Introduction

This Python project aims to explore the Vaccinations Dataset in the UK from early 2021 to mid 2022. The correlation and relationship between the selected variables in the dataset will be presented and visualised using different analysis tools. The data would also be grouped into certain categories for a more precise and readable interpretation of the vaccination progress in the said period. One of the possible implications to most datasets is reliability and accuracy as some vaccinations given could have not been recorded in the dataset, thus, there could be a trivial fluctuation compared to the actual number due to odds and errors, for example, missing values from 'FirstDose', 'SecondDose', and 'ThirdDose' will be replaced by the mean of that column, therefore, the analysis result may not be entirely accurate. By further exploring this dataset, some valuable information could be extracted and reviewed for future enhancement and development of the vaccination progress.

Importing necessary libraries for data analysis and visualisation

In [1]:
```
import pandas as pd
import os
import numpy as np
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

In [2]:
```
from bokeh.io import output_notebook
output_notebook()

from bokeh.plotting import figure
from bokeh.io import show
```

BokehJS 2.3.2 successfully loaded.

In [ ]:
```
#Importing the dataset
```

In [3]:
```
df = pd.read_excel('UK_VaccinationsData.xlsx')
df.head(5)
```

Out[3]:

	areaName	areaCode	year	month	Quarter	day	WorkingDay	FirstDose	SecondDose	ThirdDose
0	England	E92000001	2022.0	5	Q2	Mon	Yes	3034.0	3857.0	8747.0
1	England	E92000001	2022.0	5	Q2	Sun	No	5331.0	3330.0	4767.0
2	England	E92000001	2022.0	5	Q2	Sat	No	13852.0	9759.0	12335.0
3	England	E92000001	2022.0	5	Q2	Fri	Yes	5818.0	5529.0	10692.0
4	England	E92000001	2022.0	5	Q2	Thu	Yes	8439.0	6968.0	11701.0

1. Generate descriptive statistics of the dataset variables, and comment on main trends.

In [7]:
```
df.describe()
```

Out[7]:

	year	month	FirstDose	SecondDose	ThirdDose
count	903.000000	904.000000	900.000000	901.000000	898.000000
mean	2021.625692	5.946903	4994.323333	5574.125416	42529.570156
std	0.484212	4.146467	9651.335670	9174.101390	104877.579915
min	2021.000000	1.000000	0.000000	0.000000	0.000000
25%	2021.000000	2.000000	338.500000	478.000000	1313.500000
50%	2022.000000	4.000000	876.500000	971.000000	6992.000000
75%	2022.000000	11.000000	3653.250000	5770.000000	23464.750000
max	2022.000000	12.000000	115551.000000	48491.000000	830403.000000

Comments on main trends:

- The dataset contains records from the year 2021 to 2022, with a mean year of 2021.625692.
- On average, there were more second doses administered than first doses, with a mean of 5574.125 second doses compared to 4994.323 first doses per day.
- There were more third doses administered than either first or second doses, with a mean of 42529.57 third doses per day.
- The standard deviation of the number of doses administered is high, indicating a wide variability in the number of doses administered from day to day.
- The minimum number of doses administered is zero, indicating there were some days where no vaccines were administered.
- Number of data being recorded is approximately 900
- There is a wide range of doses administered, with the maximum number of first doses administered in a single day being 115551 and the maximum number of third doses administered being 830403.

2. Check any records with missing values, and handle the missing data as appropriate.

In [8]:
```
df.isnull().any()
```

Out[8]:
```
areaName      False
areaCode      False
year           True
month         False
Quarter        True
day            True
WorkingDay     True
FirstDose      True
SecondDose     True
ThirdDose      True
dtype: bool
```

In [ ]:
```
# There are missing values in every column in the dataset except for 3 columns, which are 'areaName', 'areaCode', 'month'
```

In [24]:
```
df.isnull().sum()
```

```
Out[24]: areaName      0
areaCode      0
year          1
month         0
Quarter       1
day           1
WorkingDay    2
FirstDose     4
SecondDose    3
ThirdDose     6
dtype: int64
```

In [ ]:
```python
# This show the number of missing values in each column
```

In [93]:
```python
# Handle missing year
df['year'].fillna(2022, inplace=True)
```

In [94]:
```python
# Handle missing quarter
df['Quarter'].fillna('Q4', inplace = True)
```

In [95]:
```python
# Handle missing day
df['day'].fillna('Fri', inplace = True)
```

In [96]:
```python
# Handle missing WorkingDay
df['WorkingDay'].fillna('Yes', inplace = True)
```

Using the mean value to substitute the missing values of 'FirstDose', 'SecondDose', 'ThirdDose'

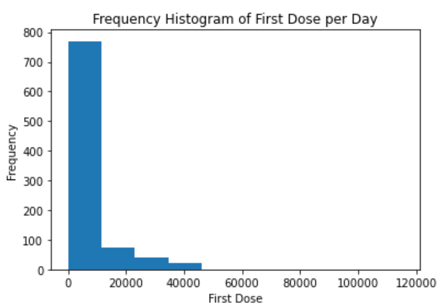
In [97]:
```python
# Handle missing FirstDose
df['FirstDose'].fillna(df['FirstDose'].median(), inplace=True)
```

In [19]:
```python
# Handle missing SecondDose
df['SecondDose'].fillna(df['SecondDose'].median(), inplace=True)
```

In [20]:
```python
# Handle missing ThirdDose
df['ThirdDose'].fillna(df['ThirdDose'].median(), inplace=True)
```

3. Plot the distribution of one or more individual continuous variables and provide comments

In [98]:
```python
plt.hist(df['FirstDose'], bins=10)
plt.xlabel('First Dose')
plt.ylabel('Frequency')
plt.gca().set(title='Frequency Histogram of First Dose per Day')
plt.show()
```

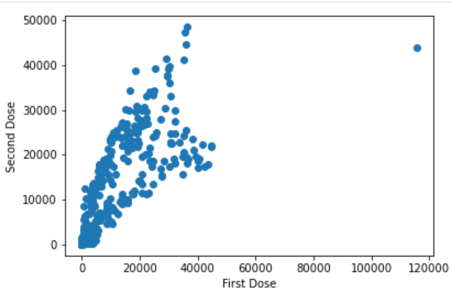


- From the histogram, we can see that the distribution of FirstDose is right-skewed, with a long tail on the right-hand side.
- Additionally, we can see that the majority of the observations are in the lower range of FirstDose, which is less than 10000 per day.
- However, there are a few observations on the higher end, which is around 40000 or more

4. Build graphs visualizing the association b/w two numeric variables and interpret them.

In [ ]:
```python
# 2 chosen numeric values for analysis and visualisation are 'FirstDose' and 'SecondDose'
```

In [99]:
```python
plt.scatter(df['FirstDose'], df['SecondDose'])
plt.xlabel('First Dose')
plt.ylabel('Second Dose')
plt.show()
```



There seems to be a relationship between X and Y, and it appears it can be fit by a line.

We can also check the correlation between them:

In [100…
```
df['FirstDose'].corr(df['SecondDose'])
```

Out[100…
```
0.8349717023208916
```

Correlation between two variables is close to 1, hence, there is a positive correlation between these FirstDose and SecondDose

The model is defined by a formula, where first comes the Y variable, followed by the tilda sign (~), followed by the X variable.

In [101…
```
model = sm.OLS.from_formula('SecondDose ~ FirstDose', data=df).fit()
```

We can plot the fitted line. To do that, we first obtain the intercept and the slope - they are available in the `params` attribute of the fitted model.

In [102…
```
intercept, slope = model.params
print(intercept)
print(slope)
```

```
1612.1382814923518
0.7940417366785499
```

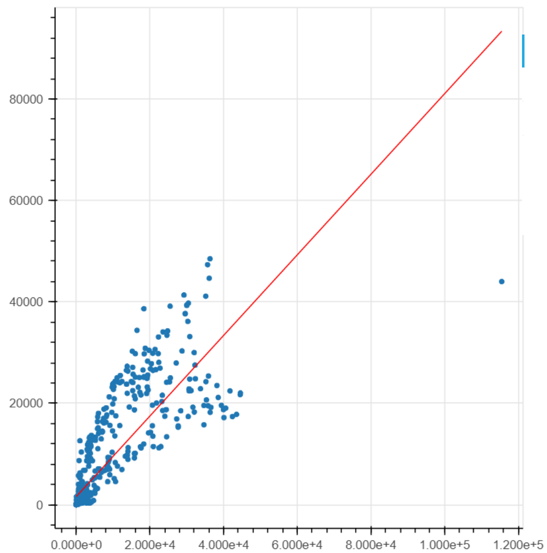
We can see that the slope of the regression line is positive, which confirms the positive linear relationship between FirstDose and SecondDose

Next, we can obtain predicted values for Y, given the X values, using the intercept and the slope:

In [103…
```
y_pred = [slope*i + intercept  for i in df['FirstDose']]
```

We can now plot the fitted line:

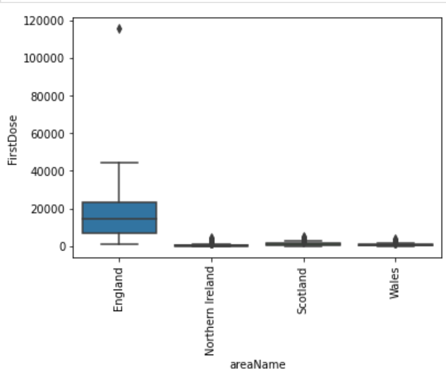
In [104…
```
fig = figure(height=500, width=500)
fig.circle(df['FirstDose'], df['SecondDose'])
fig.line(df['FirstDose'], y_pred, color='red')
show(fig)
```



We can see that the majority of the observations fall close to the regression line, suggesting a strong association between the variables.

5. Visualise the relationship b/w a categorical and a numeric variables, provide comments.

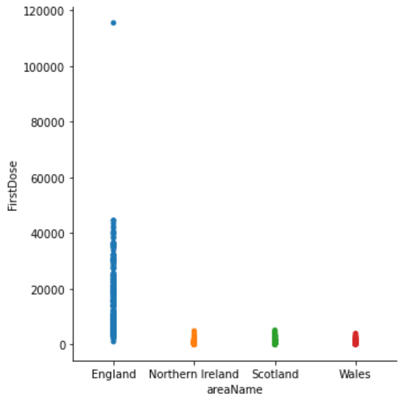
In [105…
```
# First visualization
sns.boxplot(x='areaName', y='FirstDose', data = df)
plt.xticks(rotation=90)
plt.show()
```



In [106…
```
# Second Visualization
sns.catplot(data=df, x="areaName", y="FirstDose", jitter=False)
```

Out[106…

```
<seaborn.axisgrid.FacetGrid at 0x1f1abd99a00>
```



Comments: From the two plots, we can see that the first doses given are highest in England, followed by Scotland, Wales and Northern Ireland

6. Build a contingency table of two potentially related categorical variables, then conduct a statistical test of the independence between them and interpret the results.

In [ ]:

```
# Contigency Table of 'WorkingDay' and 'Quarter'
```

In [107…

```
data_crosstab = pd.crosstab(df['WorkingDay'], df['Quarter'])
print(data_crosstab)
```

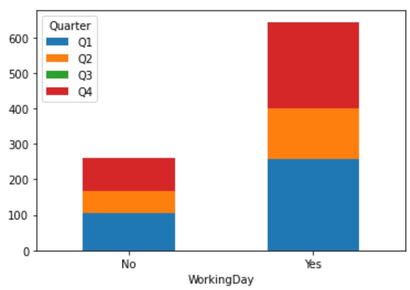
```
Quarter      Q1   Q2  Q3   Q4
WorkingDay
No          104   62   0   94
Yes         256  144   2  242
```

In [108…

```
data_crosstab.plot(kind="bar", stacked=True, rot=0)
```

Out[108…

```
<AxesSubplot:xlabel='WorkingDay'>
```



We use Chi-square test

The null hypothesis of the chi-square test is always that the two variables are independent, the alternative hypothesis is that they are dependent.

In [109…

```
from scipy import stats
chi2, p_val, dof, expected = stats.chi2_contingency(data_crosstab)
print(f"p-value: {p_val}")
```

```
p-value: 0.7792933611960982
```

the p-value is 0.78, which indicates that we fail to reject the null hypothesis of independence at a significance level of 0.05. This means that there is not enough evidence to suggest that the WorkingDay and Quarter variables are related.

7. Retrieve one or more subset of rows based on two or more criteria and present descriptive statistics on the subset(s).

In [110…

```
df.head(3)
```

Out[110…

	areaName	areaCode	year	month	Quarter	day	WorkingDay	FirstDose	SecondDose	ThirdDose
0	England	E92000001	2022.0	5	Q2	Mon	Yes	3034.0	3857.0	8747.0
1	England	E92000001	2022.0	5	Q2	Sun	No	5331.0	3330.0	4767.0
2	England	E92000001	2022.0	5	Q2	Sat	No	13852.0	9759.0	12335.0

In [111…

```
scotland_subset =df.loc[df['areaName']== 'Scotland'].loc[df['Quarter']=='Q1']
scotland_subset
scotland_subset.describe()
```

Out[111…

	year	month	FirstDose	SecondDose	ThirdDose
count	90.0	90.000000	90.000000	90.000000	90.000000
mean	2022.0	2.000000	939.244444	1833.633333	5508.788889
std	0.0	0.834643	881.343756	2008.968506	5044.416763
min	2022.0	1.000000	0.000000	0.000000	0.000000
25%	2022.0	1.000000	453.000000	735.750000	2173.250000
50%	2022.0	2.000000	776.000000	1028.000000	4233.500000

	year	month	FirstDose	SecondDose	ThirdDose
75%	2022.0	3.000000	1079.750000	2477.250000	6868.500000
max	2022.0	3.000000	4666.000000	12612.000000	25763.000000

There are 90 observations in this subset

On average, there were more third doses administered per day than second and first doses

8. Conduct a statistical test of the significance of the difference between the means of two subsets of the data and interpret the results.

Create another subset for England in the First Quarter

In [112…

```
england_subset=df.loc[df['areaName']== 'England'].loc[df['Quarter']=='Q1']
england_subset
england_subset.describe()
```

Out[112…

	year	month	FirstDose	SecondDose	ThirdDose
count	90.0	90.000000	90.000000	90.000000	90.000000
mean	2022.0	2.000000	9716.422222	19072.755556	41625.133333
std	0.0	0.834643	6081.817286	7811.600452	45317.807403
min	2022.0	1.000000	2203.000000	2684.000000	6366.000000
25%	2022.0	1.000000	4092.000000	12755.500000	16093.250000
50%	2022.0	2.000000	8118.500000	17851.000000	23614.000000
75%	2022.0	3.000000	13946.500000	24319.500000	45082.250000
max	2022.0	3.000000	29231.000000	41351.000000	206676.000000

We use independent two samples T-test

The null hypothesis is always that there is no difference between the means of the two populations that the samples represent. The alternative hypothesis is that there is a significant (not accidental) difference between them.

$H_0 : \mu = 0$

$H_A : \mu \neq 0$

In [113…

```
scotland = scotland_subset['FirstDose']
scotland.mean()
```

Out[113… 939.2444444444444

In [114…

```
england = england_subset['FirstDose']
england.mean()
```

Out[114… 9716.422222222222

In [115…

```
t_val, p_val = stats.ttest_ind(scotland, england)

print(f"t-value: {t_val}, p-value: {p_val}")
```

```
t-value: -13.549705694801181, p-value: 3.3622361029491423e-29
```

The p-value is smaller than the significance level ($\alpha = 0.05$), i.e., the difference between the two means falls inside the rejection area.

Therefore we reject the null hypothesis that the mean number of first dose given in Q1 in scotland is not different from the mean number of first dose given in Q1 in england.

9. Create one or more tables that group the data by a certain categorical variable and display summarized information for each group (e.g. the mean or sum within the group).

In [116…

```
df_grouped = df.groupby('areaName')
df_grouped.mean()
```

Out[116…

	year	month	FirstDose	SecondDose	ThirdDose
areaName					
England	2021.605932	6.084746	16869.427966	18469.016949	136510.710638
Northern Ireland	2021.605932	6.084746	496.453390	576.340426	4803.544681
Scotland	2021.639640	5.864865	1170.425676	1569.130631	14798.669683
Wales	2021.657143	5.723810	667.688095	864.480769	8271.487923

In [117…

```
df_grouped.sum()
```

Out[117…

	year	month	FirstDose	SecondDose	ThirdDose
areaName					
England	477099.0	1436	3981185.0	4358688.0	32080017.0
Northern Ireland	477099.0	1436	117163.0	135440.0	1128833.0
Scotland	448804.0	1302	259834.5	348347.0	3270506.0
Wales	424548.0	1202	140214.5	179812.0	1712198.0

In [118…

```
df_grouped.describe()
```

Out[118…

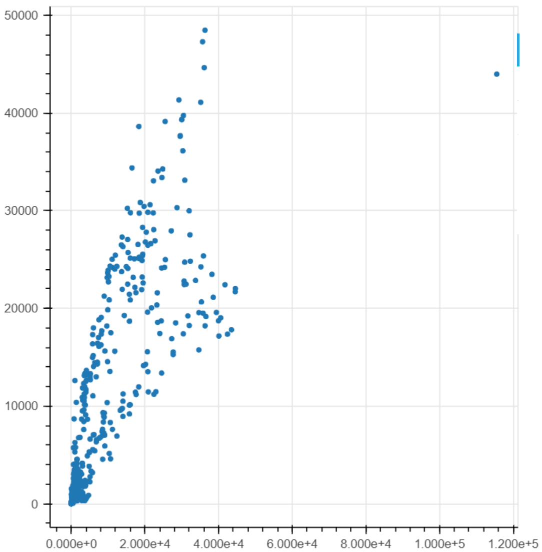
								year		month	...		SecondDose								ThirdDose
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
areaName																					
England	236.0	2021.605932	0.489688	2021.0	2021.0	2022.0	2022.0	2022.0	236.0	6.084746	...	24306.50	48491.0	235.0	136510.710638	172285.886197	2287.0	14665.0	33560.0	223502.5	830403.0
Northern Ireland	236.0	2021.605932	0.489688	2021.0	2021.0	2022.0	2022.0	2022.0	236.0	6.084746	...	740.00	8677.0	235.0	4803.544681	5893.712522	0.0	859.0	2057.0	6535.5	30803.0
Scotland	222.0	2021.639640	0.481190	2021.0	2021.0	2022.0	2022.0	2022.0	222.0	5.864865	...	2093.75	12612.0	221.0	14798.669683	17497.931038	0.0	1144.0	5387.0	28989.0	78146.0
Wales	210.0	2021.657143	0.475798	2021.0	2021.0	2022.0	2022.0	2022.0	210.0	5.723810	...	978.00	6247.0	207.0	8271.487923	10386.601598	18.0	940.5	2445.0	14742.5	50524.0

4 rows × 40 columns

10. Implement a linear regression model and interpret its outputs.

In [119…

```
fig = figure(height=500, width=500)
fig.circle(df['FirstDose'], df['SecondDose'])
show(fig)
```



In [120…

```
df['FirstDose'].corr(df['SecondDose'])
```

Out[120…

```
0.8349717023208916
```

In [121…

```
model = sm.OLS.from_formula('SecondDose ~ FirstDose', data=df).fit()
```

In [122…

```
intercept, slope = model.params
print(intercept)
print(slope)
```

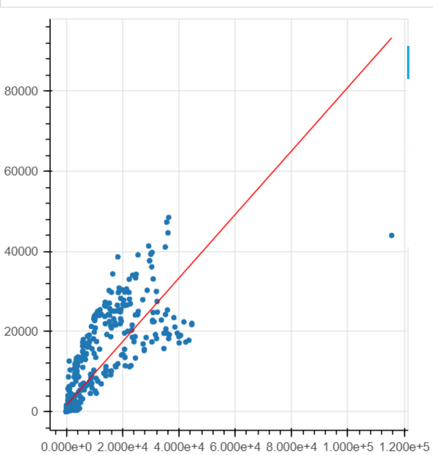
```
1612.1382814923518
0.7940417366785499
```

In [123…

```
y_pred = [slope*i + intercept  for i in df['FirstDose']]
```

In [124…

```
fig = figure(height=400, width=400)
fig.circle(df['FirstDose'], df['SecondDose'])
fig.line(df['FirstDose'], y_pred, color='red')
show(fig)
```



In [89]:

```
model.summary()
```

Out[89]:

OLS Regression Results

Dep. Variable:	SecondDose	**R-squared:**	0.697
Model:	OLS	**Adj. R-squared:**	0.697
Method:	Least Squares	**F-statistic:**	2078.
Date:	Sun, 26 Mar 2023	**Prob (F-statistic):**	2.52e-236
Time:	12:07:32	**Log-Likelihood:**	-8989.1
No. Observations:	904	**AIC:**	1.798e+04
Df Residuals:	902	**BIC:**	1.799e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1606.6292	188.820	8.509	0.000	1236.052	1977.207
FirstDose	0.7942	0.017	45.590	0.000	0.760	0.828

Omnibus:	193.649	**Durbin-Watson:**	0.317
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6085.156
Skew:	0.069	**Prob(JB):**	0.00
Kurtosis:	15.710	**Cond. No.**	1.22e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.22e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

R-squared is close to 0.7, hence it is a good model

The elements of the summary that are of main interest for us at the moment are:

(1) **Coefficients on the variables**. These are shown in the second table of the summary. The "coef" column shows the actual coefficients: 1616.6292 for the intercept, and 0.7942 for the FirstDose variable. Thus, our model is described by the line: $SecondDose = 1606.6292 + 0.7942 * FirstDose + e$.

(2) **Significance of the variables**. The summary includes results of a t-test assessing if the estimated coeffients are significantly different from 0. In this case both coefficients have a p-value of 0.000, which indicates a highly significant relationship between the two variables. The coefficient on FirstDose is significant (p=0.760, i.e., below $\alpha = 0.05$), and thus this factor does have an effect on the dependent variable.

(3) **Quality of the model**. The R^2 and the adjusted R^2 values are shown in the first table. Both are around 0.7, which indicates that the model is good. Obviously, there are other factors that affect the SecondDose that our model did not take into account.

Conclusion

By utilising different exploratory data analysis tools, it is notable that there is a strong correlation between the number of first doses given and second doses given, moreover, as it is a relatively recent dataset, the number of third doses given is much higher in this period, which indicates that the vaccination progress in the UK is conducted in a speedy manner, which is effective in preventing and minimising the impacts of COVID-19 in the upcoming period.