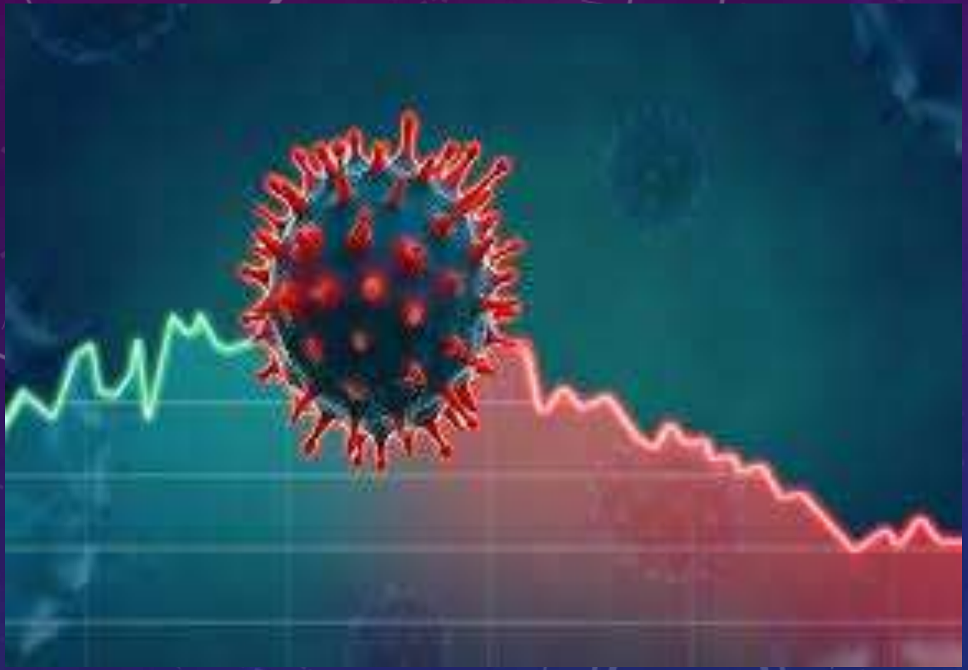
COVID-19 USING COGNOS



THE PROJECT INVOLVES ANALYZING COVID-19 CASES AND DEATHS USING IBM COGNOS

Covid-19 cases analysis:

1.Overall cases Trend:

Analyze the trend of COVID-19 cases over time, highlighting peaks, troughs, and any patterns.

2. Geographical Distribution:

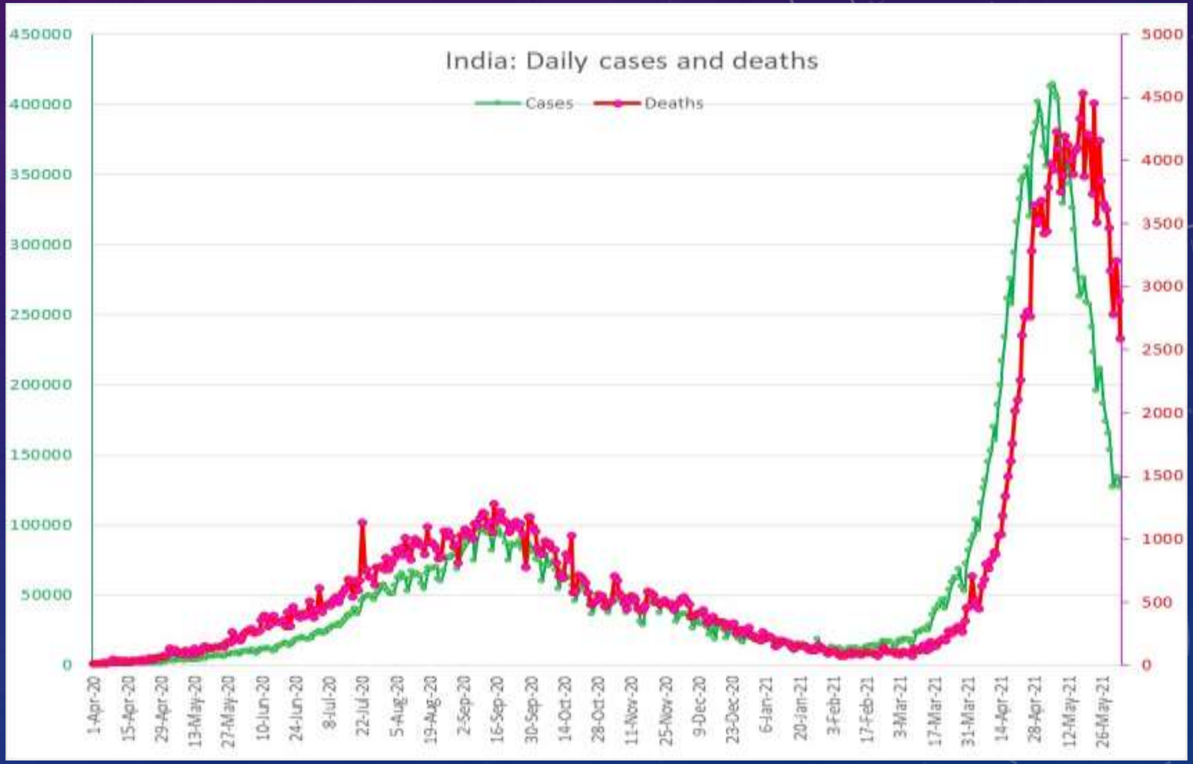
Explore how cases are distributed across different regions, countries, or continents.

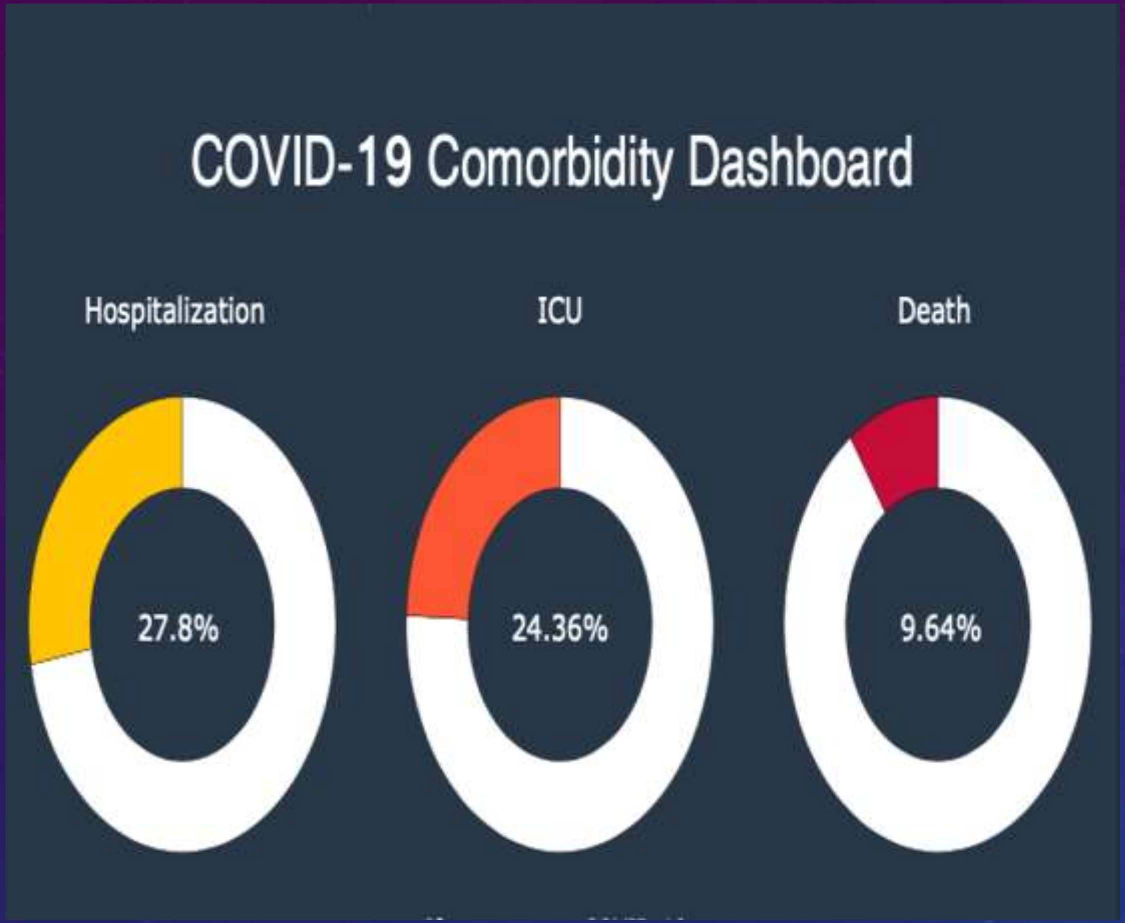
3. Demographics and Age Groups:

Break down cases by age groups to understand the impact on different demographics.

4.Testing and Positivity Rate:

Discuss testing efforts and how they correlate with the number of positive cases.





COVID-19 Comorbidity Dashboard

COVID-19 Deaths Analysis:

1. Death Rate Trend:

Analyze the trend of COVID-19 death rates over time and identify any variations.

2. Comorbidity Factors:

Investigate underlying health conditions and how they influence mortality rates.

3.Age and Mortality:

Examine how age affects the mortality rate and the vulnerability of different age groups

4.Healthcare System Impact:

Discuss how the capacity and efficiency of the healthcare system correlate with the number of COVID-19 deaths.

Education:

1.Shift to Remote Learning:

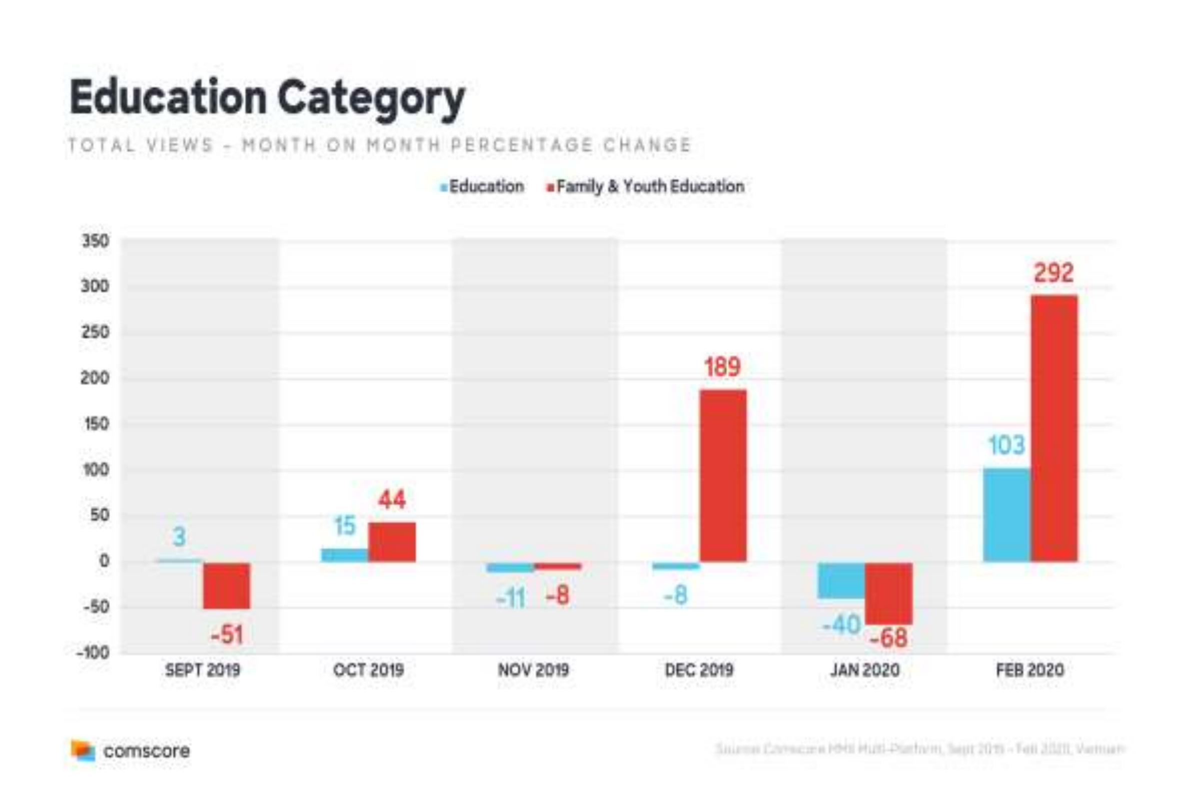
- The closure of schools and universities led to a rapid shift to online and remote learning. Educational institutions had to adapt quickly to deliver lessons and coursework through digital platforms.

2. Disruption in Academic Calendar:

- School closures disrupted academic calendars, affecting examination schedules, graduations, and overall progress for students.

3. Impact on Practical Learning:

- Fields that require hands-on experience, such as laboratory work or vocational training, were significantly impacted, posing challenges for students in those disciplines.



Education Category

TOTAL VIEWS - MONTH ON MONTH PERCENTAGE CHANGE

Source: Comscore MMX Multi-Platform, Sept 2019 - Feb 2020, Vietnam

conclusion:

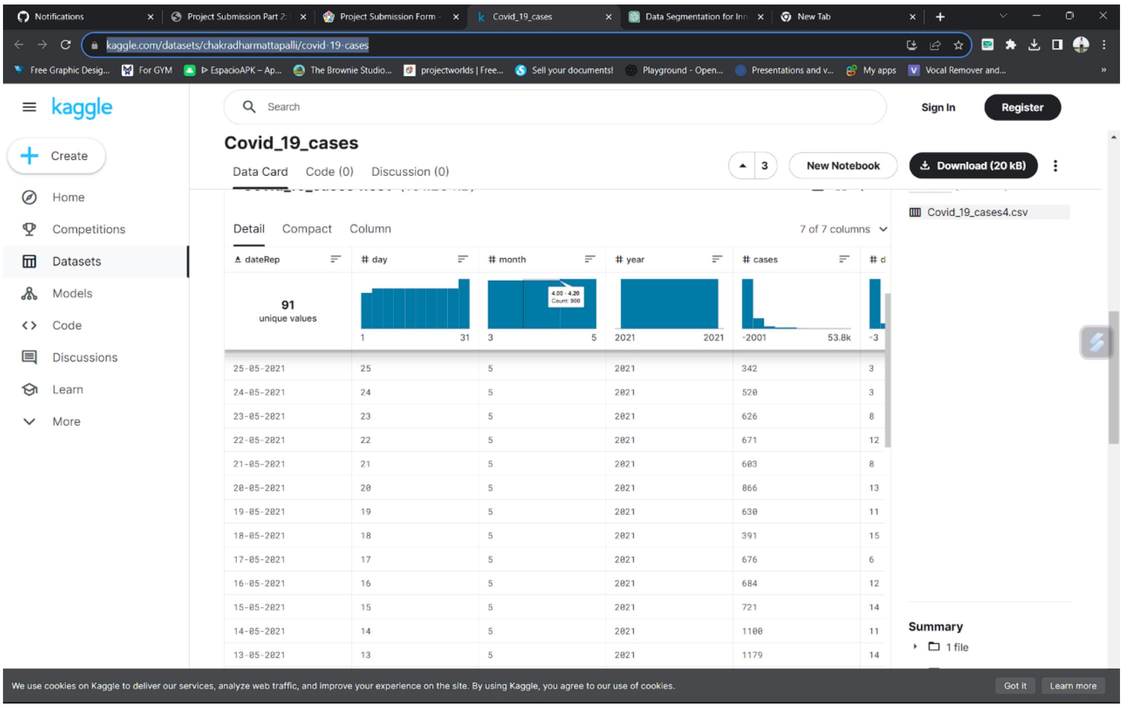
understanding the patterns, demographics, and impacts on both cases and deaths is vital for effective pandemic management, response strategies, and healthcare resource allocation. Tailored measures considering these factors are essential to mitigate the spread of COVID-19 and reduce its associated mortality.

Title: Data-Driven Innovation for Problem Solving

Introduction

In today's data-driven world, innovation is key to solving complex problems. Businesses and organizations are constantly seeking new ways to leverage data to gain deeper insights and make informed decisions. In this document, we will explore how data segmentation by time periods and countries can be used as an innovative solution to address specific issues.

Data Set Link : https://www.kaggle.com/datasets/chakradharmattapalli/covid-19-cases



dateRep	day	month	year	cases	d
25-05-2021	25	5	2021	342	3
24-05-2021	24	5	2021	520	3
23-05-2021	23	5	2021	626	8
22-05-2021	22	5	2021	671	12
21-05-2021	21	5	2021	603	8
20-05-2021	20	5	2021	866	13
19-05-2021	19	5	2021	630	11
18-05-2021	18	5	2021	391	15
17-05-2021	17	5	2021	676	6
16-05-2021	16	5	2021	684	12
15-05-2021	15	5	2021	721	14
14-05-2021	14	5	2021	1100	11
13-05-2021	13	5	2021	1179	14

Problem Statement

Define the problem you aim to solve. For instance, let's consider a retail business facing the challenge of declining sales. To solve this problem, we will focus on data-driven innovation.

Data Segmentation

1. Time Periods:

Daily Segmentation: Analyzing data on a daily basis can provide insights into daily sales trends. This can help identify specific days of the week or times of the day when sales are consistently lower.

Weekly Segmentation: Weekly segmentation can help identify sales patterns and trends over different weeks, allowing for the assessment of sales performance based on changes in marketing strategies or external factors.

Monthly and Quarterly Segmentation: These time periods can help in assessing longer-term trends, seasonal variations, and the impact of marketing campaigns.

2. Countries:

Geographic Segmentatio: Dividing data by countries can help identify regional variations in sales. This can be useful for tailoring marketing and sales strategies to specific regions.

Cultural and Economic Factors: Different countries have unique cultural and economic characteristics that can impact sales. Segmenting data by country can help in identifying the influence of these factors.

Innovative Data Analysis Techniques

1. Predictive Analytics: By analyzing historical data segmented by time periods, predictive analytics can be employed to forecast future sales trends. This can assist in planning inventory and staffing more effectively.

2.Customer Segmentation: By analyzing data by country and time periods, businesses can create customer segments based on buying habits. Tailored marketing campaigns can be then be designed for each segment.

3. A/B Testing: By comparing the performance of different strategies over time or across countries, businesses can use A/B testing to determine the most effective approaches.

Data Visualization and Reporting

To make the most of this innovative data segmentation, businesses should invest in robust data visualization tools and reporting systems. Dashboards and reports can provide a clear and concise view of the insights gathered from the segmented data. These visuals should be designed to highlight key performance indicators for both time periods and countries.

Benefits

- Enhanced Decision-Making: Data segmentation allows for more precise decision-making based on in-depth insights.
- Improved Resource Allocation: Businesses can allocate resources more effectively based on data-driven insights.
- Enhanced Customer Experience: Tailored marketing and sales strategies can improve the customer experience, leading to increased loyalty and sales.

Please provide specific data and details related to your problem statement to receive a more detailed analysis and design for implementing data segmentation for innovative problem-solving.

Source Code:

```
import pandas as pd

# Load your sales data into a DataFrame
data = pd.read_csv('sales_data.csv')

# Segmentation by time periods (e.g., monthly)
data['OrderDate'] = pd.to_datetime(data['OrderDate'])  # Convert date column to datetime
data['Month'] = data['OrderDate'].dt.to_period('M')  # Extract month from date

# Segmentation by countries
country_groups = data.groupby('Country')

# Calculate total sales per country
country_sales = country_groups['SalesAmount'].sum()

# Print the results
print("Monthly Sales:")
print(data.groupby('Month')['SalesAmount'].sum())

print("\nTotal Sales by Country:")
print(country_sales)
```

Conclusion

Data segmentation by time periods and countries is a powerful and innovative solution for problem-solving. By leveraging this approach, businesses can gain deeper insights into their operations, identify trends and patterns, and make informed decisions that drive growth and success.

Analysis of covid 19 data

Abstract

The pandemic of Coronavirus Disease 2019 (COVID-19) is a timely reminder of the nature and impact of Public Health Emergencies of International Concern. As of 12 January 2022, there were over 314 million cases and over 5.5 million deaths notified since the start of the pandemic. The COVID-19 pandemic takes variable shapes and forms, in terms of cases and deaths, in different regions and countries of the world. The objective of this study is to analyse the variable expression of COVID-19 pandemic so that lessons can be learned towards an effective public health emergency response

Methods

We conducted a mixed-methods study to understand the heterogeneity of cases and deaths due to the COVID-19 pandemic. Correlation analysis and scatter plot were employed for the quantitative data. We used Spearman’s correlation analysis to determine relationship strength between cases and deaths and socio-economic and health systems. We organized qualitative information from the literature and conducted a thematic analysis to recognize patterns of cases and deaths and explain the findings from the quantitative data.

STEPS TO PREPROCESSING THE DATASET

import pandas as pd

import scipy

import numpy as np

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

import matplotlib.pyplot as plt

df=pd.read_csv('20140171.CSV')

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10857234 entries, 0 to 10857233
Data columns (total 6 columns):
 #   Column                Dtype
---  ------                -----
 0   DataRep               object
 1   Day                   int91
 2   Month                 int91
 3   Year                  int91
 4   Cases                 int91
 5   Deaths                object
dtypes: int91(4), object(2)



memory usage: 497.0+ MB
```

In [83]:

df.head(5)

Out[83]:

	dateRep	day	Month	year	deaths	Countries and territories
0	**31-05-2021**	31	5	2021	5	Austria

	dateRep	day	Month	year	deaths	Countries and territories
1	**31-05-2021**	30	5	2021	6	Austria
2	**29-05-2021**	29	5	2021	11	Austria
3	**28-05-2021**	28	5	2021	4	Austria
4	**27-05-2021**	27	5	2021	19	Austria

```
df.tail(5)
```

```
Out[84]:
```

daterep	day	month	year	cases	deaths	Countries and territories
06-03-2021	6	3	2021	3455	17	sweden
05-03-2021	5	3	2021	4069	12	sweden
04-03-2021	4	3	2021	4884	14	sweden
03-03-2021	3	3	2021	4876	19	sweden
02-03-2021	2	3	2021	6191	19	sweden

df.isnull

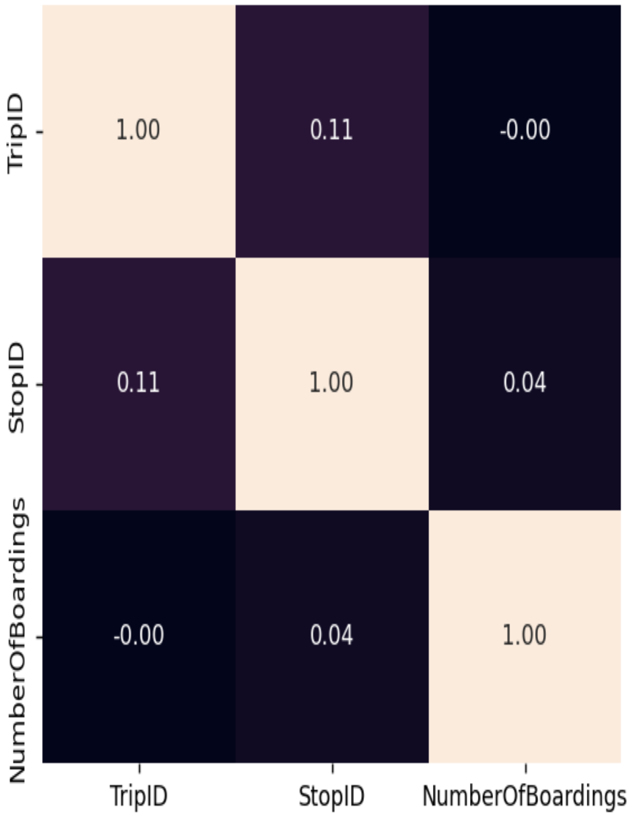
```
<bound method NDFrame._add_numeric_operations.<locals
>.sum of           TripID  RouteID  StopID  StopName
WeekBeginning  NumberOfBoardings
0          False    False   False     False
False              False
1          False    False   False     False
False              False
2          False    False   False     False
False              False
3          False    False   False     False
False              False
4          False    False   False     False
False              False
...          ...      ...     ...       ...
...                ...
10857229   False    False   False     False
False              False
10857230   False    False   False     False
False              False
10857231   False    False   False     False
False              False
10857232   False    False   False     False
False              False
10857233   False    False   False     False
False              False
```

```
df.describe()
```

	TripID	StopID	NumberOfBoardings
count	1.085723e+07	1.085723e+07	1.085723e+07
mean	2.952100e+04	1.366132e+04	4.743737e+00
std	1.960938e+04	1.971760e+03	9.382286e+00
min	7.900000e+01	1.000100e+04	1.000000e+00
25%	1.191700e+04	1.231100e+04	1.000000e+00
50%	2.747900e+04	1.334600e+04	2.000000e+00
75%	4.885800e+04	1.491600e+04	4.000000e+00
max	6.553500e+04	1.871500e+04	9.770000e+02

```
corr = df.corr()

plt.figure(dpi=130)
sns.heatmap(df.corr(), annot=True, fmt= '.2f')
plt.show()
```

	TripID	StopID	NumberOfBoardings



df.isnull().sum

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of
TripID  RouteID  StopID  StopName  WeekBeginning  NumberOfBoar
dings
0          False    False   False     False          False
False
1          False    False   False     False          False
False
2          False    False   False     False          False
False
3          False    False   False     False          False
False
4          False    False   False     False          False
False
...          ...      ...     ...       ...            ...
...
10857229   False    False   False     False          False
False
10857230   False    False   False     False          False
False
```

```
10857231    False     False    False      False            False
False
10857232    False     False    False      False            False
False
10857233    False     False    False      False            False
False

[10857234 rows x 6 columns]>
```

```
corr['StopID'].sort_values(ascending = False)
```

Out[69]:

```
StopID                  1.000000
TripID                  0.105974
NumberOfBoardings       0.038397
Name: StopID, dtype: float64
```

In [77]:

```
X = df.drop(columns =['StopID'])
Y = df.StopID
```

In [68]:

```
def mean_imputation(data, inplace = False):
    data.fillna(data.mean(), inplace = inplace)
```

In [ ]:

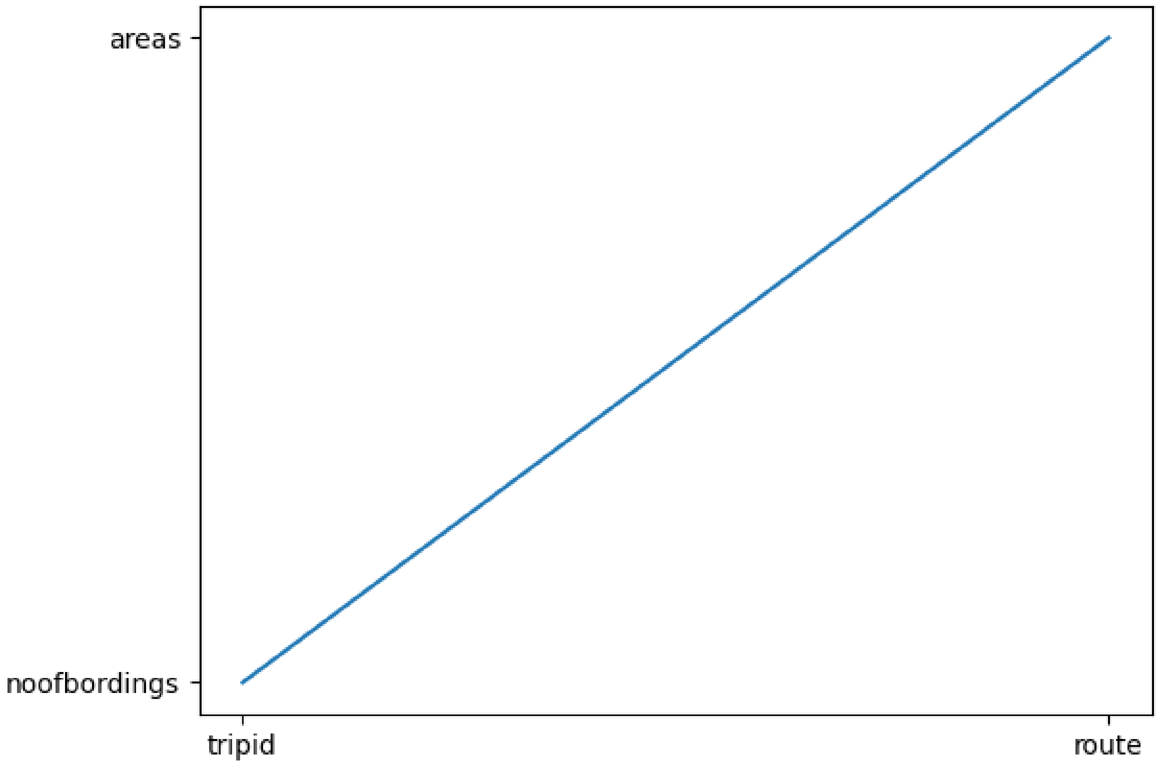
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_standardized = scaler.transform(X_train)
X_cv_standardized = scaler.transform(X_cv)
```

In [49]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import TomekLinks
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
def sampler_function(data_x, data_y, sampler = 0, random_state = 101):

    if sampler == 0:
        sampler = RandomOverSampler(random_state = random_state)
    elif sampler == 1:
        sampler = TomekLinks()
    elif sampler == 2:
        sampler = SMOTE()
    else:
        sampler = NearMiss()
    X_transformed, y_transformed = sampler.fit_resample(data_x, data_y)

    print('Original dataset shape:', Counter(data_y))
    print('Resample dataset shape:', Counter(y_transformed))

    return X_transformed, y_transformed
```

In [87]:

```
from matplotlib import pyplot as plt
x = ['tripid', 'route']
y = ['noofbordings', 'areas']
plt.plot(x, y)
plt.show()
```

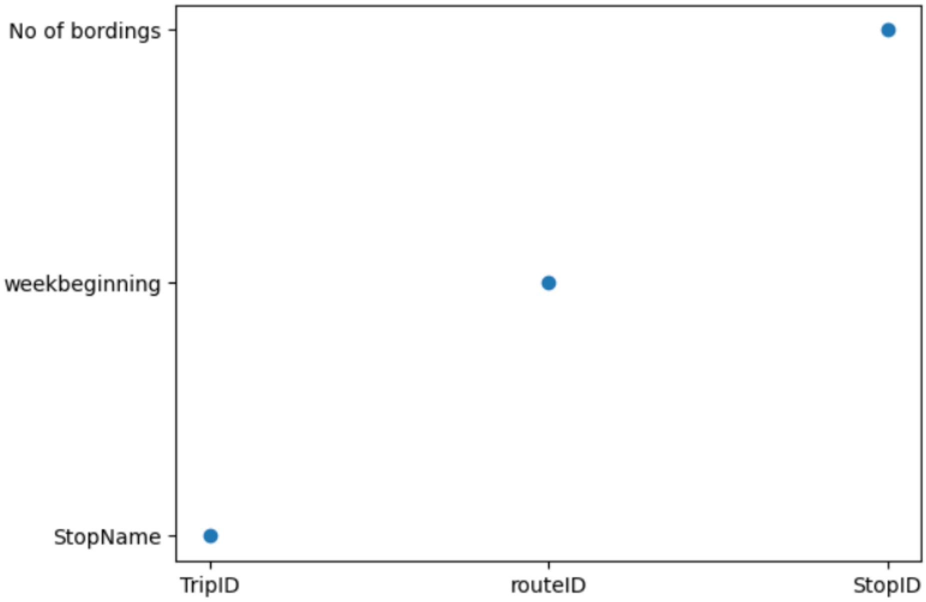


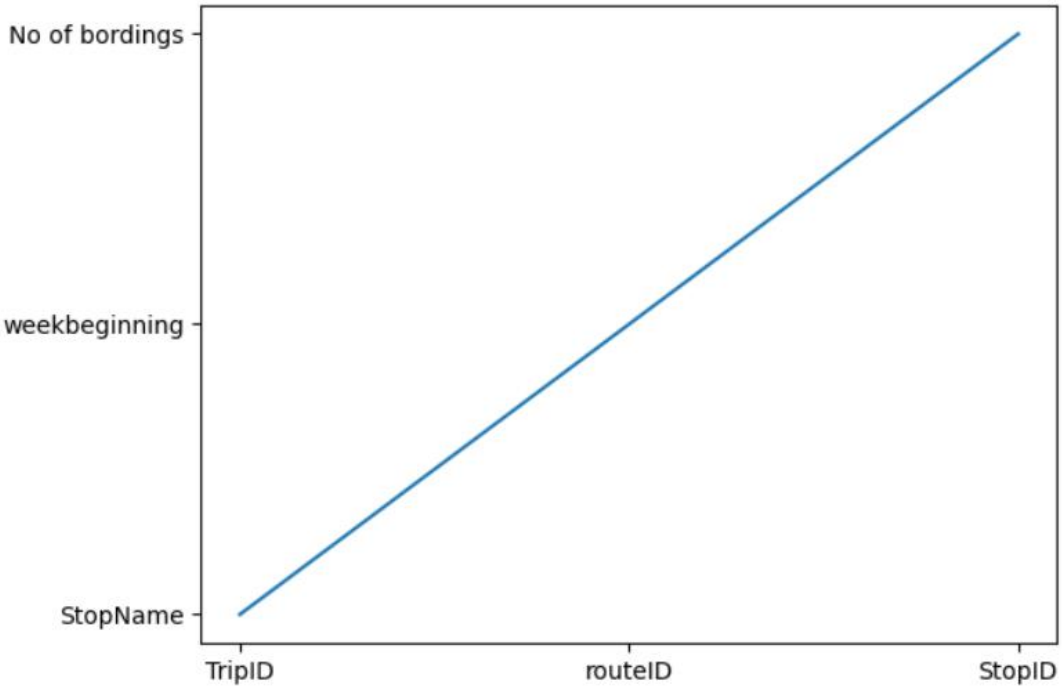
In [89]:

```
from matplotlib import pyplot as plt


x = ['TripID','routeID','StopID']
y = ['StopName','weekbeginning','No of bordings']
plt.scatter(x, y)


plt.show()
```

plt.plot(x, y)

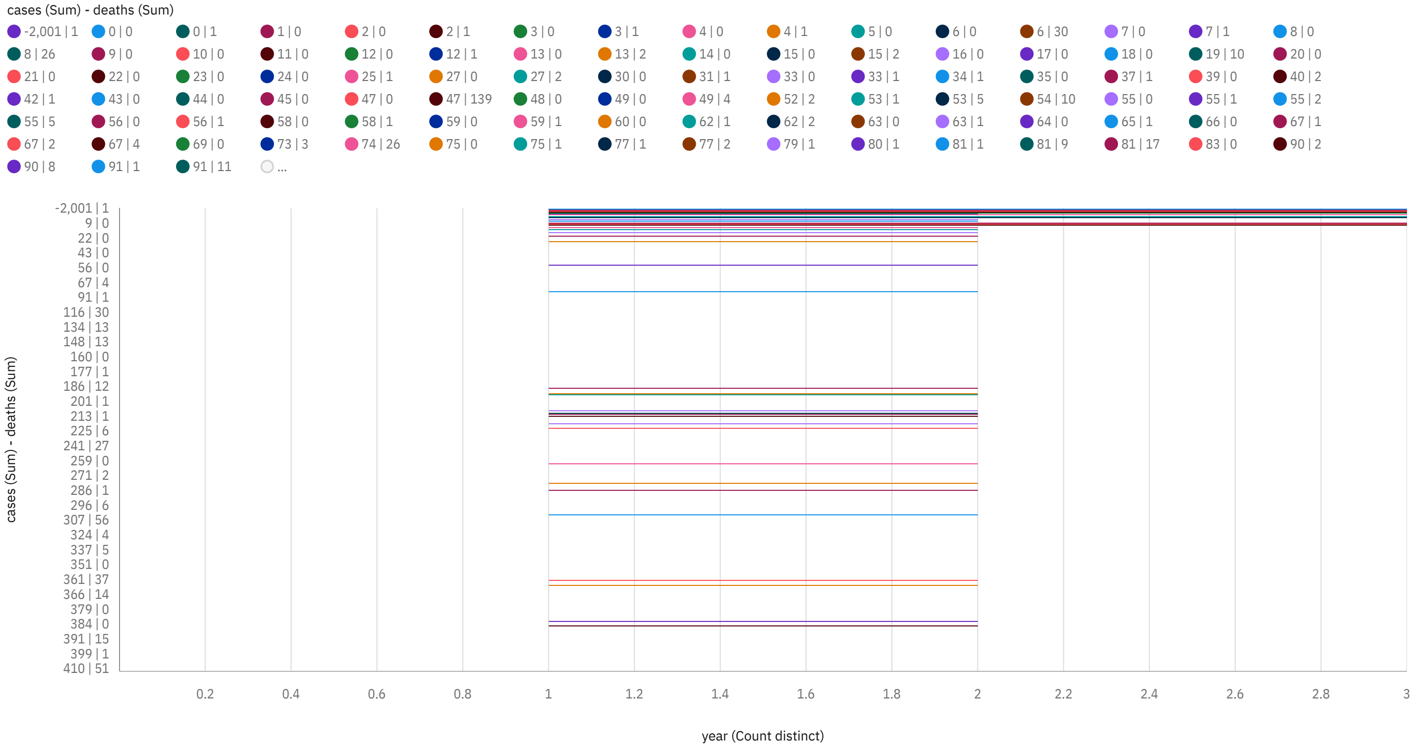
plt.show()





Tab 1

year, cases, deaths


cases (Sum) - deaths (Sum)
● -2,001 | 1
● 0 | 0
● 0 | 1
● 1 | 0
● 2 | 0
● 2 | 1
● 3 | 0
● 3 | 1
● 4 | 0
● 4 | 1
● 5 | 0
● 6 | 0
● 6 | 30
● 7 | 0
● 7 | 1
● 8 | 0
● 8 | 26
● 9 | 0
● 10 | 0
● 11 | 0
● 12 | 0
● 12 | 1
● 13 | 0
● 13 | 2
● 14 | 0
● 15 | 0
● 15 | 2
● 16 | 0
● 17 | 0
● 18 | 0
● 19 | 10
● 20 | 0
● 21 | 0
● 22 | 0
● 23 | 0
● 24 | 0
● 25 | 1
● 27 | 0
● 27 | 2
● 30 | 0
● 31 | 1
● 33 | 0
● 33 | 1
● 34 | 1
● 35 | 0
● 37 | 1
● 39 | 0
● 40 | 2
● 42 | 1
● 43 | 0
● 44 | 0
● 45 | 0
● 47 | 0
● 47 | 139
● 48 | 0
● 49 | 0
● 49 | 4
● 52 | 2
● 53 | 1
● 53 | 5
● 54 | 10
● 55 | 0
● 55 | 1
● 55 | 2
● 55 | 5
● 56 | 0
● 56 | 1
● 58 | 0
● 58 | 1
● 59 | 0
● 59 | 1
● 60 | 0
● 62 | 1
● 62 | 2
● 63 | 0
● 63 | 1
● 64 | 0
● 65 | 1
● 66 | 0
● 67 | 1
● 67 | 2
● 67 | 4
● 69 | 0
● 73 | 3
● 74 | 26
● 75 | 0
● 75 | 1
● 77 | 1
● 77 | 2
● 79 | 1
● 80 | 1
● 81 | 1
● 81 | 9
● 81 | 17
● 83 | 0
● 90 | 2
● 90 | 8
● 91 | 1
● 91 | 11
○ ...
cases (Sum) - deaths (Sum)
-2,001 | 1
9 | 0
22 | 0
43 | 0
56 | 0
67 | 4
91 | 1
116 | 30
134 | 13
148 | 13
160 | 0
177 | 1
186 | 12
201 | 1
213 | 1
225 | 6
241 | 27
259 | 0
271 | 2
286 | 1
296 | 6
307 | 56
324 | 4
337 | 5
351 | 0
361 | 37
366 | 14
379 | 0
384 | 0
391 | 15
399 | 1
410 | 51
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
year (Count distinct)