&metric=transmissionRateMax&metric=transmissionRateMin&format=csv&release=2021-09-01)

Creating a function that takes a date and an area code as inputs and

(a) creating the corresponding URL to link to these data,

(b) retrieving and loading the data into a pandas dataframe and

(c) plotting a figure showing the minimum and maximum transmission rates over time (R value).

Ensuring that function can handle errors if a wrong URL string is provided. Testing the function, by running a code with the North West code “E40000010” using the historic data released on the 26th of February 2021. Comparing it to a plot for the South East, using the same date.

URL Creation, Data Retrieval and Loading, Plotting function

Date Verification

- dateVerification function is defined for verification of date when a user inputs date.
- The URL and dataset has date in YYYY-MM-DD format. Hence, user is required to enter date in the same format. If any other format is provided, URL will not be recognized or read into the corresponding dataset and this will result in an error.
- Date verification step allows to handle errors in input date and suggest user to enter date in the required format.

In [16]:
```python
def dateVerification(date):
    try:
        datetime.datetime.strptime(date, '%Y-%m-%d')
        return True
    except:
        print('Invalid date format', 'Please enter date in YYYY-MM-DD format')
        return False
```

Area Code Verification

- areaCodeVerification function is defined for verification of areaCode when a user inputs areaCode.
- There are a total of 10 NHS England Regions - Areas with specific areacodes. If user provides any other value for areacode, it will not be recognized in URL or read into the corresponding dataset and will result in an error.
- Areacode verification step allows to handle errors in input areaCode and suggest user to enter a valid areaCode.

In [17]:
```python
def areaCodeVerification(areaCode):
    area = {'E40000001' : 'North of England', 'E40000002' : 'Midlands and East of England',
            'E40000003' : 'London', 'E40000004' : 'South of England', 'E40000005' : 'South East',
            'E40000006' : 'South West', 'E40000007' : 'East of England',  'E40000008' : 'Midlands',
            'E40000009' : 'North East and Yorkshire', 'E40000010' : 'North West'}
    if areaCode in area:
        return True
    else:
        print('Invalid Area Code')
        return False
```

URL Verification

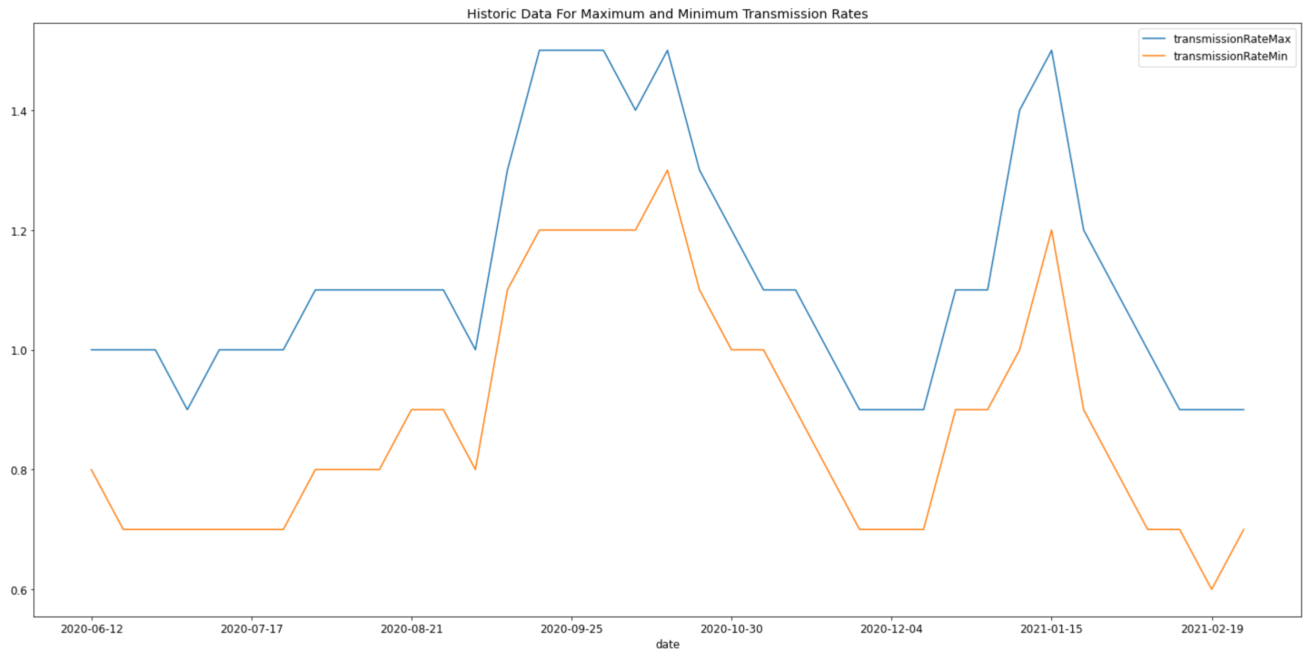
- createURLAndLoadData function is defined for creation of url string when user inputs date and areaCode
- Firstly, the url string will be formatted with the input date and areaCode
- Then,
  - If dateVerification and areaCodeVerification are both True, it will enter into 'try' function to read the URL and load the dataset
    - If the data exists for that URL, it will move further by plotting the graph for historical releases of transmission rate data for that NHS region.
    - Else if, data or URL does not exist for input date and areaCode, pandas function will fail to read it and an exception will be raised which will print 'URL does not exist'
  - If any of the dateVerification and areaCodeVerification is False, it will print whichever is invalid

In [18]:
```python
def createURLAndLoadData(date, areaCode):
    url = 'https://api.coronavirus.data.gov.uk/v2/data?areaType=nhsRegion&areaCode={}&metric=transmissionRateMax&metric=transmissionRateMi
    if dateVerification(date) and areaCodeVerification(areaCode):
        try:
            dataset = pd.read_csv(url).sort_values('date')
            dataset.plot(x = 'date', y = ['transmissionRateMax', 'transmissionRateMin'], title = 'Historic Data For Maximum and Minimum Tra
        except:
            print('URL does not exist')
```

Plot for North West

In [19]:
```python
date = input('Enter date: ')
areaCode = input('Enter Area code: ')
createURLAndLoadData(date, areaCode)
```

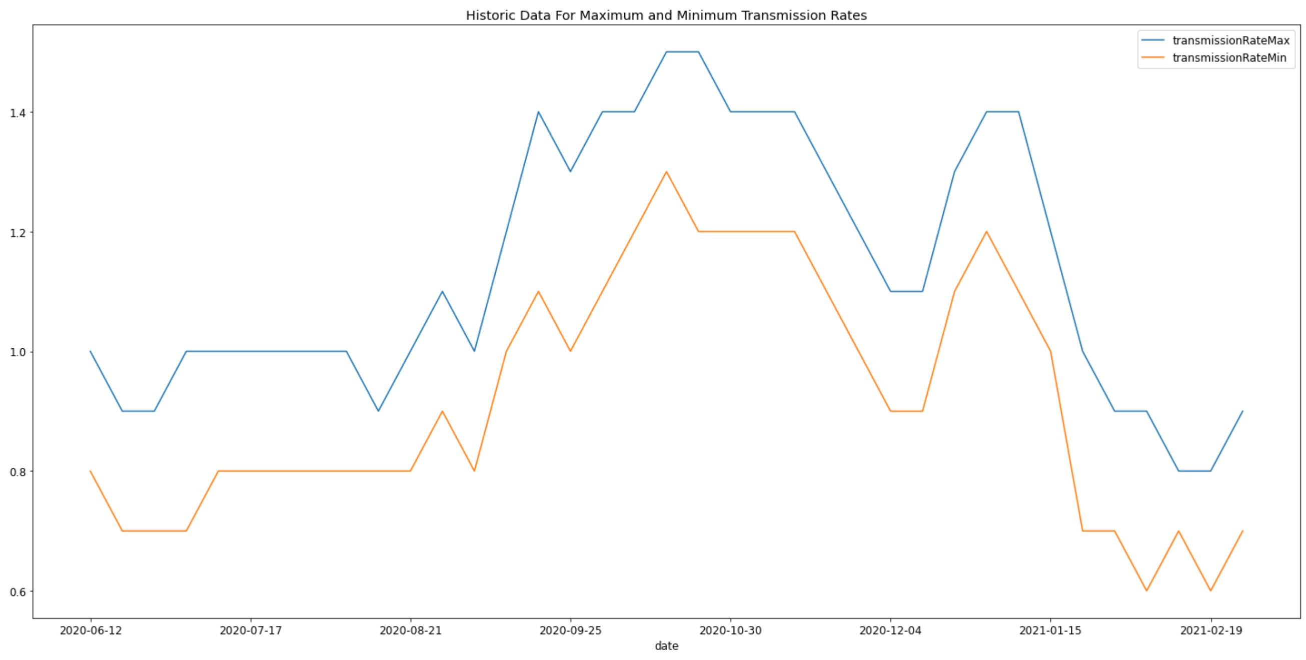
```
Enter date: 2021-02-26
Enter Area code: E40000010
```



Plot for South East

In [20]:
```
date = input('Enter date: ')
areaCode = input('Enter Area code: ')
createURLAndLoadData(date, areaCode)
```

```
Enter date: 2021-02-26
Enter Area code: E40000005
```



R value is the average number of secondary infections produced by a single infected person.

- For both North West and South East, maximum and minimum transmission rates are similar for the time period between Jun 2020 to Feb 2021.
- The highest R value for the given time period is around Oct 2020 with max transmission rate just above 1.4 and minimum transmission rate approx 1.3 for both the areas.
- It is clear from the both the graphs that transmission rates were high in Oct 2020 and Jan 2021 with values greater than 1.0, meaning the epidemic was growing in the England. This is logical because new covid cases were also high during these months (as shown in previous plots). Hence, chances for a single infected person to produce secondary infections were more.

Dental data in England

Evaluating the usage of dental practices in England. For this purpose, data from NHS Digital is being used from June 2020 to June 2021. It contain the NHS dental statistics for adults and child patients seen by age.

Focusing only on the data for the period ending (PSEEN_END_DATE) on the 30th of June 2021, finding top 5 dental practices in London that saw the most children (younger than 18). Create a plot for visualising results

Reading the dataset for period ending (PSEEN_END_DATE) on the 30th of June 2021 and printing first 5 rows of dataset

In [21]:
```python
datasetJanJun21 = pd.read_csv("nhs-dent-stat-eng-jan-jun-21-anx3-ps-prac.csv")
datasetJanJun21.head()
```
Out[21]:

	PSEEN_END_DATE	GEOG_TYPE	PRACTICE_CODE	PRACTICE_NAME	PRAC_POSTCODE	CCG_CODE	CCG_ONS_CODE	CCG_NAME	LA_CODE	LA_NAME	REGION_CODE	REGION_ONS
0	2021-01-31	CCG	NaN	NaN	NaN	00L	E38000130	NHS Northumberland CCG	NaN	NaN	Y63	E4
1	2021-01-31	CCG	NaN	NaN	NaN	00L	E38000130	NHS Northumberland CCG	NaN	NaN	Y63	E4
2	2021-01-31	CCG	NaN	NaN	NaN	00L	E38000130	NHS Northumberland CCG	NaN	NaN	Y63	E4
3	2021-01-31	CCG	NaN	NaN	NaN	00L	E38000130	NHS Northumberland CCG	NaN	NaN	Y63	E4
4	2021-01-31	CCG	NaN	NaN	NaN	00L	E38000130	NHS Northumberland CCG	NaN	NaN	Y63	E4

STEPS:

- Filtering our data to identify dental practices with PSEEN_END_DATE = 2021-06-30, PATIENT_TYPE = Child, REGION_NAME = London
- Grouping the new filtered dataset by PRACTICE_NAME
- Sum of values for PATIENTS_SEEN and sorting the sum values in descending order
- Dropping NaN values and plotting the bar chart

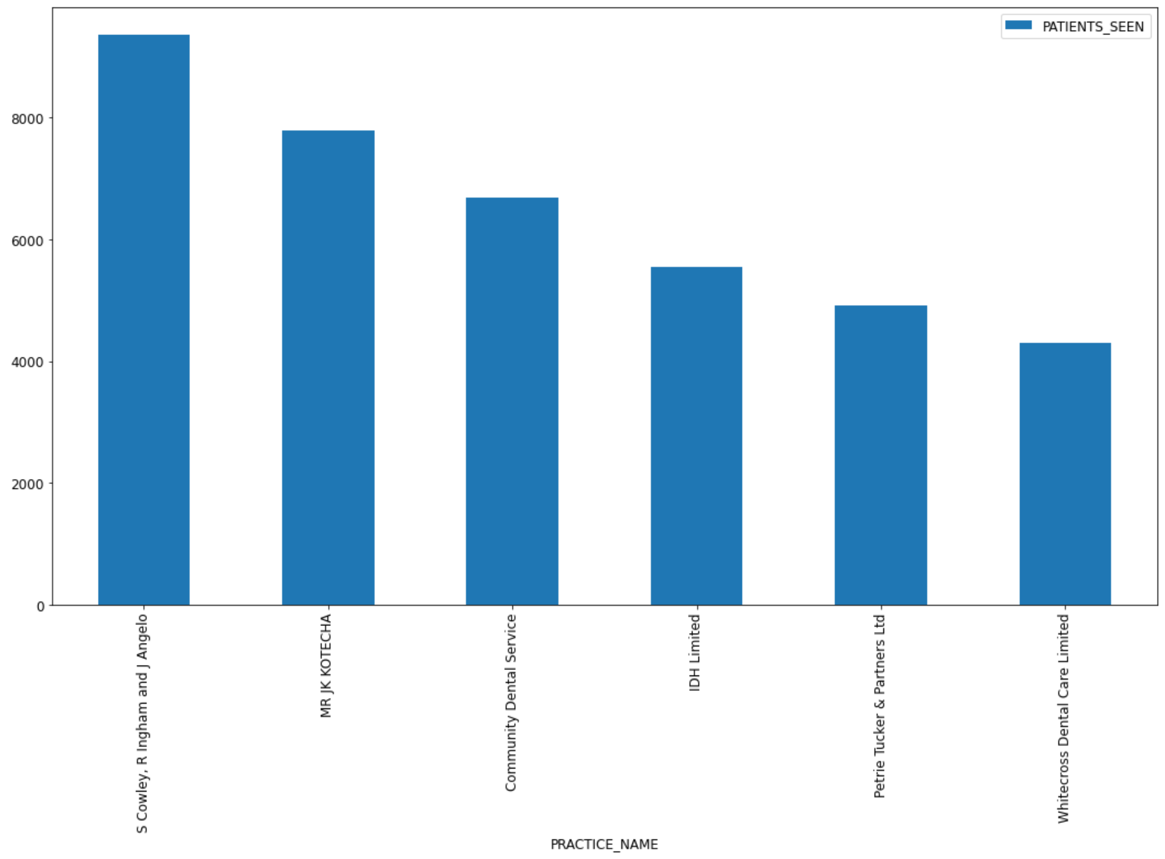
NOTE: Instead of using PRACTICE_CODE, we have used PRACTICE_NAME because multiple PRACTICE_NAME are associated with one PRACTICE_CODE. If we use PRACTICE_CODE, we might identify top 6 dental PRACTICE_CODE which saw highest children in London but the individual number of patients seen in the each PRACTICE_NAME under that PRACTICE_CODE might not represent the highest or top 6 numbers

In [22]:
```python
childLondonJanJun21 = datasetJanJun21[(datasetJanJun21.PSEEN_END_DATE == '2021-06-30') & (datasetJanJun21.PATIENT_TYPE == 'Child') & (datas
top6Practice = childLondonJanJun21.groupby(['PRACTICE_NAME'], dropna = False).sum().sort_values(['PATIENTS_SEEN'], ascending = False).rese
top6ChildPracticeLondon = top6Practice.dropna()
top6ChildPracticeLondon
```
Out[22]:

	PRACTICE_NAME	PATIENTS_SEEN	POPULATION
1	S Cowley, R Ingham and J Angelo	9348	0.0
2	MR JK KOTECHA	7789	0.0
3	Community Dental Service	6682	0.0
4	IDH Limited	5543	0.0
5	Petrie Tucker & Partners Ltd	4911	0.0
6	Whitecross Dental Care Limited	4309	0.0

For the time period ending on 30th Jun 2021 in London, the dental practice named S Cowley, R Ingham and J Angelo saw the most children

In [23]:
```python
top6ChildPracticeLondon.plot.bar(x = 'PRACTICE_NAME', y = 'PATIENTS_SEEN')
```
Out[23]:
```
<AxesSubplot:xlabel='PRACTICE_NAME'>
```



For the same time period ending on the 30th of June 2021, creating a table that lists the following per region:

(A) the number of unique practices,

(B) the mean number of children seen per practice, and

(C) the mean number of adults seen per practice. Discuss your findings.

Part A

Calculating total number of unique practices per region in the dataset

STEPS:

- Filtering our dataset for PSEEN_END_DATE = 2021-06-30
- Grouping the new filtered dataset by REGION_NAME and counting number of unique practices per region by using nunique() function.

In [24]:
```python
numberUniquePractices = datasetJanJun21[(datasetJanJun21.PSEEN_END_DATE == '2021-06-30')]
numberUniquePractices = numberUniquePractices.groupby(['REGION_NAME'])['PRACTICE_NAME'].nunique()
numberUniquePractices
```

Out[24]:
```
REGION_NAME
East of England               704
London                       1110
Midlands                     1081
North East and Yorkshire      861
North West                    832
South East                   1019
South West                    582
Unallocated                    26
Name: PRACTICE_NAME, dtype: int64
```

Part B & C

Calculating Sum of PATIENTS_SEEN for both Adult and Child per region

STEPS:

- Filtering our dataset for PSEEN_END_DATE = 2021-06-30
- Creating a table for the new filtered dataset with the sum of PATIENTS_SEEN for PATIENT_TYPE = Adult & Child per region

In [25]:
```python
byPseenDateJanJun21 = datasetJanJun21[(datasetJanJun21.PSEEN_END_DATE == '2021-06-30')]
tableSumJanJun21 = pd.pivot_table(byPseenDateJanJun21, values = 'PATIENTS_SEEN',
                                  index=['REGION_NAME'], columns = ['PATIENT_TYPE'],
                                  aggfunc=np.sum).reset_index()
tableSumJanJun21
```

Out[25]:

PATIENT_TYPE	REGION_NAME	Adult	Child
0	East of England	4196312	912176
1	London	4828262	1160068
2	Midlands	6978104	1471996
3	North East and Yorkshire	6405386	1192068
4	North West	5003666	1014118
5	South East	5178758	1342626
6	South West	3700402	772740
7	Unallocated	45542	13152

Calculating number of unique practices for both Adult and Child per region

STEPS:

- Using the filtered dataset created above for PSEEN_END_DATE = 2021-06-30
- Grouping the filtered dataset by Region and Patient type and calculating number of unqiue practices using nunique() for both Adult and Child per region.
- Creating a table for the grouped dataset

In [26]:
```python
uniquePracticeByRegion = pd.DataFrame(byPseenDateJanJun21.groupby(by = ['REGION_NAME', 'PATIENT_TYPE'])['PRACTICE_NAME'].nunique()).reset_
tableUniquePractice = pd.pivot_table(uniquePracticeByRegion, values = 'PRACTICE_NAME', index=['REGION_NAME'],
                    columns = ['PATIENT_TYPE']).reset_index()
tableUniquePractice
```

Out[26]:

PATIENT_TYPE	REGION_NAME	Adult	Child
0	East of England	704	674
1	London	1110	1052
2	Midlands	1081	1033
3	North East and Yorkshire	861	840
4	North West	832	809
5	South East	1019	990
6	South West	582	560
7	Unallocated	26	23

In [27]:
```python
regionName = ['East of England', 'London', 'Midlands', 'North East and Yorkshire', 'North West', 'South East', 'South West', 'Unallocated'
#regionName list is  created for further use while calculating mean for Adults and Children
```

Calculating Mean Adults per practice per region

STEPS:

- Dividing the values for sum of PATIENT_SEEN (Adult) per region by number of unique practices (Adult) per region as calculated above to get mean adults seen per practice per region

- Creating a dataframe showing mean adults seen per practice per region

In [28]:
```python
adultMean = tableSumJanJun21['Adult']/tableUniquePractice['Adult']
adultMeanDataframe = pd.DataFrame({'REGION_NAME' : regionName, 'MEAN_ADULTS' : adultMean})
adultMeanDataframe
```

Out[28]:

	REGION_NAME	MEAN_ADULTS
0	East of England	5960.670455
1	London	4349.785586
2	Midlands	6455.230342
3	North East and Yorkshire	7439.472706
4	North West	6014.021635
5	South East	5082.196271
6	South West	6358.079038
7	Unallocated	1751.615385

Calculating Mean Children per practice per region

STEPS:

- Dividing the values for sum of PATIENT_SEEN (Child) per region by number of unique practices (Child) per region as calculated above to get mean adults seen per practice per region
- Creating a dataframe showing mean children seen per practice per region

In [29]:
```python
childrenMean = tableSumJanJun21['Child']/tableUniquePractice['Child']
childrenMeanDataframe = pd.DataFrame({'REGION_NAME' : regionName, 'MEAN_CHILDREN' : childrenMean})
childrenMeanDataframe
```

Out[29]:

	REGION_NAME	MEAN_CHILDREN
0	East of England	1353.376855
1	London	1102.726236
2	Midlands	1424.971926
3	North East and Yorkshire	1419.128571
4	North West	1253.545117
5	South East	1356.187879
6	South West	1379.892857
7	Unallocated	571.826087

Combining results

Steps:

- Creating a table by combining individual results for:
  - Number of unique practices per region
  - Mean Adults per practice per region
  - Mean Children per practice per region

In [30]:
```python
tableSumAndUnique = pd.merge(tableSumJanJun21, tableUniquePractice, on = 'REGION_NAME') #using pd.merge function of pandas
tableMean = pd.merge(tableSumAndUnique, adultMeanDataframe)
tableMeanFinal = pd.merge(tableMean, childrenMeanDataframe)
tableMeanFinal = tableMeanFinal.rename(columns = {'Adult_x': 'SUM_ADULTS_SEEN',
                                                  'Child_x': 'SUM_CHILDREN_SEEN',
                                                  'Adult_y': 'UNIQUE_PRACTICE_ADULTS',
                                                  'Child_y': 'UNIQUE_PRACTICE_CHILDREN'})
tableMeanFinal
```

Out[30]:

	REGION_NAME	SUM_ADULTS_SEEN	SUM_CHILDREN_SEEN	UNIQUE_PRACTICE_ADULTS	UNIQUE_PRACTICE_CHILDREN	MEAN_ADULTS	MEAN_CHILDREN
0	East of England	4196312	912176	704	674	5960.670455	1353.376855
1	London	4828262	1160068	1110	1052	4349.785586	1102.726236
2	Midlands	6978104	1471996	1081	1033	6455.230342	1424.971926
3	North East and Yorkshire	6405386	1192068	861	840	7439.472706	1419.128571
4	North West	5003666	1014118	832	809	6014.021635	1253.545117
5	South East	5178758	1342626	1019	990	5082.196271	1356.187879
6	South West	3700402	772740	582	560	6358.079038	1379.892857
7	Unallocated	45542	13152	26	23	1751.615385	571.826087

NOTE:

During the time period ending 30th June 2021, the overall number of adults seen per dental practice per region was more than the number of children seen per dental practice per region for all the regions

Create a function that takes a practice code and a reporting period end date as inputs and creates a pie chart that shows the distribution of patients seen by age group.

Answer 3:

Creating a new column 'AGE_CATEGORY' to make the interpretation of pie chart better with age categories and not individual ages

(Reference for categorisation of age - Center for Disease Control and Prevention)

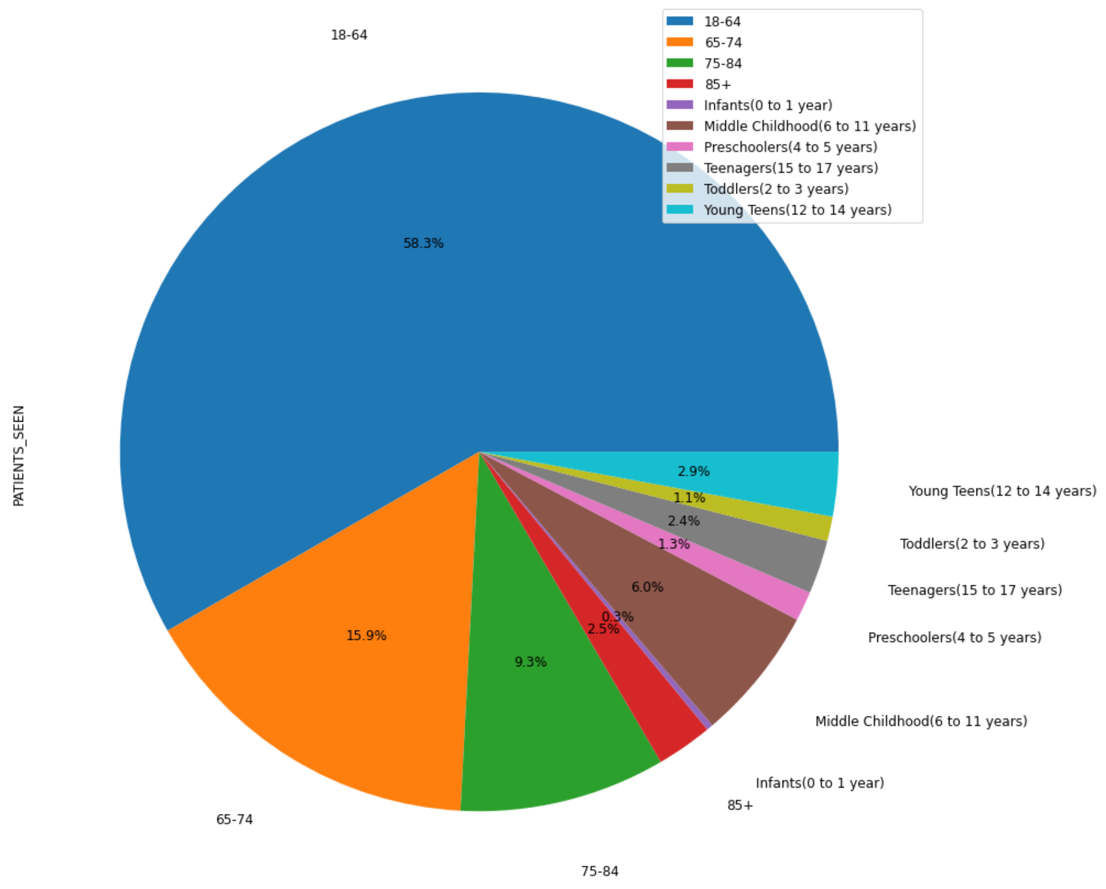
In [31]:
```python
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_BAND']
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['0', '1'], 'Infants(0 to 1 year)')
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['2', '3'], 'Toddlers(2 to 3 years)')
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['4', '5'], 'Preschoolers(4 to 5 years)')
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['6', '7', '8', '9', '10', '11'], 'Middle Childhood(6 to 11 year
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['12', '13', '14'], 'Young Teens(12 to 14 years)')
datasetJanJun21['AGE_CATEGORY'] = datasetJanJun21['AGE_CATEGORY'].replace(['15', '16', '17'], 'Teenagers(15 to 17 years)')
```

STEPS:

- Creating a function which takes reportingDate (PSEEN_END_DATE) and practiceCode (PRACTICE_CODE) as inputs
- Grouping dataset by AGE_CATEGORY and summing up number of PATIENTS_SEEN
- Plotting the pie chart

In [32]:
```python
def distributionPatients(reportingDate, practiceCode):
    datasetJanJun21Pie = datasetJanJun21[(datasetJanJun21['PSEEN_END_DATE'] == reportingDate) & (datasetJanJun21['PRACTICE_CODE'] == pract
    ageCatJanJun21 = datasetJanJun21Pie.groupby(['AGE_CATEGORY']).sum()
    ageCatJanJun21.plot.pie(x = 'AGE_CATEGORY', y = 'PATIENTS_SEEN', autopct = '%1.1f%%', labeldistance = 1.2, figsize=(15,15))
```

In [33]:
```python
distributionPatients('2021-06-30', 'V07661') #plotting pie chart for a specific reportingDate and practiceCode
```



For reporting period ending on 30th Jun 2021 and practice code, V07661 :

- More than half of patients seen (58.3 %) were from age category 18 - 64
- This was followed by patients in age category 65 - 74 (15.9 %) and 9.3 % in age category 75 - 84 (9.3 %)
- Rest 16.5 % was covered by patients from age categories less than 18 years including Infants, Toddlers, PreSchoolers, Middle Childhood, Young Teens and Teenagers

While most variables stay the same across the full reporting period, from January 2020 to end June 2021, there are some small differences in terms of the way the data is stored by the NHS.

Creating a loop that loads in all three data files and combine them in a dataframe, covering the full reporting period and taking into account any differences between the files.

Reporting the total numbers of

(A) observations and

(B) unique practice codes across all datasets.

Using a loop that loads in all three data files, covering the full reporting period into a single pandas dataframe.

In [34]:
```python
files = ["nhs-dent-stat-eng-jan-jun-20-anx3-ps-prac", "nhs-dent-stat-eng-jul-dec-20-anx3-ps-prac", "nhs-dent-stat-eng-jan-jun-21-anx3-ps-p
fullData = []
for i in range(len(files)):
    df = pd.read_csv(files[i] + '.csv')
    fullData.append(df)

datasetJan20Jun21 = pd.concat(fullData, axis = 0, ignore_index = True)
datasetJan20Jun21.head()
datasetJan20Jun21.shape
```

Out[34]:
```
(2799177, 18)
```

In [35]:
```python
datasetJan20Jun21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2799177 entries, 0 to 2799176
Data columns (total 18 columns):
 #   Column           Dtype  
---  ------           -----  
 0   PSEEN_END_DATE   object 
 1   GEOTYPE          object 
 2   PRACTICE_CODE    object 
 3   PRACTICE_NAME    object 
 4   PRAC_POSTCODE    object 
 5   CCG_CODE         object 
 6   CCG_ONS_CODE     object 
 7   CCG_NAME         object 
 8   LA_CODE          object 
 9   LA_NAME          object 
 10  REGION_CODE      object 
 11  REGION_ONS_CODE  object 
 12  REGION_NAME      object 
 13  PATIENT_TYPE     object 
 14  AGE_BAND         object 
 15  PATIENTS_SEEN    int64  
 16  POPULATION       float64
 17  GEOG_TYPE        object 
dtypes: float64(1), int64(1), object(16)
memory usage: 384.4+ MB
```

NOTE:

There are three problems with the full dataset 'datasetJan20Jun21' that we created above:-

- The date format for dataset from Jan20 to Jun20 is as '31/01/2020' and for Jul20 to Jun 2021 is as '2021-06-30'. Hence, we need to correct these differences to have same date format overall.
- As per the information for Dtype, 'PSEEN_END_DATE' column has datatype 'object'. This needs to be converted to datetime format using a pandas function (as shown below) so that a uniform datetime format is present for column 'PSEEN_END_DATE'. This will be also helpful when we do an analysis of the time-series data of the dataset in the next question.
- Column name for Geography type for dataset from Jan20 to Jun20 is 'GEOTYPE'. However, the name of same column for dataset from Jul20 to Jun21 is 'GEOG_TYPE'. This is the reason we see 2 columns for geography in our full dataset 'GEOTYPE' and 'GEOG_TYPE'. As expected, all the 'GEOTYPE' column values for dataset from Jul20 to Jun21 would be NaN and all the 'GEOG_TYPE' column values for dataset from Jan20 to Jun20 would be NaN.

Correcting date format for full dataset 'datasetJan20Jun21'

In [36]:
```python
datasetJan20Jun21['PSEEN_END_DATE'] = pd.to_datetime(datasetJan20Jun21['PSEEN_END_DATE'])
datasetJan20Jun21.info()
datasetJan20Jun21.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2799177 entries, 0 to 2799176
Data columns (total 18 columns):
 #   Column           Dtype         
---  ------           -----         
 0   PSEEN_END_DATE   datetime64[ns]
 1   GEOTYPE          object        
 2   PRACTICE_CODE    object        
 3   PRACTICE_NAME    object        
 4   PRAC_POSTCODE    object        
 5   CCG_CODE         object        
 6   CCG_ONS_CODE     object        
 7   CCG_NAME         object        
 8   LA_CODE          object        
 9   LA_NAME          object        
 10  REGION_CODE      object        
 11  REGION_ONS_CODE  object        
 12  REGION_NAME      object        
 13  PATIENT_TYPE     object        
 14  AGE_BAND         object        
 15  PATIENTS_SEEN    int64         
 16  POPULATION       float64       
 17  GEOG_TYPE        object        
dtypes: datetime64[ns](1), float64(1), int64(1), object(15)
memory usage: 384.4+ MB
```

Out[36]:

	PSEEN_END_DATE	GEOTYPE	PRACTICE_CODE	PRACTICE_NAME	PRAC_POSTCODE	CCG_CODE	CCG_ONS_CODE	CCG_NAME	LA_CODE	LA_NAME	REGION_CODE	REGION_ONS_COD
0	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
1	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
2	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
3	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
4	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na

Creating a new column 'GEOGRAPHY_TYPE' for full dataset 'datasetJan20Jun21' containing values from 'GEOTYPE' and 'GEOG_TYPE'

The function .fillna(' ') is used to take care of NaN values while merging the 2 columns 'GEOTYPE' and 'GEOG_TYPE' as also explained above

In [37]:
```python
datasetJan20Jun21['GEOGRAPHY_TYPE'] = datasetJan20Jun21['GEOTYPE'].fillna('').astype(str) + datasetJan20Jun21['GEOG_TYPE'].fillna('').asty
datasetJan20Jun21.head()

```
Out[37]:

	PSEEN_END_DATE	GEOTYPE	PRACTICE_CODE	PRACTICE_NAME	PRAC_POSTCODE	CCG_CODE	CCG_ONS_CODE	CCG_NAME	LA_CODE	LA_NAME	REGION_CODE	REGION_ONS_COD
0	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
1	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
2	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
3	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na
4	2020-01-31	CCG	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		Na

As we now have a column 'GEOGRAPHY_TYPE' containing geography type values for our full dataset, we may wish to drop columns 'GEOTYPE' and 'GEOG_TYPE' as they add no value to our full dataset 'datasetJan20Jun21'.

In [38]:
```python
datasetJan20Jun21Final = datasetJan20Jun21.drop(columns = ['GEOTYPE', 'GEOG_TYPE'])
datasetJan20Jun21Final
```
Out[38]:

	PSEEN_END_DATE	PRACTICE_CODE	PRACTICE_NAME	PRAC_POSTCODE	CCG_CODE	CCG_ONS_CODE	CCG_NAME	LA_CODE	LA_NAME	REGION_CODE	REGION_ONS_CODE
0	2020-01-31	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		NaN
1	2020-01-31	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		NaN
2	2020-01-31	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		NaN
3	2020-01-31	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		NaN
4	2020-01-31	NaN	NaN	NaN	Unallocated	Unallocated	Unallocated	NaN	NaN		NaN
...	...	...	...	...	...	...	...	...	...	...	...
2799172	2021-06-30	V98262	McCormick and Harrington Ltd	TS23 2LU	16C	E38000247	NHS Tees Valley CCG	E06000004	Stockton-on-Tees Borough Council	Y63	E40000009
2799173	2021-06-30	V98262	McCormick and Harrington Ltd	TS23 2LU	16C	E38000247	NHS Tees Valley CCG	E06000004	Stockton-on-Tees Borough Council	Y63	E40000009
2799174	2021-06-30	V98262	McCormick and Harrington Ltd	TS23 2LU	16C	E38000247	NHS Tees Valley CCG	E06000004	Stockton-on-Tees Borough Council	Y63	E40000009
2799175	2021-06-30	V98262	McCormick and Harrington Ltd	TS23 2LU	16C	E38000247	NHS Tees Valley CCG	E06000004	Stockton-on-Tees Borough Council	Y63	E40000009
2799176	2021-06-30	V98262	McCormick and Harrington Ltd	TS23 2LU	16C	E38000247	NHS Tees Valley CCG	E06000004	Stockton-on-Tees Borough Council	Y63	E40000009

2799177 rows × 17 columns

Part A

There are 2799177 observations (rows) and 17 variables (columns) for the full dataset from Jan 2020 to Jun 2021

In [39]:
```python
datasetJan20Jun21Final.shape
```
Out[39]:
```
(2799177, 17)
```

Part B

Total number of unique practice codes for the full dataset from Jan 2020 to Jun 2021 are 7271

In [40]:
```python
numberUniquePracticeCode = datasetJan20Jun21Final['PRACTICE_CODE'].dropna().unique()
print('Number of unique practice codes = ', len(numberUniquePracticeCode))
for i in numberUniquePracticeCode:
    print(i)
```
```
Number of unique practice codes =  7271
V00003
V00004
V00005
V00006
V00007
V00009
V00010
V00011
V00013
V00014
V00015
V00016
V00018
V00019
V00021
V00023
V00024
V00025
```

An analysis of the time-series of the total number of adult patients seen across the full period in London. Visualising and discussing how the COVID-19 variables from PHE relate to the number of patients seen at the London dentistry clinics.

STEPS:

- Filtering our full dataset for PATIENT_TYPE = 'Adult' & REGION_NAME = 'London'
- Grouping the new filtered dataset by PSEEN_END_DATE and summing the number of PATIENTS_SEEN by this date
- Plotting the line graph

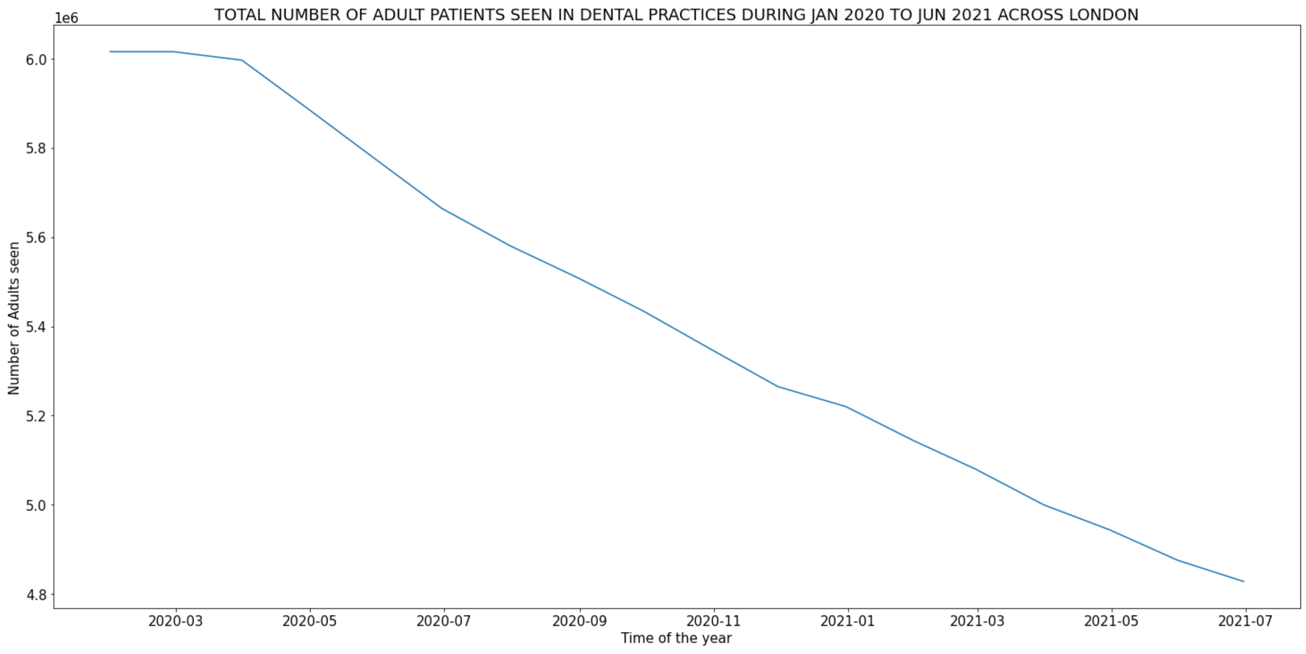
In [41]:
```
datasetJan20Jun21Final = datasetJan20Jun21Final[(datasetJan20Jun21Final['PATIENT_TYPE'] == 'Adult') & (datasetJan20Jun21Final['REGION_NAME

```

In [42]:
```
adultLondonJan20Jun21 = datasetJan20Jun21Final.groupby(['PSEEN_END_DATE'], dropna = False).sum().reset_index()
adultLondonJan20Jun21 = adultLondonJan20Jun21[['PSEEN_END_DATE', 'PATIENTS_SEEN']]
print(adultLondonJan20Jun21)

plt.rcParams['figure.figsize'] = (25, 12)
plt.rcParams.update({'font.size': 15})
plt.plot(adultLondonJan20Jun21['PSEEN_END_DATE'], adultLondonJan20Jun21['PATIENTS_SEEN'])
plt.title('TOTAL NUMBER OF ADULT PATIENTS SEEN IN DENTAL PRACTICES DURING JAN 2020 TO JUN 2021 ACROSS LONDON')
plt.xlabel('Time of the year')
plt.ylabel('Number of Adults seen')
plt.show()
```

```
   PSEEN_END_DATE  PATIENTS_SEEN
0      2020-01-31        6015404
1      2020-02-29        6015214
2      2020-03-31        5996360
3      2020-04-30        5887684
4      2020-05-31        5773834
5      2020-06-30        5664248
6      2020-07-31        5580672
7      2020-08-31        5508480
8      2020-09-30        5433542
9      2020-10-31        5347204
10     2020-11-30        5264916
11     2020-12-31        5220000
12     2021-01-31        5143270
13     2021-02-28        5080010
14     2021-03-31        5000034
15     2021-04-30        4944036
16     2021-05-31        4875746
17     2021-06-30        4828262
```



We can add another column to see difference in PATIENTS_SEEN across months between Jan 2020 to Jun 2021

- This is calculated by substracting the values for consecutive rows in 'PATIENTS_SEEN'
- Negative sign indicates the reduction in patient numbers from previous month

In [43]:
```
adultLondonJan20Jun21['DIFFERENCE_PATIENTS_SEEN'] = adultLondonJan20Jun21['PATIENTS_SEEN'].diff(1)
adultLondonJan20Jun21
```

Out[43]:

	PSEEN_END_DATE	PATIENTS_SEEN	DIFFERENCE_PATIENTS_SEEN
0	2020-01-31	6015404	NaN
1	2020-02-29	6015214	-190.0
2	2020-03-31	5996360	-18854.0
3	2020-04-30	5887684	-108676.0
4	2020-05-31	5773834	-113850.0
5	2020-06-30	5664248	-109586.0
6	2020-07-31	5580672	-83576.0
7	2020-08-31	5508480	-72192.0
8	2020-09-30	5433542	-74938.0
9	2020-10-31	5347204	-86338.0
10	2020-11-30	5264916	-82288.0
11	2020-12-31	5220000	-44916.0
12	2021-01-31	5143270	-76730.0
13	2021-02-28	5080010	-63260.0
14	2021-03-31	5000034	-79976.0
15	2021-04-30	4944036	-55998.0
16	2021-05-31	4875746	-68290.0
17	2021-06-30	4828262	-47484.0

Discussion

Looking at the table and line chart for total number of adult patients seen in dental practices in London during Jan 2020 to Jun 2021, there is clear decreasing trend in the number of patients visiting dental practices. The overall decrease for the given time period is approx 20% with number of adult patients seen dropping from 6 million (approx) in Jan 2020 to 4.8 million (approx) in Jun 2021.

The number of adult patients seen at the London dentistry clinics can be correlated with COVID-19 variables from Public Health England data

As number of new covid cases started going up mainly after Mar 2020, the number of patients seen reduced drastically with an approx drop of 100,000 patients in months of Apr, May and Jun 2020. This might be expected as going out for dental check ups when the covid-19 pandemic was growing would mean more exposure to infection. Moreover, the lockdown was prevalent in most of the England including London during that time.

The fall in number was less after Jul 2020 because number of new covid cases also leveled off and and remained stable until Oct 2020.

The next considerable increase in decline was from 44000 (approx) in Nov 2020 to 76000 (approx) in Dec 2021. This could be explained by the rising of covid cases at that time which made the dental appointments unlikely again. The number remained almost similar for next 5 months (a mix of upward and downward trend) as the new covid cases were also decreasing gradually.

Lastly, the drop was smaller for the month of Jun 2021 as compared to previous 6 months. This could be possibly because number of people getting second dose vaccinations were quite high between Apr 2021 to Jun 2021. Therefore, most people might be doubly vaccinated and were now considering their visits to dental practices. This could also be justified if the drop in the number of patients seen at dental practices had also plummeted down for rest of the year after Jun 2021