Introduction

One of the most efficient ways to stop the spread of the COVID-19 pandemic as it continues to have an influence on human lives is through vaccination. From early 2021, the government of the United Kingdom has begun distributing vaccination programmes to its residents with the intention of immunising as many individuals as possible to decrease the effects of the virus. By gathering statistics on the numbers of individuals who have received the first, second, and third doses of the COVID-19 immunisation in order to better understand the development of the vaccination programme in various areas of the UK. The information, which ranges from early 2021 to mid-2022, is the data provided for statistical analysis. The resulting analysis provides valuable insights to interested parties.

In [51]:
```
# Import the pandas, numpy & stats library
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

In [52]:
```
# Load the UK vaccination dataset into a pandas dataframe
df = pd.read_excel('UK_VaccinationsData(1).xlsx')
# Print to check
df.head()
```

Out[52]:

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

1 Generating Descriptive Dataset

In [53]:
```
#Generate descriptive statistics
df.describe()
```

Out[53]:

	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

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

In [54]:

```
#Check for missing values.
df.isnull().sum()
```

Out[54]:

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

In [55]:

```
#filling any missing values with the value of 0  instead of creating a new one
df.fillna(0, inplace=True)
print(df)
```

```
    areaName   areaCode    year  month Quarter  day WorkingDay  FirstDose  \
0    England  E92000001  2022.0      5      Q2  Mon        Yes     3034.0
1    England  E92000001  2022.0      5      Q2  Sun         No     5331.0
2    England  E92000001  2022.0      5      Q2  Sat         No    13852.0
3    England  E92000001  2022.0      5      Q2  Fri        Yes     5818.0
4    England  E92000001  2022.0      5      Q2  Thu        Yes     8439.0
..       ...        ...     ...    ...     ...  ...        ...        ...
899    Wales  W92000004  2021.0     10      Q4  Mon        Yes     3266.0
900    Wales  W92000004  2021.0     10      Q4  Sun         No     2831.0
901    Wales  W92000004  2021.0     10      Q4  Sat         No     3921.0
902    Wales  W92000004  2021.0     10      Q4  Fri        Yes     1238.0
903    Wales  W92000004  2021.0     10      Q4  Thu        Yes     1142.0

     SecondDose  ThirdDose
0        3857.0     8747.0
1        3330.0     4767.0
2        9759.0    12335.0
3        5529.0    10692.0
4        6968.0    11701.0
..          ...        ...
899       528.0    22390.0
900       322.0     6546.0
901       439.0    10787.0
902       717.0    18583.0
903       696.0        0.0

[904 rows x 10 columns]
```

Any missing values in the dataset are filled with the value of 0 using the fillna() function.

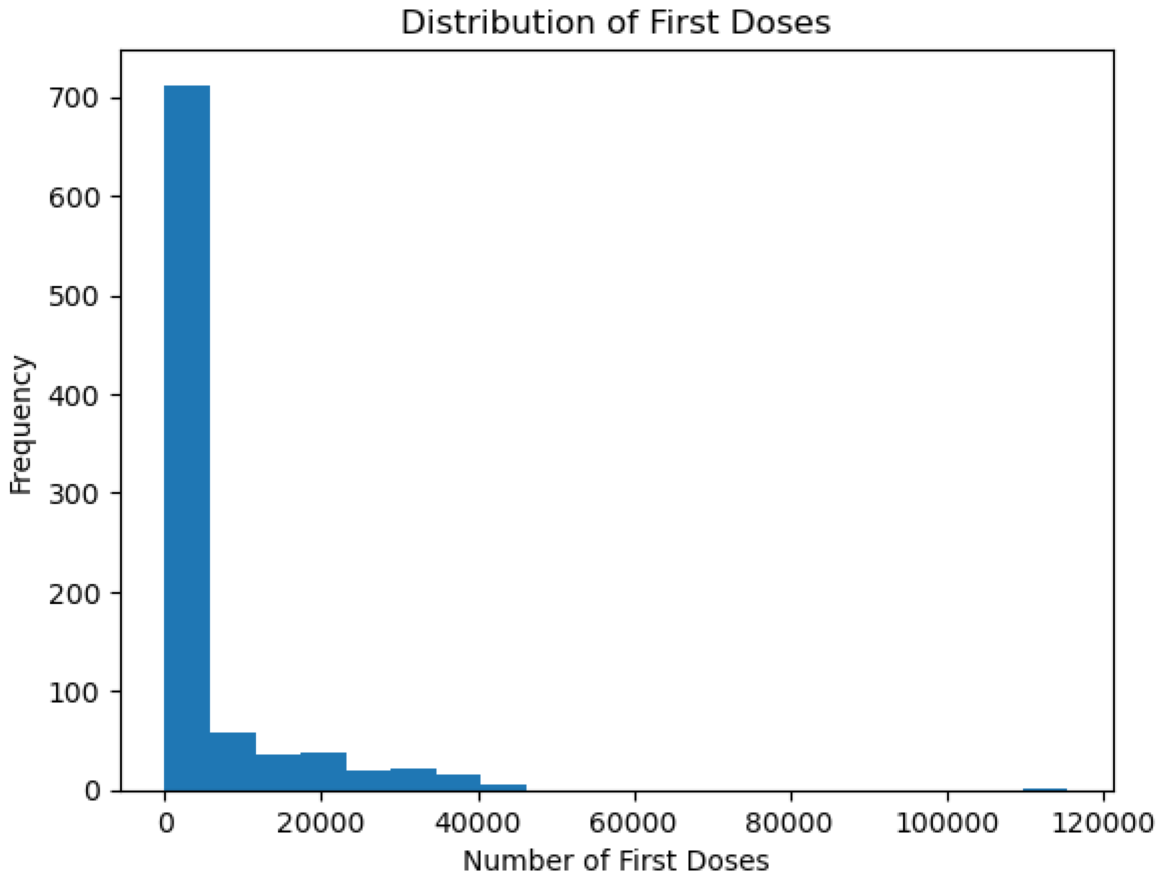
In [56]:
```
#Repeat check for missing values.
df.isnull().sum()
```

Out[56]:
```
areaName      0
areaCode      0
year          0
month         0
Quarter       0
day           0
WorkingDay    0
FirstDose     0
SecondDose    0
ThirdDose     0
dtype: int64
```

3. Build graphs visualizing the following and comment on the results

QA we are taking the first dose and second dose because the distribution of one or more individual continuous variables is telling for one or more.

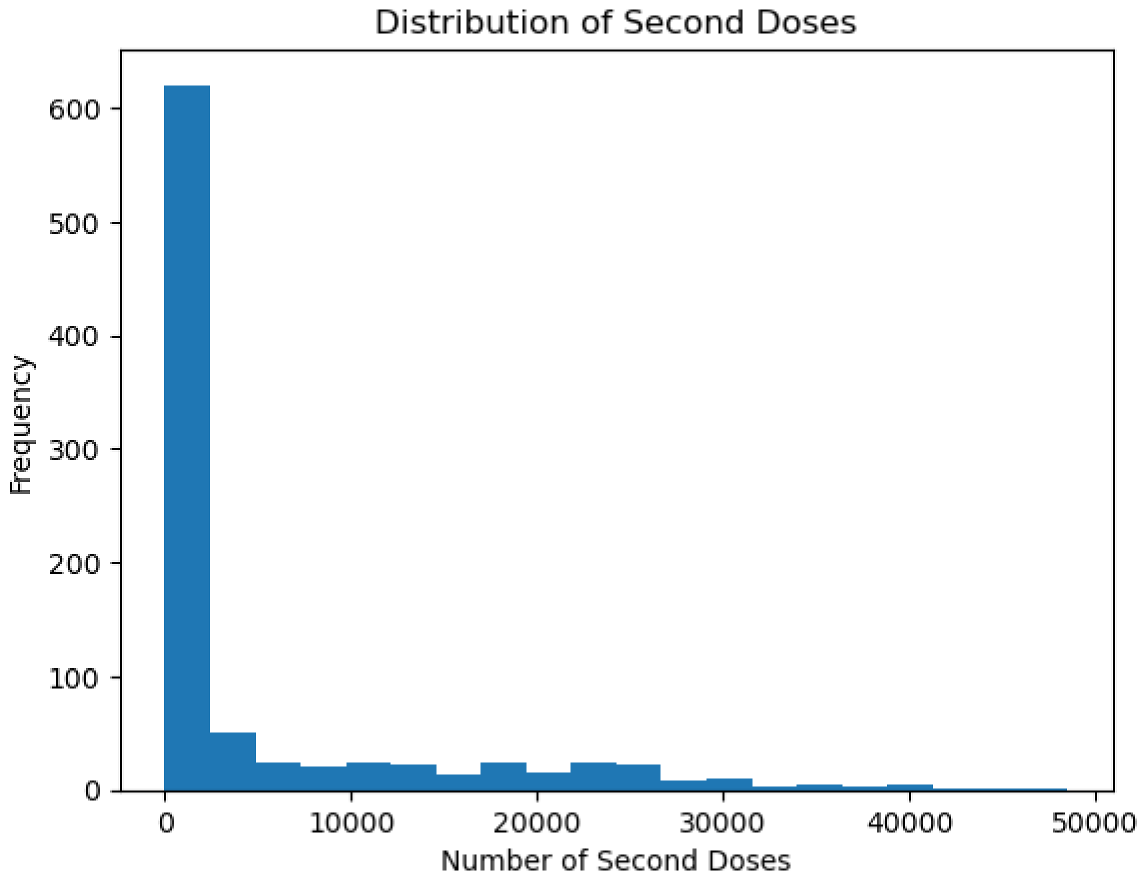
In [58]:
```
#3. Build graphs visualizing the following and comment on the results:
import matplotlib.pyplot as plt
# Plot the histogram
plt.hist(df['FirstDose'], bins=20)
plt.xlabel('Number of First Doses')
plt.ylabel('Frequency')
plt.title('Distribution of First Doses')
plt.show()
```



It displays the frequency of each bin.The majority of the population has received between 0 and 5000 first doses, with a peak at around 2500 first dose.The distribution of first doses appears to be positively skewed.

In [30]:

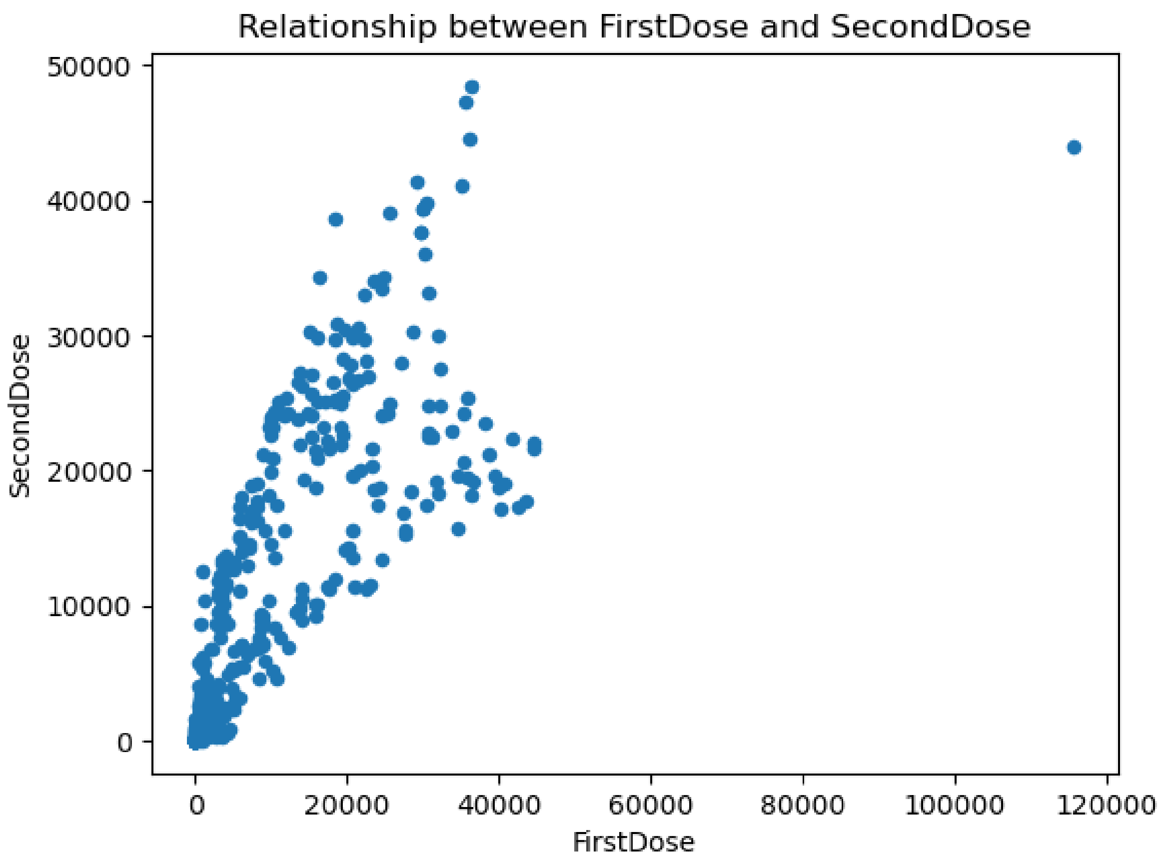
```
#Visualizing for the second dose
import matplotlib.pyplot as plt
# Plot the histogram
plt.hist(df['SecondDose'], bins=20)
plt.xlabel('Number of Second Doses')
plt.ylabel('Frequency')
plt.title('Distribution of Second Doses')
plt.show()
```



The distribution of second doses appears to be positively skewed. However, the majority of the population has received between 0 and 2000 first doses, with a peak at around 1000 first doses.

B the relationship of a pair of continuous variables.

In [31]:
```
#plotting the scatter plot graph
df.plot(kind='scatter', x='FirstDose', y='SecondDose')
plt.xlabel('FirstDose')
plt.ylabel('SecondDose')
plt.title('Relationship between FirstDose and SecondDose')
plt.show()
```

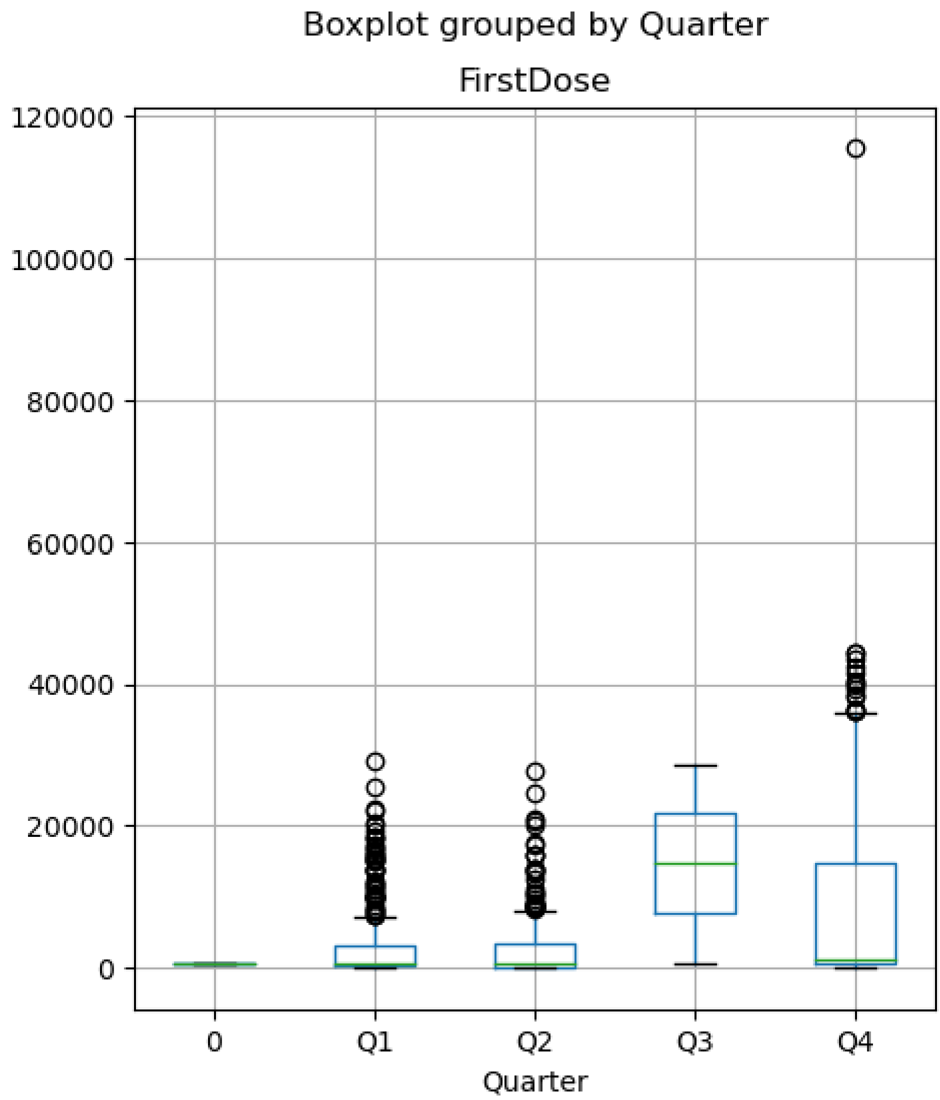


The scatter plot demonstrates that the number of First Doses and Second Doses administered are positively correlated.

C The relationship between a categorical variable and a continuous one

In [32]:
```
#plotting a whisker plot graph for the First dose vs Quater
df.boxplot(column = 'FirstDose', by='Quarter', figsize = [5, 6])
```

Out[32]:
```
<Axes: title={'center': 'FirstDose'}, xlabel='Quarter'>
```



The boxplot displays FirstDose distribution per quarter. Q1 and Q2 had median ≤1000 and IQR of 2500-4500 with anomalies. Q3 had median 15000 and IQR 10000-21000 with no anomalies. Q4 had median ≈1000 and IQR 2000-4000 with some outliers.

4 Display unique values of a categorical variable and their frequencies

In [33]:
```
#Counting the number of times each unique value appears in a column of a dataframe
print(df['areaName'].value_counts())
```

```
England             236
Northern Ireland    236
Scotland            222
Wales               210
Name: areaName, dtype: int64
```

The numbers 0,1,2,3 refers to the data of England, North Ireland, Scotland and Wales respectively.

5Build a contingency table of two potentially related categorical variables.

C

In [34]:
```
# Build a contingency table
cont_table = pd.crosstab(df['areaName'], df['day'])
#printing the contingency table
cont_table
```

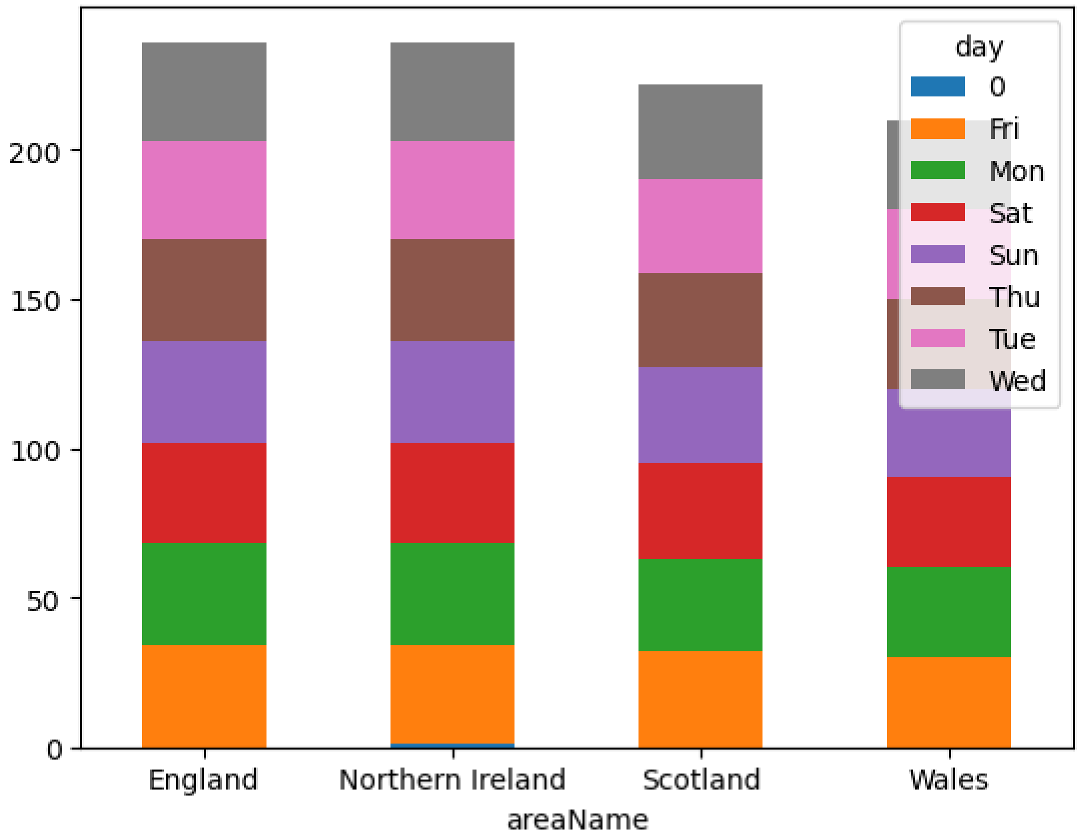
Out[34]:

day	0	Fri	Mon	Sat	Sun	Thu	Tue	Wed
areaName								
England	0	34	34	34	34	34	33	33
Northern Ireland	1	33	34	34	34	34	33	33
Scotland	0	32	31	32	32	32	31	32
Wales	0	30	30	30	30	30	30	30

It shows that the vaccination effort appears to be consistent throughout the week in all regions.

In [35]:
```
# plotting the contingency table as a stacked bar chart
cont_table.plot(kind="bar", stacked=True, rot=0)
```

Out[35]: <Axes: xlabel='areaName'>



In [36]:
```
#importing the data
from scipy import stats
# Conducting a statistical test of the independence between them using chi-square t
chi2, p_val, dof, expected = stats.chi2_contingency(cont_table)
print(f"p-value: {p_val}")
```

```
p-value: 0.9999988487406009
```

From the chi squared test we have the calculated p-value is greater than 0.05, we fail to reject the null hypothesis. Therefore, we can conclude that "day" and "AreaName" are not dependent and there is no significant relationship between them.

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

In [37]:
```
# Calculating descriptive statistics for FirstDose in the subset dataframe

FirstDose = df[df['areaName'] == 'England']['FirstDose']

# Calculating mean lead time
FirstDose.describe()
```

Out[37]:
```
count       236.000000
mean      16869.427966
std       12752.457581
min         955.000000
25%        7088.250000
50%       14223.000000
75%       23470.250000
max      115551.000000
Name: FirstDose, dtype: float64
```

In [38]:
```
# Calculating descriptive statistics for FirstDose in the subset dataframe
SecondDose = df[df['areaName'] == 'England']['SecondDose']

# Calculating mean lead time
SecondDose.describe()
```

Out[38]:
```
count      236.000000
mean     18469.016949
std       9615.471011
min        916.000000
25%      10887.000000
50%      17904.000000
75%      24306.500000
max      48491.000000
Name: SecondDose, dtype: float64
```

Both doses of vaccines administered exhibits some variability.

7Conduct a statistical test of the significance of the difference between the means of two subsets of the data and interpret the results

In [39]:
```
# Calculating the paired t-test for t samples between two groups: FirstDose and Se

t_val, p_val = stats.ttest_rel(FirstDose, SecondDose)
```

```
# This line prints the calculated t-value and p-value
print(f"t-value: {t_val}, p-value: {p_val}")
```

```
t-value: -2.366490490249986, p-value: 0.01876966907569725
```

p-value is 0.12457631582010473 and > 0.05 . Thus, we draw the inference that there is no observable difference between the mean values and fail to reject the null hypothesis.

8. 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 [40]:
```
#Grouping the data by the variables named "areaName" and summarizing the first, sec
grouped_data = df.groupby('areaName').agg({'FirstDose': 'sum', 'SecondDose': 'sum'

# Display the summarized data
print(grouped_data)
```

```
                  FirstDose  SecondDose   ThirdDose
areaName
England           3981185.0   4358688.0  32080017.0
Northern Ireland   117163.0    135440.0   1128833.0
Scotland           258958.0    348347.0   3270506.0
Wales              137585.0    179812.0   1712198.0
```

In [41]:
```
# Group the data by the "areaName" variable and calculate the mean value of the "F
grouped_data = df.groupby(['areaName'])['FirstDose', 'SecondDose', 'ThirdDose'].mea
```

```
C:\Users\rirti\AppData\Local\Temp\ipykernel_10868\344316704.py:2: FutureWarning: I
ndexing with multiple keys (implicitly converted to a tuple of keys) will be depre
cated, use a list instead.
  grouped_data = df.groupby(['areaName'])['FirstDose', 'SecondDose', 'ThirdDose'].
mean()
```

In [42]:
```
# Display the grouped data
print(grouped_data)
```

```
                     FirstDose    SecondDose      ThirdDose
areaName
England           16869.427966  18469.016949  135932.275424
Northern Ireland    496.453390    573.898305    4783.190678
Scotland           1166.477477   1569.130631   14732.009009
Wales               655.166667    856.247619    8153.323810
```

This one shows the average number of doses administered. The choice of which summary statistic to use depends on the analysis needs and research questions.

9 Multiple Regression

In [59]:
```
#importing libraries
import pandas as pd
import statsmodels.api as sm
```

```
#imports the output_notebook(), show(),Bokeh.io module
from bokeh.io import output_notebook
output_notebook()
from bokeh.plotting import figure
from bokeh.io import show
#input the data from directory
df.head(3)
```

BokehJS 2.4.3 successfully loaded.

Out[59]:

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

In [60]:

```
#conversion and encoding categorical variables in a pandas dataFrame object
df['areaName'] = df['areaName'].astype('category')
df['areaName'] = df['areaName'].cat.codes
df['Quarter'] = df['Quarter'].astype('category')
df['Quarter'] = df['Quarter'].cat.codes
df['day'] = df['day'].astype('category')
df['day'] = df['day'].cat.codes
df['WorkingDay'] = df['WorkingDay'].astype('category')
df['WorkingDay'] = df['WorkingDay'].cat.codes
df.head(500)
```

Out[60]:

	areaName	areaCode	year	month	Quarter	day	WorkingDay	FirstDose	SecondDose	T
0	0	E92000001	2022.0	5	2	2	2	3034.0	3857.0	
1	0	E92000001	2022.0	5	2	4	1	5331.0	3330.0	
2	0	E92000001	2022.0	5	2	3	1	13852.0	9759.0	
3	0	E92000001	2022.0	5	2	1	2	5818.0	5529.0	
4	0	E92000001	2022.0	5	2	5	2	8439.0	6968.0	
...	...	...	...	...	...	...	...	...	...	
495	2	S92000003	2022.0	4	2	1	2	398.0	737.0	
496	2	S92000003	2022.0	4	2	5	2	305.0	554.0	
497	2	S92000003	2022.0	4	2	7	2	322.0	545.0	
498	2	S92000003	2022.0	4	2	6	2	355.0	582.0	
499	2	S92000003	2022.0	4	2	2	2	427.0	688.0	

500 rows × 10 columns

In [61]:

```
# computing Pearson's correlation coefficients
corr_matrix = df.corr()
print(corr_matrix)
```

```
              areaName      year     month   Quarter       day  WorkingDay  \
areaName      1.000000 -0.016196 -0.034706 -0.048076  0.004951   -0.007440
year         -0.016196  1.000000  0.000710  0.001724 -0.016836   -0.021248
month        -0.034706  0.000710  1.000000  0.976501  0.000323    0.015092
Quarter      -0.048076  0.001724  0.976501  1.000000 -0.014238    0.004580
day           0.004951 -0.016836  0.000323 -0.014238  1.000000    0.157483
WorkingDay   -0.007440 -0.021248  0.015092  0.004580  0.157483    1.000000
FirstDose    -0.565147  0.014485  0.238436  0.265702  0.032789    0.035101
SecondDose   -0.642094  0.017447  0.106329  0.115529  0.016939    0.014019
ThirdDose    -0.405219  0.010292  0.385925  0.376870  0.026133    0.029622

             FirstDose  SecondDose  ThirdDose
areaName     -0.565147   -0.642094  -0.405219
year          0.014485    0.017447   0.010292
month         0.238436    0.106329   0.385925
Quarter       0.265702    0.115529   0.376870
day           0.032789    0.016939   0.026133
WorkingDay    0.035101    0.014019   0.029622
FirstDose     1.000000    0.835138   0.768961
SecondDose    0.835138    1.000000   0.766219
ThirdDose     0.768961    0.766219   1.000000
```

```
C:\Users\rirti\AppData\Local\Temp\ipykernel_10868\368849840.py:2: FutureWarning: T
he default value of numeric_only in DataFrame.corr is deprecated. In a future vers
ion, it will default to False. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  corr_matrix = df.corr()
```

The above code converts and encodes categorical variables, calculates pair-wise correlation coefficients between all pairs of columns in the dataframe.

In [62]:
```
# input the linear regression model formula with multiple independent variables
model = sm.OLS.from_formula('ThirdDose ~ FirstDose + SecondDose + areaName + year -

# Print a summary of the model statistics and coefficients
model.summary()
```

Out[62]:

OLS Regression Results

Dep. Variable:	ThirdDose	**R-squared:**	0.718
Model:	OLS	**Adj. R-squared:**	0.717
Method:	Least Squares	**F-statistic:**	458.0
Date:	Thu, 30 Mar 2023	**Prob (F-statistic):**	3.95e-244
Time:	09:12:43	**Log-Likelihood:**	-11158.
No. Observations:	904	**AIC:**	2.233e+04
Df Residuals:	898	**BIC:**	2.236e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-6.319e+04	5.59e+04	-1.131	0.259	-1.73e+05	4.65e+04
FirstDose	3.4138	0.365	9.362	0.000	2.698	4.129
SecondDose	6.5254	0.400	16.305	0.000	5.740	7.311
areaName	1.397e+04	2183.669	6.399	0.000	9687.390	1.83e+04
year	-3.1355	27.553	-0.114	0.909	-57.212	50.941
month	6439.1282	468.108	13.756	0.000	5520.415	7357.841

Omnibus:	520.198	**Durbin-Watson:**	0.287
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11808.871
Skew:	2.156	**Prob(JB):**	0.00
Kurtosis:	20.173	**Cond. No.**	4.46e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.46e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

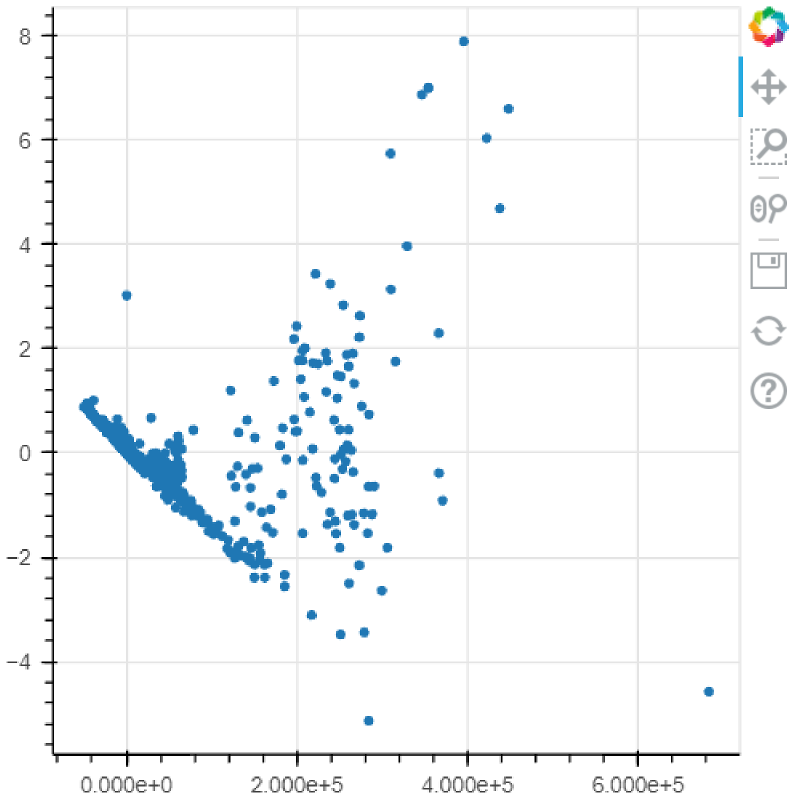
Coefficients:The intercept coefficient is -6.319e+04, and the FirstDose and SecondDose variables have coefficients of 3.4138 and 6.5254, respectively. The areaName variable has a coefficient of 1.397e+04, while the year variable has a coefficient of -3.1355, and the month variable has a coefficient of 6439.1282.

Thus, our model is described by the line: ThirdDose = -6.319e+04 + 3.4138FirstDose + 6.5254SecondDose + ... + e

Significance of the variables: According to the Adjusted R-squared value of 0.717, the model's predictors can account for 71.7% of the variation in the ThirdDose.

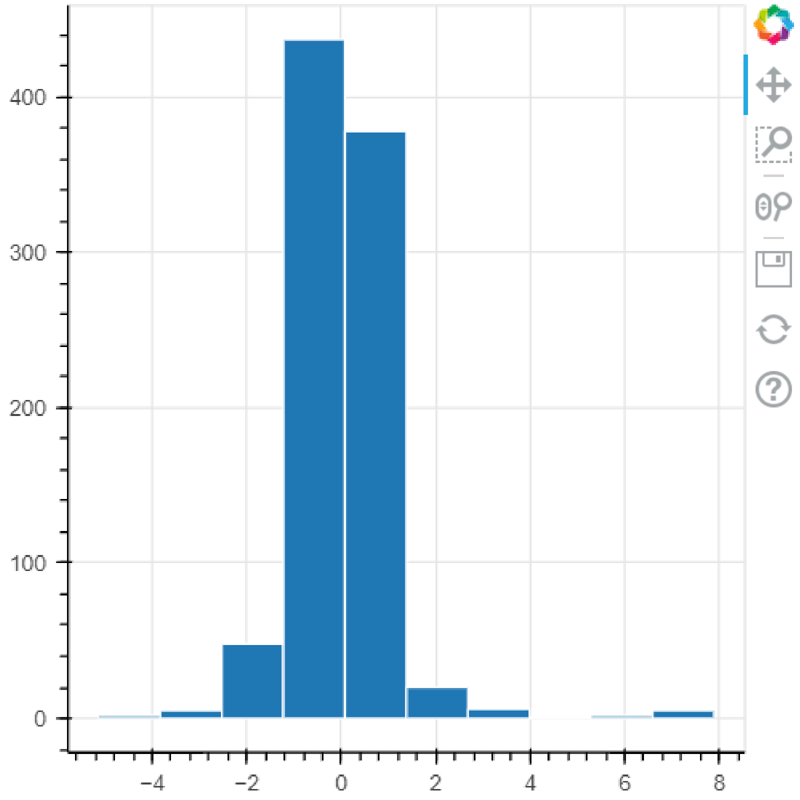
Quality of the Model:The presence of strong multicollinearity or other numerical problems in the model suggests that caution should be taken when interpreting the results.

In [63]:
```
#plotting the regression grPH
fig = figure(height=400, width=400)

# the x axis is the fitted values
# the y axis is the standardized residuals
st_resids = model.get_influence().resid_studentized_internal
fig.circle(model.fittedvalues, st_resids)

show(fig)
```



The Scater plot displays the non uniform pattern which can be inferred that homoscedasticity is complied as the residuals have constant variance

In [64]:
```
import numpy as np

# create a histogram with 10 bins
hist, edges = np.histogram(st_resids, bins=10)
fig = figure(height=400, width=400)
fig.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], line_color="white")
show(fig)
```



The historgram has a normal distribution of studentized residuals. Therefore, it indicates that that the model is a good bell curve and a good fit.

Conclusion

The code reads the data from an Excel file into a pandas dataframe and then performs tasks such as checking for missing values, creating histograms to visualize the distribution of vaccinations, counting the frequency of unique values, summarizing statistical measures, and performing t-tests. The methodology uses OLS regression to create a model and visualizes the residuals using Bokeh.

In a nut shell, a large portion of the population has received their first and second COVID-19 vaccine doses, with some areas, like London and the South East, demonstrating higher vaccination rates than others. The distribution of third doses is positively skewed, indicating disparities among regions. A strong positive correlation exists between the number of first and second doses given, and the timing of the vaccination, specifically the year and month, significantly impacts the number of third doses provided. The linear regression model predicts that the number of third doses distributed is significantly influenced by the number of first and second doses given, the region, and the vaccination's year and month. Finally,greater insights have been provided to the audience of this report to predict the third dose.

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

In [ ]:

In [ ]: