Load files

In [28]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

covid = pd.read_csv('covid.csv')
covid = pd.DataFrame(covid)
datewise = pd.read_csv('datewise.csv')
datewise = pd.DataFrame(datewise)
countrywise = pd.read_csv('countrywise.csv')
countrywise = pd.DataFrame(countrywise)
```

In [29]:

```
covid.index = pd.to_datetime(covid['ObservationDate'])
covid = covid.drop(columns='Unnamed: 0')
```

In [30]:

```
covid
```

Out[30]:

ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered	
2020-01-22	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0	
2020-01-22	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0	
2020-01-22	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0	
2020-01-22	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0	
2020-01-22	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0	
...	...	...	...	...	...	...	
2020-06-29	NaN	West Bank and Gaza	2020-06-30 04:33:48	2185.0	5.0	447.0	
2020-06-29	NaN	Western Sahara	2020-06-30 04:33:48	10.0	1.0	8.0	
2020-06-29	NaN	Yemen	2020-06-30 04:33:48	1128.0	304.0	432.0	
2020-06-29	NaN	Zambia	2020-06-30 04:33:48	1568.0	22.0	1311.0	
2020-06-29	NaN	Zimbabwe	2020-06-30 04:33:48	574.0	7.0	152.0	

341063 rows × 7 columns

In [31]:

```
datewise.head()
```

Out[31]:

	ObservationDate	Confirmed	Recovered	Deaths	Days Since	WeekOfYear	Mortalit Rat
0	2020-01-22	555.0	28.0	17.0	0 days 00:00:00.000000000	4	3.06306
1	2020-01-23	653.0	30.0	18.0	1 days 00:00:00.000000000	4	2.75650
2	2020-01-24	941.0	36.0	26.0	2 days 00:00:00.000000000	4	2.76301
3	2020-01-25	1438.0	39.0	42.0	3 days 00:00:00.000000000	4	2.92072
4	2020-01-26	2118.0	52.0	56.0	4 days 00:00:00.000000000	4	2.64400

In [32]:

```
datewise.index = datewise['ObservationDate']
datewise.index = pd.to_datetime(datewise.index)
datewise.drop(columns='ObservationDate')
```

Out[32]:

ObservationDate	Confirmed	Recovered	Deaths	Days Since	WeekOfYear	Mortality Rate
2020-01-22	555.0	28.0	17.0	0 days 00:00:00.000000000	4	3.063063
2020-01-23	653.0	30.0	18.0	1 days 00:00:00.000000000	4	2.756508
2020-01-24	941.0	36.0	26.0	2 days 00:00:00.000000000	4	2.763018
2020-01-25	1438.0	39.0	42.0	3 days 00:00:00.000000000	4	2.920723
2020-01-26	2118.0	52.0	56.0	4 days 00:00:00.000000000	4	2.644004
...	...	...	...	...	...	..
2020-06-25	9609829.0	4838921.0	489312.0	155 days 00:00:00.000000000	26	5.091787
2020-06-26	9801572.0	4945557.0	494181.0	156 days 00:00:00.000000000	26	5.041855
2020-06-27	9979535.0	5051864.0	498710.0	157 days 00:00:00.000000000	26	4.997327
2020-06-28	10145791.0	5140899.0	501893.0	158 days 00:00:00.000000000	26	4.946810
2020-06-29	10302151.0	5235813.0	505505.0	159 days 00:00:00.000000000	27	4.906791

160 rows × 9 columns

In [33]:

```
countrywise
```

Out[33]:

	Country/Region	Confirmed	Recovered	Deaths	Mortality	Recovery	Active Cases	C
0	US	2590651.0	705203.0	126140.0	4.869046	27.221073	1759308.0	831
1	Brazil	1368195.0	757811.0	58314.0	4.262112	55.387646	552070.0	816
2	Russia	640246.0	402778.0	9152.0	1.429451	62.909882	228316.0	411
3	India	566840.0	334822.0	16893.0	2.980206	59.068167	215125.0	351
4	United Kingdom	313470.0	1368.0	43659.0	13.927649	0.436405	268443.0	45
...	...	...	...	...	...	...	...	
183	Saint Kitts and Nevis	15.0	15.0	0.0	0.000000	100.000000	0.0	
184	Holy See	12.0	12.0	0.0	0.000000	100.000000	0.0	
185	Papua New Guinea	11.0	8.0	0.0	0.000000	72.727273	3.0	
186	Western Sahara	10.0	8.0	1.0	10.000000	80.000000	1.0	
187	MS Zaandam	9.0	0.0	2.0	22.222222	0.000000	7.0	

188 rows × 11 columns

In [34]:

```
countrywise.index = countrywise['Country/Region']
countrywise = countrywise.drop(columns='Country/Region')
```

In [ ]:

```

```

Prediction by Cluster using Machine Learning Models

In [35]:

```
print('Few Countries belonging to Cluster 0: ', list(countrywise[countrywise['Clusters']==0].hea
d(10).index))
print('Few Countries belonging to Cluster 1: ', list(countrywise[countrywise['Clusters']==1].hea
d(10).index))
print('Few Countries belonging to Cluster 2: ', list(countrywise[countrywise['Clusters']==2].hea
d(10).index))
```

```
Few Countries belonging to Cluster 0:  ['Russia', 'Chile', 'Iran', 'Turkey', 'Germ
any', 'Saudi Arabia', 'Canada', 'Qatar', 'Mainland China', 'Belarus']
Few Countries belonging to Cluster 1:  ['US', 'Brazil', 'India', 'Peru', 'Pakista
n', 'South Africa', 'Bangladesh', 'Colombia', 'Sweden', 'Egypt']
Few Countries belonging to Cluster 2:  ['United Kingdom', 'Spain', 'Italy', 'Mexic
o', 'France', 'Belgium', 'Netherlands', 'Hungary', 'Yemen', 'Bahamas']
```

In [36]:

```
cluster0 = countrywise[countrywise['Clusters']==0]
cluster1 = countrywise[countrywise['Clusters']==1]
cluster2 = countrywise[countrywise['Clusters']==2]
```

In [37]:

```
covid0 = covid[covid['Country/Region'].isin(cluster0.index)]
covid0 = covid0.drop(columns='ObservationDate')
datewise_cluster0 = covid0.groupby(['ObservationDate']).agg({"Confirmed":'sum',"Recovered":'sum'
,"Deaths":'sum'})

covid1 = covid[covid['Country/Region'].isin(cluster1.index)]
covid1 = covid1.drop(columns='ObservationDate')
datewise_cluster1 = covid1.groupby(['ObservationDate']).agg({"Confirmed":'sum',"Recovered":'sum'
,"Deaths":'sum'})

covid2 = covid[covid['Country/Region'].isin(cluster2.index)]
covid2 = covid2.drop(columns='ObservationDate')
datewise_cluster2 = covid2.groupby(['ObservationDate']).agg({"Confirmed":'sum',"Recovered":'sum'
,"Deaths":'sum'})
```

In [38]:

```
covid0
```

Out[38]:

ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
2020-01-22	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
2020-01-22	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2020-01-22	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
2020-01-22	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
2020-01-22	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
...	...	...	...	...	...	...
2020-06-29	NaN	United Arab Emirates	2020-06-30 04:33:48	48246.0	314.0	37076.0
2020-06-29	NaN	Uruguay	2020-06-30 04:33:48	932.0	27.0	822.0
2020-06-29	NaN	Uzbekistan	2020-06-30 04:33:48	8222.0	23.0	5496.0
2020-06-29	NaN	Vietnam	2020-06-30 04:33:48	355.0	0.0	335.0
2020-06-29	NaN	Zambia	2020-06-30 04:33:48	1568.0	22.0	1311.0

24250 rows × 6 columns

In [39]:

```
import matplotlib.pyplot as plt
import plotly.graph_objects as go

plt.figure(figsize=(10, 5))

fig = go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise_cluster0['Confirmed'],
                         mode='markers', name='Cluster 0 for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=datewise_cluster1['Confirmed'],
                         mode='markers', name='Cluster 1 for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=datewise_cluster2['Confirmed'],
                         mode='markers', name='Cluster 2 for Confirmed Cases'))

fig.update_layout(title='Confirmed Cases by Cluster',
                  xaxis_title='date', yaxis_title='Numver of Cases',
                  legend= dict(x=0, y=1, traceorder='normal'))
fig.show()
```

```
<Figure size 720x360 with 0 Axes>
```

It looks there are different patterns for each of the 3 Cluster Groups.

So, Analyze and predict the number of confirmed cases for each group.

In [40]:

```
datewise0.index
```

Out[40]:

```
Index(['2020-01-22', '2020-01-23', '2020-01-24', '2020-01-25', '2020-01-26',
       '2020-01-27', '2020-01-28', '2020-01-29', '2020-01-30', '2020-01-31',
       ...
       '2020-06-20', '2020-06-21', '2020-06-22', '2020-06-23', '2020-06-24',
       '2020-06-25', '2020-06-26', '2020-06-27', '2020-06-28', '2020-06-29'],
      dtype='object', name='ObservationDate', length=160)
```

In [41]:

```
datewise0 = covid0.groupby(
    ["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})
datewise0["Days Since"] = datewise0.index - datewise0.index.min()

datewise1 = covid1.groupby(
    ["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})
datewise1["Days Since"] = datewise1.index - datewise1.index.min()

datewise2 = covid2.groupby(
    ["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})
datewise2["Days Since"] = datewise2.index - datewise2.index.min()
```

In [42]:

```
import numpy as np

covid = covid0
datewise = datewise0
print("Basic Information of Cluster0")

print()

print("Total number of countries with Disease Spread: ", len(covid["Country/Region"].unique()))
print("Total number of Confirmed Cases around the World: ", datewise["Confirmed"].iloc[-1])
print("Total number of Recovered around the world: ",datewise["Recovered"].iloc[-1] )
print("Total number of Deaths Cases around the world: ", datewise["Deaths"].iloc[-1])
print("Total number of Active Cases around the world: ", (datewise["Confirmed"].iloc[-1] - datew
ise[
    "Recovered"].iloc[-1] - datewise["Deaths"].iloc[-1])) #Confirmed - Recovered - Deaths
print("Total number of Closed Cases around the world: ", datewise["Recovered"].iloc[-1]+datewise
["Deaths"].iloc[-1])

print()

print("Approximate number of Confirmed Cases per Day around the world: ",
      np.round(datewise["Confirmed"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Recovered Cases per Day around the world: ",
      np.round(datewise["Recovered"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Death Cases per Day around the world: ",
      np.round(datewise["Deaths"].iloc[-1]/datewise.shape[0]))

print()

print("Approximate number of Confirmed Cases per hour around the world: ",
      np.round(datewise["Confirmed"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Recovered Cases per hour around the word: ",
      np.round(datewise["Recovered"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Death Cases per hour around the world: ",
      np.round(datewise["Deaths"].iloc[-1]/(datewise.shape[0]*24)))

print()

print("Number of Confirmed Cases in last 24 hours: ", datewise["Confirmed"].iloc[-1]-datewise["C
onfirmed"].iloc[-2])
print("Number of Recovered Cases in last 24 hours: ", datewise["Recovered"].iloc[-1]-datewise["R
ecovered"].iloc[-2])
print("Number of Death Cases in last 24 hours: ", datewise["Deaths"].iloc[-1]-datewise["Deaths"]
.iloc[-2])
```

```
Basic Information of Cluster0

Total number of countries with Disease Spread:  94
Total number of Confirmed Cases around the World:  2641757.0
Total number of Recovered around the world:  2039365.0
Total number of Deaths Cases around the world:  70129.0
Total number of Active Cases around the world:  532263.0
Total number of Closed Cases around the world:  2109494.0

Approximate number of Confirmed Cases per Day around the world:  16511.0
Approximate number of Recovered Cases per Day around the world:  12746.0
Approximate number of Death Cases per Day around the world:  438.0

Approximate number of Confirmed Cases per hour around the world:  688.0
Approximate number of Recovered Cases per hour around the word:  531.0
Approximate number of Death Cases per hour around the world:  18.0

Number of Confirmed Cases in last 24 hours:  26257.0
Number of Recovered Cases in last 24 hours:  21985.0
Number of Death Cases in last 24 hours:  502.0
```

In [43]:

```
covid = covid1
datewise = datewise1
print("Basic Information of Cluster1")

print()

print("Total number of countries with Disease Spread: ", len(covid["Country/Region"].unique()))
print("Total number of Confirmed Cases around the World: ", datewise["Confirmed"].iloc[-1])
print("Total number of Recovered around the world: ",datewise["Recovered"].iloc[-1] )
print("Total number of Deaths Cases around the world: ", datewise["Deaths"].iloc[-1])
print("Total number of Active Cases around the world: ", (datewise["Confirmed"].iloc[-1] - datew
ise[
    "Recovered"].iloc[-1] - datewise["Deaths"].iloc[-1])) #Confirmed - Recovered - Deaths
print("Total number of Closed Cases around the world: ", datewise["Recovered"].iloc[-1]+datewise
["Deaths"].iloc[-1])

print()

print("Approximate number of Confirmed Cases per Day around the world: ",
     np.round(datewise["Confirmed"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Recovered Cases per Day around the world: ",
     np.round(datewise["Recovered"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Death Cases per Day around the world: ",
     np.round(datewise["Deaths"].iloc[-1]/datewise.shape[0]))

print()

print("Approximate number of Confirmed Cases per hour around the world: ",
     np.round(datewise["Confirmed"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Recovered Cases per hour around the word: ",
     np.round(datewise["Recovered"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Death Cases per hour around the world: ",
     np.round(datewise["Deaths"].iloc[-1]/(datewise.shape[0]*24)))

print()

print("Number of Confirmed Cases in last 24 hours: ", datewise["Confirmed"].iloc[-1]-datewise["C
onfirmed"].iloc[-2])
print("Number of Recovered Cases in last 24 hours: ", datewise["Recovered"].iloc[-1]-datewise["R
ecovered"].iloc[-2])
print("Number of Death Cases in last 24 hours: ", datewise["Deaths"].iloc[-1]-datewise["Deaths"]
.iloc[-2])
```

```
Basic Information of Cluster1

Total number of countries with Disease Spread:  82
Total number of Confirmed Cases around the World:  6318149.0
Total number of Recovered around the world:  2588898.0
Total number of Deaths Cases around the world:  254929.0
Total number of Active Cases around the world:  3474322.0
Total number of Closed Cases around the world:  2843827.0

Approximate number of Confirmed Cases per Day around the world:  39488.0
Approximate number of Recovered Cases per Day around the world:  16181.0
Approximate number of Death Cases per Day around the world:  1593.0

Approximate number of Confirmed Cases per hour around the world:  1645.0
Approximate number of Recovered Cases per hour around the word:  674.0
Approximate number of Death Cases per hour around the world:  66.0

Number of Confirmed Cases in last 24 hours:  122939.0
Number of Recovered Cases in last 24 hours:  66767.0
Number of Death Cases in last 24 hours:  2560.0
```

In [44]:

```
covid = covid2
datewise = datewise2
print("Basic Information of Cluster2")

print()

print("Total number of countries with Disease Spread: ", len(covid["Country/Region"].unique()))
print("Total number of Confirmed Cases around the World: ", datewise["Confirmed"].iloc[-1])
print("Total number of Recovered around the world: ",datewise["Recovered"].iloc[-1] )
print("Total number of Deaths Cases around the world: ", datewise["Deaths"].iloc[-1])
print("Total number of Active Cases around the world: ", (datewise["Confirmed"].iloc[-1] - datew
ise[
    "Recovered"].iloc[-1] - datewise["Deaths"].iloc[-1])) #Confirmed - Recovered - Deaths
print("Total number of Closed Cases around the world: ", datewise["Recovered"].iloc[-1]+datewise
["Deaths"].iloc[-1])

print()

print("Approximate number of Confirmed Cases per Day around the world: ",
     np.round(datewise["Confirmed"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Recovered Cases per Day around the world: ",
     np.round(datewise["Recovered"].iloc[-1]/datewise.shape[0]))
print("Approximate number of Death Cases per Day around the world: ",
     np.round(datewise["Deaths"].iloc[-1]/datewise.shape[0]))

print()

print("Approximate number of Confirmed Cases per hour around the world: ",
     np.round(datewise["Confirmed"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Recovered Cases per hour around the word: ",
     np.round(datewise["Recovered"].iloc[-1]/(datewise.shape[0]*24)))
print("Approximate number of Death Cases per hour around the world: ",
     np.round(datewise["Deaths"].iloc[-1]/(datewise.shape[0]*24)))

print()

print("Number of Confirmed Cases in last 24 hours: ", datewise["Confirmed"].iloc[-1]-datewise["C
onfirmed"].iloc[-2])
print("Number of Recovered Cases in last 24 hours: ", datewise["Recovered"].iloc[-1]-datewise["R
ecovered"].iloc[-2])
print("Number of Death Cases in last 24 hours: ", datewise["Deaths"].iloc[-1]-datewise["Deaths"]
.iloc[-2])
```

```
Basic Information of Cluster2

Total number of countries with Disease Spread:  12
Total number of Confirmed Cases around the World:  1342245.0
Total number of Recovered around the world:  607550.0
Total number of Deaths Cases around the world:  180447.0
Total number of Active Cases around the world:  554248.0
Total number of Closed Cases around the world:  787997.0

Approximate number of Confirmed Cases per Day around the world:  8442.0
Approximate number of Recovered Cases per Day around the world:  3821.0
Approximate number of Death Cases per Day around the world:  1135.0

Approximate number of Confirmed Cases per hour around the world:  352.0
Approximate number of Recovered Cases per hour around the word:  159.0
Approximate number of Death Cases per hour around the world:  47.0

Number of Confirmed Cases in last 24 hours:  7164.0
Number of Recovered Cases in last 24 hours:  6162.0
Number of Death Cases in last 24 hours:  550.0
```

In [45]:

```
covid0.index = pd.to_datetime(covid0.index)
covid1.index = pd.to_datetime(covid1.index)
covid2.index = pd.to_datetime(covid2.index)
datewise0.index = pd.to_datetime(datewise0.index)
datewise1.index = pd.to_datetime(datewise1.index)
datewise2.index = pd.to_datetime(datewise2.index)
```

In [46]:

```
datewise0["WeekOfYear"]=datewise0.index.weekofyear
datewise1["WeekOfYear"]=datewise1.index.weekofyear
datewise2["WeekOfYear"]=datewise2.index.weekofyear
```

In [ ]:

```

```

In [47]:

```
datewise = datewise0

datewise["WeekOfYear"]=datewise.index.weekofyear

week_num=[]
weekwise_confirmed=[]
weekwise_recovered=[]
weekwise_deaths=[]
w=1
for i in list(datewise["WeekOfYear"].unique()):
    weekwise_confirmed.append(datewise[datewise["WeekOfYear"]==i]["Confirmed"].iloc[-1])
    weekwise_recovered.append(datewise[datewise["WeekOfYear"]==i]["Recovered"].iloc[-1])
    weekwise_deaths.append(datewise[datewise["WeekOfYear"]==i]["Deaths"].iloc[-1])
    week_num.append(w)
    w=w+1

fig=go.Figure()
fig.add_trace(go.Scatter(x=week_num, y=weekwise_confirmed,
                    mode='lines+markers',
                    name='Weekly Growth of Confirmed Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_recovered,
                    mode='lines+markers',
                    name='Weekly Growth of Recovered Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_deaths,
                    mode='lines+markers',
                    name='Weekly Growth of Death Cases'))
fig.update_layout(title="Weekly Growth of different types of Cases in Cluster0",
                  xaxis_title="Week Number",yaxis_title="Number of Cases",legend=dict(x=0,y=1,tra
ceorder="normal"))
fig.show()
```

In [48]:

```
datewise = datewise1

datewise["WeekOfYear"]=datewise.index.weekofyear

week_num=[]
weekwise_confirmed=[]
weekwise_recovered=[]
weekwise_deaths=[]
w=1
for i in list(datewise["WeekOfYear"].unique()):
    weekwise_confirmed.append(datewise[datewise["WeekOfYear"]==i]["Confirmed"].iloc[-1])
    weekwise_recovered.append(datewise[datewise["WeekOfYear"]==i]["Recovered"].iloc[-1])
    weekwise_deaths.append(datewise[datewise["WeekOfYear"]==i]["Deaths"].iloc[-1])
    week_num.append(w)
    w=w+1

fig=go.Figure()
fig.add_trace(go.Scatter(x=week_num, y=weekwise_confirmed,
                    mode='lines+markers',
                    name='Weekly Growth of Confirmed Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_recovered,
                    mode='lines+markers',
                    name='Weekly Growth of Recovered Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_deaths,
                    mode='lines+markers',
                    name='Weekly Growth of Death Cases'))
fig.update_layout(title="Weekly Growth of different types of Cases in Cluster1",
                  xaxis_title="Week Number",yaxis_title="Number of Cases",legend=dict(x=0,y=1,tra
ceorder="normal"))
fig.show()
```

In [49]:

```
datewise = datewise2

datewise["WeekOfYear"]=datewise.index.weekofyear

week_num=[]
weekwise_confirmed=[]
weekwise_recovered=[]
weekwise_deaths=[]
w=1
for i in list(datewise["WeekOfYear"].unique()):
    weekwise_confirmed.append(datewise[datewise["WeekOfYear"]==i]["Confirmed"].iloc[-1])
    weekwise_recovered.append(datewise[datewise["WeekOfYear"]==i]["Recovered"].iloc[-1])
    weekwise_deaths.append(datewise[datewise["WeekOfYear"]==i]["Deaths"].iloc[-1])
    week_num.append(w)
    w=w+1

fig=go.Figure()
fig.add_trace(go.Scatter(x=week_num, y=weekwise_confirmed,
                    mode='lines+markers',
                    name='Weekly Growth of Confirmed Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_recovered,
                    mode='lines+markers',
                    name='Weekly Growth of Recovered Cases'))
fig.add_trace(go.Scatter(x=week_num, y=weekwise_deaths,
                    mode='lines+markers',
                    name='Weekly Growth of Death Cases'))
fig.update_layout(title="Weekly Growth of different types of Cases in Cluster2",
                  xaxis_title="Week Number",yaxis_title="Number of Cases",legend=dict(x=0,y=1,tra
ceorder="normal"))
fig.show()
```

In turn, they have different shapes:

convex downward curve, convex upward curve, and increasing straight line.

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

Prediction Confirmed cases according the clusters

In [467]:

```
# datewise = datewise0
# datewise = datewise1
datewise = datewise2
```

Linear Regression Model for Confirm Cases Prediction

In [468]:

```
datewise['Days Since'] = datewise.index - datewise.index[0]
datewise['Days Since'] = datewise['Days Since'].dt.days
```

In [469]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.95):]
model_scores=[]
```

In [470]:

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
```

In [471]:

```
import numpy as np

lin_reg.fit(np.array(train_ml['Days Since']).reshape(-1, 1),
            np.array(train_ml['Confirmed']).reshape(-1,1))
```

Out[471]:

```
LinearRegression(normalize=True)
```

In [472]:

```
prediction_valid_linreg = lin_reg.predict(np.array(valid_ml['Days Since']).reshape(-1,1))
```

In [473]:

```
from sklearn.metrics import mean_squared_error

model_scores.append(np.sqrt(mean_squared_error(valid_ml['Confirmed'],
                                               prediction_valid_linreg)))
print('Root Mean Square Error for Linear Regression: ',
      np.sqrt( mean_squared_error(valid_ml['Confirmed'], prediction_valid_linreg) ) )
```

```
Root Mean Square Error for Linear Regression:  62290.676478909285
```

In [474]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(11, 6))
prediction_linreg = lin_reg.predict(np.array(datewise['Days Since']).reshape(-1, 1))

linreg_output = []
for i in range(prediction_linreg.shape[0]):
    linreg_output.append(prediction_linreg[i][0])
```

```
<Figure size 792x432 with 0 Axes>
```

In [475]:

```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=linreg_output,
                         mode='lines', name='Linear Regression Best Fit Line',
              line = dict(color='black', dash='dot')))

fig.update_layout(title='Confirmed Cases Linear Regression Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                   legend=dict(x=0, y=1))

fig.show()
```

The Linear Regression Model is absolutely falling apart. As it is clearly visible that the trend of Confirmed Cases in absolutely not Linear.

In [476]:

```
from datetime import timedelta

new_date = []
new_prediction_lr = []

for i in range(1, 100):
    new_date.append(datewise.index[-1]+timedelta(days=i))
    new_prediction_lr.append(lin_reg.predict(np.array(datewise['Days Since'].max() + i).reshape(
-1,1))[0][0])
```

In [ ]:

```

```

Polynomial Regression for Prediction of Confirmed Cases

In [477]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.95):]
```

In [478]:

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 8)
```

In [479]:

```
train_poly=poly.fit_transform(np.array(train_ml["Days Since"]).reshape(-1,1))
valid_poly=poly.fit_transform(np.array(valid_ml["Days Since"]).reshape(-1,1))
y=train_ml["Confirmed"]
```

In [480]:

```
linreg=LinearRegression(normalize=True)
linreg.fit(train_poly,y)
```

Out[480]:

```
LinearRegression(normalize=True)
```

In [481]:

```
prediction_poly=linreg.predict(valid_poly)
rmse_poly=np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_poly))
model_scores.append(rmse_poly)
print("Root Mean Squared Error for Polynomial Regression: ",rmse_poly)
```

```
Root Mean Squared Error for Polynomial Regression:  60825.623237753156
```

In [482]:

```
comp_data=poly.fit_transform(np.array(datewise["Days Since"]).reshape(-1,1))
plt.figure(figsize=(11,6))
predictions_poly=linreg.predict(comp_data)

fig=go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                    mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=datewise.index, y=predictions_poly,
                    mode='lines',name="Polynomial Regression Best Fit",
                    line=dict(color='black', dash='dot')))
fig.update_layout(title="Confirmed Cases Polynomial Regression Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",
                  legend=dict(x=0,y=1,traceorder="normal"))
fig.show()
```

<Figure size 792x432 with 0 Axes>

In [483]:

```
new_prediction_poly = []

for i in range(1, 100):
    new_date_poly = poly.fit_transform(np.array(datewise['Days Since'].max()+i).reshape(-1,1))
    new_prediction_poly.append(linreg.predict(new_date_poly)[0])
```

In [484]:

```
pd.set_option('display.float_format', lambda x: '%.6f' % x)

model_predictions = pd.DataFrame(zip(new_date, new_prediction_lr, new_prediction_poly),
                                 columns=['Dates', 'Linear Regression Prediction', 'Polynomial Re
gression Prediction'])
model_predictions.head()
```

Out[484]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction
0	2020-06-30	1427482.232025	1234325.771486
1	2020-07-01	1438232.302353	1223235.073952
2	2020-07-02	1448982.372680	1210625.133176
3	2020-07-03	1459732.443008	1196505.902317
4	2020-07-04	1470482.513336	1180905.269617

In [ ]:

```

```

Support Vector Machine ModelRegressor for Prediction of Confirmed Cases

Takes too long times ,so skip it

In [485]:

```
train_ml = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid_ml = datewise.iloc[int(datewise.shape[0]*0.9):]
```

In [486]:

```
from sklearn.svm import SVR

# Initializing SVR Model
svm = SVR(C=1, degree=5, kernel='poly', epsilon=0.01)
```

In [487]:

```
# Fitting model on the training data
svm.fit(np.array(train_ml['Days Since']).reshape(-1, 1), np.array(train_ml['Confirmed']).reshape
(-1,1))
```

Out[487]:

```
SVR(C=1, degree=5, epsilon=0.01, kernel='poly')
```

In [488]:

```
prediction_valid_svm = svm.predict(np.array(valid_ml['Days Since']).reshape(-1,1))
```

In [489]:

```
model_scores.append(np.sqrt(mean_square_error(valid_ml['Confirmed'], prediction_valid_svm)))
print('Root Mean Square Error for Support Vector Machine: ', np.sqrt(mean_squared_error(
    valid_ml['Confirmed'], prediction_valid_svm)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-489-78f770b00c7c> in <module>
----> 1 model_scores.append(np.sqrt(mean_square_error(valid_ml['Confirmed'], p
rediction_valid_svm)))
      2 print('Root Mean Square Error for Support Vector Machine: ', np.sqrt(mean_
squared_error(
      3     valid_ml['Confirmed'], prediction_valid_svm)))

NameError: name 'mean_square_error' is not defined
```

In [490]:

```
plt.figure(figsize=(11, 6))
prediction_svm = svm.predict(np.array(datewise['Days Since']).reshape(-1,1))

fig = go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=datewise.index, y=prediction_svm,
                         mode='lines', name='Support Vector Machine Best fit Kernel',
                         line=dict(color='black', dash='dot')))
fig.update_layout(title='Confirmed Cases Support Vector Machine Regressor Prediction',
                  xais_title='Date', yaxis_title='Confirmed Cases',
                   legend = dict(x=0,y=1, 'normal'))

fig.show()
```

```
  File "<ipython-input-490-5a59e741244f>", line 12
    legend = dict(x=0,y=1, 'normal'))
                           ^
SyntaxError: positional argument follows keyword argument
```

In [491]:

```
new_date = []
new_prediction_svm = []

for i in range(1, 100):
    new_date.append(datewise.index[-1]+timedelta(days=i))
    new_prediction_svm.append(svm.predict(np.array(datewise['Days Since'].max() + i).reshape(-1,
1))[0])
```

In [492]:

```
pd.set_option('display.float_format', lambda x: '%.6f' % x)

model_predictions = pd.DataFrame(zip(new_date_poly, new_prediction_lr, new_prediction_poly),
                                 columns=['Dates', 'Linear Regression Prediction', 'Polynomial Re
gression Prediction', 'SVM Prediction'])
model_predictions.head()
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
list_to_arrays(data, columns, coerce_float, dtype)
    499         result = _convert_object_array(
--> 500             content, columns, dtype=dtype, coerce_float=coerce_float
    501         )

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
convert_object_array(content, columns, coerce_float, dtype)
    582                 "{col:d} columns passed, passed data had "
--> 583                 "{con} columns".format(col=len(columns), con=len(co
ntent))
    584             )

AssertionError: 4 columns passed, passed data had 3 columns

The above exception was the direct cause of the following exception:

ValueError                                Traceback (most recent call last)
<ipython-input-492-7d81091c768c> in <module>
      2
      3 model_predictions = pd.DataFrame(zip(new_date_poly, new_prediction_lr, new
_prediction_poly),
----> 4                                    columns=['Dates', 'Linear Regres
sion Prediction', 'Polynomial Regression Prediction', 'SVM Prediction'])
      5 model_predictions.head()

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩frame.py in __init__(self, dat
a, index, columns, dtype, copy)
    448                     if is_named_tuple(data[0]) and columns is None:
    449                         columns = data[0]._fields
--> 450                     arrays, columns = to_arrays(data, columns, dtype=
dtype)
    451                     columns = ensure_index(columns)
    452

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in t
o_arrays(data, columns, coerce_float, dtype)
    462         return [], []  # columns if columns is not None else []
    463     if isinstance(data[0], (list, tuple)):
--> 464         return _list_to_arrays(data, columns, coerce_float=coerce_floa
t, dtype=dtype)
    465     elif isinstance(data[0], abc.Mapping):
    466         return _list_of_dict_to_arrays(

~₩Anaconda3₩lib₩site-packages₩pandas₩core₩internals₩construction.py in _
list_to_arrays(data, columns, coerce_float, dtype)
    501         )
    502     except AssertionError as e:
--> 503         raise ValueError(e) from e
    504     return result
    505

ValueError: 4 columns passed, passed data had 3 columns
```

Predictions of Linear Regression are nowhere close to actual values.

In [ ]:

```
```

Time Series Forecasting

Holt's Linear Model

In [493]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid = datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred = valid.copy()
```

In [494]:

```
from statsmodels.tsa.api import Holt, SimpleExpSmoothing, ExponentialSmoothing


holt = Holt(np.asarray(model_train['Confirmed'])).fit(
    smoothing_level=0.2, smoothing_slope=1.8, optimized=False)
```

In [495]:

```
y_pred['Holt'] = holt.forecast(len(valid))
model_scores.append(np.sqrt(mean_squared_error(y_pred['Confirmed'], y_pred['Holt'])))
print("Root Mean Square Error Holt's Linear Model: ", np.sqrt(mean_squared_error(y_pred['Confirm
ed'], y_pred['Holt'])))
```

```
Root Mean Square Error Holt's Linear Model:  4870.127846388703
```

In [496]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred['Holt'],
                         mode='lines+markers', name='Prediction of Confirmed Cases'))
fig.update_layout(title="Confirmed Cases Holt's Linear Model Prediction",
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                   legend = dict(x=0,y=1,traceorder='normal'))

fig.show()
```

In [497]:

```
from datetime import timedelta

holt_new_date = []
holt_new_prediction = []

for i in range(1, 100):
    holt_new_date.append(datewise.index[-1]+timedelta(days=i))
    holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holt's Linear Model Prediction"] = holt_new_prediction
model_predictions.head()
```

Out[497]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644

In [ ]:

```

```

Holt's Winter Model for Daily Time Series

In [498]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid = datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred = valid.copy()
```

In [500]:

```
import warnings
warnings.filterwarnings('ignore')
# es = ExponentialSmoothing(np.asarray(model_train['Confirmed']),
#                           seasonal_periods = 11, trend='mul', seasonal='add').fit()
es = ExponentialSmoothing(np.asarray(model_train['Confirmed']),
                          seasonal_periods = 11, seasonal='add').fit()
```

In [501]:

```
y_pred["Holt's Winter Model"] = es.forecast(len(valid))
model_scores.append(np.sqrt(mean_squared_error(y_pred['Confirmed'],
                                               y_pred["Holt's Winter Model"])))
print("Root Mean Square Error for Holt's Winter Model: ",
     np.sqrt(mean_squared_error(y_pred['Confirmed'], y_pred["Holt's Winter Model"])))
```

```
Root Mean Square Error for Holt's Winter Model:  37779.75322926537
```

In [502]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt's Winter Model"],
                         mode='lines+markers', name='Prediction of Confirmed Cases'))
fig.update_layout(title="Confirmed Cases Holt's Winter Model Prediction",
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend= dict(x=0,y=1, traceorder='normal'))

fig.show()
```

In [503]:

```
holt_winter_new_prediction = []

for i in range(1, 100):
    holt_winter_new_prediction.append(es.forecast((len(valid)+i))[-1])

model_predictions["Holt's Winter Model Prediction"] = holt_winter_new_prediction
model_predictions.head()
```

Out[503]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252

In [ ]:

```

```

AR Model (using AUTO ARIMA)

In [504]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid = datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred = valid.copy()
```

In [505]:

```
# from tqdm.notebook import tqdm   # 진행표시바
# import time

# for i in tqdm(range(100), desc="ing"):
#     time.sleep(0.1)
```

In [506]:

```
from sys import executable
print(executable)
```

C:₩Users₩kheed₩Anaconda3₩python.exe

In [507]:

```
from pmdarima import *   # pyramid-arima 는 예전에 쓰던 모듈명
# from pmdarima import model_selection
```

In [508]:

```
model_ar = auto_arima(model_train['Confirmed'], trace=True, error_action='ignore',
                      start_p = 0, start_q=0, max_p=5, max_q=0,
                      suppress_warnings=True, stepwise=False, seasonal=False)
model_ar.fit(model_train['Confirmed'])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=2958.699, Time=0.02 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=2939.325, Time=0.03 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=2912.714, Time=0.03 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=2908.272, Time=0.05 sec
 ARIMA(4,2,0)(0,0,0)[0] intercept   : AIC=2909.125, Time=0.07 sec
 ARIMA(5,2,0)(0,0,0)[0] intercept   : AIC=2910.231, Time=0.08 sec
Total fit time: 0.285 seconds
```

Out[508]:

```
ARIMA(order=(3, 2, 0), scoring_args={}, suppress_warnings=True)
```

In [509]:

```
prediction_ar = model_ar.predict(len(valid))
y_pred["AR Model Prediction"] = prediction_ar
```

In [510]:

```
model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],
                                               y_pred["AR Model Prediction"])))
print("Root Mean Square Error for AR Model: ", np.sqrt(mean_squared_error(y_pred["Confirmed"],
                                                                          y_pred["AR Model Predic
tion"])))
```

```
Root Mean Square Error for AR Model:  2298.004181053553
```

In [511]:

```
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                    mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                    mode='lines+markers',name="Validation Data for Confirmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["AR Model Prediction"],
                    mode='lines+markers',name="Prediction of Confirmed Cases",))
fig.update_layout(title="Confirmed Cases AR Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder
="normal"))
fig.show()
```

In [512]:

```
AR_model_new_prediction = []
for i in range(1, 100):
    AR_model_new_prediction.append(model_ar.predict(len(valid)+i)[-1])

model_predictions["AR Model Prediction"] = AR_model_new_prediction
model_predictions.head()
```

Out[512]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Model Prediction
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518	1354997.537028
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278	1363171.008041
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256	1371397.533724
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940	1379679.532255
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252	1388012.782047

In [ ]:

```

```

In [ ]:

```

```

MA Model (using AUTO ARIMA)

In [513]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.95)]
valid = datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred = valid.copy()
```

In [514]:

```
model_ma = auto_arima(model_train["Confirmed"], trace=True, error_action='ignore',
                      start_p=0, start_q=0, max_p=0, max_q=5,
                      suppress_warnings=True, stepwise=False, seasonal=False)
model_ma.fit(model_train["Confirmed"])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=2958.699, Time=0.02 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=2910.764, Time=0.04 sec
 ARIMA(0,2,2)(0,0,0)[0] intercept   : AIC=2913.079, Time=0.06 sec
 ARIMA(0,2,3)(0,0,0)[0] intercept   : AIC=2906.823, Time=0.09 sec
 ARIMA(0,2,4)(0,0,0)[0] intercept   : AIC=2907.304, Time=0.11 sec
 ARIMA(0,2,5)(0,0,0)[0] intercept   : AIC=2911.207, Time=0.14 sec
Total fit time: 0.464 seconds
```

Out[514]:

```
ARIMA(order=(0, 2, 3), scoring_args={}, suppress_warnings=True)
```

In [515]:

```
prediction_ma = model_ar.predict(len(valid))
y_pred["MA Model Prediction"] = prediction_ma
```

In [516]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["MA Model Prediction"],
                          mode='lines+markers', name="Prediction for Confirmed Cases"))

fig.update_layout(title='Confirmed Cases MA Model Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend=dict(x=0, y=1, traceorder='normal'))

fig.show()
```

In [517]:

```
MA_model_new_prediction = []

for i in range(1, 100):
    MA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])

model_predictions["MA Model Prediction"] = MA_model_new_prediction
model_predictions
```

Out[517]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictic
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518	1354997.5370
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278	1363171.0080
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256	1371397.5337
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940	1379679.5322
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252	1388012.7820
...	...	...	...	...	...	
94	2020-10-02	2437988.842823	314642237.799825	2164547.619828	1283627.558735	2351528.5632
95	2020-10-03	2448738.913151	334071672.734841	2173108.150575	1283630.099566	2364607.3887
96	2020-10-04	2459488.983479	354507944.869660	2181668.681321	1283630.606759	2377738.3701
97	2020-10-05	2470239.053806	375993948.622174	2190229.212068	1283631.419019	2390921.5072
98	2020-10-06	2480989.124134	398574043.815988	2198789.742814	1283630.748242	2404156.8000

99 rows × 7 columns

In [ ]:

```

```

ARIMA Model (using AUTOARIMA)

In [518]:

```
model_train = datewise.iloc[:int(datewise.shape[0]*0.9)]
valid = datewise.iloc[int(datewise.shape[0]*0.9):]
y_pred = valid.copy()
```

In [519]:

```
model_arima = auto_arima(model_train['Confirmed'], trace=True, error_action='ignore',
                         start_p=1, start_q=1, max_p=3, max_q=3,
                         suppress_warnings=True, stepwise=False, seasonal=False)
model_arima.fit(model_train['Confirmed'])
```

```
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=2807.568, Time=0.02 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=2762.241, Time=0.04 sec
 ARIMA(0,2,2)(0,0,0)[0] intercept   : AIC=2764.534, Time=0.06 sec
 ARIMA(0,2,3)(0,0,0)[0] intercept   : AIC=2758.634, Time=0.07 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=2789.318, Time=0.02 sec
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=2764.156, Time=0.05 sec
 ARIMA(1,2,2)(0,0,0)[0] intercept   : AIC=2764.071, Time=0.12 sec
 ARIMA(1,2,3)(0,0,0)[0] intercept   : AIC=2759.739, Time=0.17 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=2763.986, Time=0.04 sec
 ARIMA(2,2,1)(0,0,0)[0] intercept   : AIC=2758.504, Time=0.09 sec
 ARIMA(2,2,2)(0,0,0)[0] intercept   : AIC=2760.634, Time=0.14 sec
 ARIMA(2,2,3)(0,0,0)[0] intercept   : AIC=2759.781, Time=0.62 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=2759.913, Time=0.05 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=2760.413, Time=0.17 sec
 ARIMA(3,2,2)(0,0,0)[0] intercept   : AIC=2761.976, Time=0.43 sec
Total fit time: 2.098 seconds
```

Out[519]:

```
ARIMA(order=(2, 2, 1), scoring_args={}, suppress_warnings=True)
```

In [520]:

```
prediction_arima = model_arima.predict(len(valid))
y_pred["ARIMA Model Prediction"] = prediction_arima
```

In [521]:

```
model_scores.append(np.sqrt(mean_squared_error(valid['Confirmed'], prediction_arima)))
print("Root Mean Squares Error for ARIMA Model: ", np.sqrt(mean_squared_error(valid['Confirmed'
], prediction_arima)))
```

```
Root Mean Squares Error for ARIMA Model:  7313.003091775848
```

In [522]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train['Confirmed'],
                         mode='lines+markers', name='Train Data for Confirmed Cases'))
fig.add_trace(go.Scatter(x=valid.index, y=valid['Confirmed'],
                         mode='lines+markers', name='Validation Data for Confirmed cases'))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["ARIMA Model Prediction"],
                          mode='lines+markers', name="Prediction for Confirmed Cases"))

fig.update_layout(title='Confirmed Cases ARIMA Model Prediction',
                  xaxis_title='Date', yaxis_title='Confirmed Cases',
                  legend=dict(x=0, y=1, traceorder='normal'))

fig.show()
```

In [523]:

```
ARIMA_model_new_prediction = []

for i in range(1, 100):
    ARIMA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])

model_predictions["ARIMA Model Prediction"] = ARIMA_model_new_prediction
model_predictions.head()
```

Out[523]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Model Prediction
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518	1354997.537028
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278	1363171.008041
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256	1371397.533724
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940	1379679.532255
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252	1388012.782047

In [ ]:

```
```

SARIMA Model (using AUTO ARIMA)

In [524]:

```
model_sarima = auto_arima(model_train["Confirmed"], trace=True, error_action='ignore',
                          start_p=0, start_q=0, max_p=2, max_q=2, m=7,
                          suppress_warnings=True, stepwise=True, seasonal=True)
model_sarima.fit(model_train["Confirmed"])
```

```
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(1,0,1)[7]             : AIC=2807.912, Time=0.14 sec
 ARIMA(0,2,0)(0,0,0)[7]             : AIC=2805.594, Time=0.01 sec
 ARIMA(1,2,0)(1,0,0)[7]             : AIC=2789.344, Time=0.08 sec
 ARIMA(0,2,1)(0,0,1)[7]             : AIC=2760.938, Time=0.07 sec
 ARIMA(0,2,1)(0,0,0)[7]             : AIC=2760.164, Time=0.05 sec
 ARIMA(0,2,1)(1,0,0)[7]             : AIC=2761.496, Time=0.11 sec
 ARIMA(0,2,1)(1,0,1)[7]             : AIC=2759.185, Time=0.29 sec
 ARIMA(0,2,1)(2,0,1)[7]             : AIC=2758.357, Time=0.37 sec
 ARIMA(0,2,1)(2,0,0)[7]             : AIC=2757.538, Time=0.21 sec
 ARIMA(0,2,0)(2,0,0)[7]             : AIC=2797.362, Time=0.10 sec
 ARIMA(1,2,1)(2,0,0)[7]             : AIC=2759.212, Time=0.38 sec
 ARIMA(0,2,2)(2,0,0)[7]             : AIC=2759.238, Time=0.25 sec
 ARIMA(1,2,0)(2,0,0)[7]             : AIC=2776.884, Time=0.16 sec
 ARIMA(1,2,2)(2,0,0)[7]             : AIC=2760.447, Time=0.46 sec
 ARIMA(0,2,1)(2,0,0)[7] intercept   : AIC=2759.471, Time=0.22 sec

Best model:  ARIMA(0,2,1)(2,0,0)[7]
Total fit time: 2.920 seconds
```

Out[524]:

```
ARIMA(order=(0, 2, 1), scoring_args={}, seasonal_order=(2, 0, 0, 7),
      suppress_warnings=True, with_intercept=False)
```

In [525]:

```
prediction_sarima = model_sarima.predict(len(valid))
y_pred["SARIMA Model Prediction"] = prediction_sarima
```

In [531]:

```
model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"], y_pred["SARIMA Model Predict
ion"])))
print("Root Mean Square Error for SARIMA Model: ", np.sqrt(mean_squared_error(y_pred["Confirmed"
], y_pred["SARIMA Model Prediction"])))
```

```
Root Mean Square Error for SARIMA Model:  2987.212971735211
```

In [532]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["SARIMA Model Prediction"],
                         mode='lines+markers', name="Train Data for Confirmed Cases"))
fig.update_layout(title="Confirmed Cases SARIMA Model Prediction",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [533]:

```
SARIMA_model_new_prediction = []

for i in range(1, 100):
    SARIMA_model_new_prediction.append(model_sarima.predict(len(valid)+i)[-1])

model_predictions["SARIMA Model Prediction"] = SARIMA_model_new_prediction
model_predictions.head()
```

Out[533]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Model Prediction
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518	1354997.537028
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278	1363171.008041
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256	1371397.533724
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940	1379679.532255
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252	1388012.782047

In [ ]:

In [ ]:

In [ ]:

Summarization of Forecasts using different Models

Case1 : Cluster1

In [393]:

```
model_predictions_cluster1 = model_predictions
model_predictions_cluster1
```

Out[393]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Pre
0	2020-06-30	3833291.440078	5738146.097477	6320391.797495	6404201.939134	6282803.
1	2020-07-01	3864305.441704	5742444.665584	6441092.301092	6543982.148160	6407172.
2	2020-07-02	3895319.443331	5729584.963176	6561792.804688	6688560.389186	6525640.
3	2020-07-03	3926333.444958	5697558.084724	6682493.308285	6837090.019450	6653075.
4	2020-07-04	3957347.446584	5644204.803660	6803193.811881	6984825.963931	6772635.
...	...	...	...	...	...	
94	2020-10-02	6748607.592977	-1495512743.404660	17666239.135562	42385132.647010	21030824.
95	2020-10-03	6779621.594603	-1568965785.749074	17786939.639159	43206071.142179	21224479.
96	2020-10-04	6810635.596230	-1645458132.578820	17907640.142755	44042353.196937	21418907.
97	2020-10-05	6841649.597856	-1725094284.510598	18028340.646351	44894309.118142	21614111.
98	2020-10-06	6872663.599483	-1807981651.832482	18149041.149948	45761874.861450	21810088.

99 rows × 9 columns

In [396]:

```
model_names=["Linear Regression", "Polynomial Regression","Holt's Linear","Holt's Winter Model",
             "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Mode
l"]
model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model Name","Root Mean Square
d Error"]).sort_values(["Root Mean Squared Error"])
model_summary
```

Out[396]:

	Model Name	Root Mean Squared Error
3	Holt's Winter Model	44001.721669
2	Holt's Linear	76177.823534
4	Auto Regressive Model (AR)	100230.132317
5	Moving Average Model (MA)	176939.392096
6	ARIMA Model	255456.527738
7	SARIMA Model	255456.527738
1	Polynomial Regression	355280.018103
0	Linear Regression	2158644.361009

In [ ]:

In [ ]:

In [397]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise1.index, y=datewise1["Confirmed"],
                         mode='lines+markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["Ho
lt's Winter Model Prediction"],
                         mode='lines+markers', name="Holt's Winter Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["SA
RIMA Model Prediction"],
                         mode='lines+markers', name="SARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["Ho
lt's Linear Model Prediction"],
                         mode='lines+markers', name="Holt's Linear Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["AR
Model Prediction"],
                         mode='lines+markers', name="AR Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["MA
Model Prediction"],
                         mode='lines+markers', name="MA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["AR
IMA Model Prediction"],
                         mode='lines+markers', name="ARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["Li
near Regression Prediction"],
                         mode='lines+markers', name="Linear Regression Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster1["Dates"], y=model_predictions_cluster1["Po
lynomial Regression Prediction"],
                         mode='lines+markers', name="Polynomial Regression Model Prediction"))

fig.update_layout(title="Compare Prediction by the best model in Cluster1",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=-1, traceorder="normal"))

fig.show()
```

In [399]:

```
datewise = datewise1
model_predictions = model_predictions_cluster1

fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=(datewise["Confirmed"]-datewise["Confirmed"].shift
().fillna(0)),
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=(model_predictions["Holt's Winter Model
Prediction"]-model_predictions["Holt's Winter Model Prediction"].shift()),
                         mode='markers', name="ARIMA Model Prediction"))

fig.update_layout(title="Amount of Change by a day",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [ ]:

```

```

In [ ]:

```

```

Case2 : Cluster0

In [326]:

```
model_predictions_cluster0 = model_predictions
model_predictions_cluster0
```

Out[326]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictic
0	2020-06-30	2124270.113174	2879274.609469	2694615.250282	2666049.225714	2740705.7185
1	2020-07-01	2140563.993367	2947026.430366	2721954.929206	2694243.362910	2775201.2418
2	2020-07-02	2156857.873561	3020323.473853	2749294.608130	2724138.780884	2810069.9343
3	2020-07-03	2173151.753754	3099792.726438	2776634.287054	2753874.216283	2845591.9629
4	2020-07-04	2189445.633947	3186115.726791	2803973.965978	2784272.583417	2881299.9683
...	...	...	...	...	...	
94	2020-10-02	3655894.851342	643998945.821120	5264545.069132	6556142.305211	6970813.9122
95	2020-10-03	3672188.731535	677303948.846480	5291884.748056	6616131.259719	7026142.1487
96	2020-10-04	3688482.611728	712099894.531798	5319224.426980	6676811.575141	7081687.7159
97	2020-10-05	3704776.491922	748443028.557486	5346564.105904	6738164.784884	7137450.6138
98	2020-10-06	3721070.372115	786391332.890334	5373903.784828	6800149.368897	7193430.8425

99 rows × 9 columns

In [329]:

```
model_names=["Linear Regression", "Polynomial Regression","Holt's Linear","Holt's Winter Model",
             "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Mode
l"]
model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model Name","Root Mean Square
d Error"]).sort_values(["Root Mean Squared Error"])
model_summary
```

Out[329]:

	Model Name	Root Mean Squared Error
3	Holt's Winter Model	7306.711512
5	Moving Average Model (MA)	17017.861655
6	ARIMA Model	18416.033889
7	SARIMA Model	18416.033889
2	Holt's Linear	22695.646030
4	Auto Regressive Model (AR)	41156.823807
1	Polynomial Regression	100799.369166
0	Linear Regression	498568.202112

In [ ]:

```

```

In [ ]:

```

```

In [330]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise0.index, y=datewise0["Confirmed"],
                         mode='lines+markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["Ho
lt's Winter Model Prediction"],
                         mode='lines+markers', name="Holt's Winter Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["SA
RIMA Model Prediction"],
                         mode='lines+markers', name="SARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["Ho
lt's Linear Model Prediction"],
                         mode='lines+markers', name="Holt's Linear Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["AR
Model Prediction"],
                         mode='lines+markers', name="AR Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["MA
Model Prediction"],
                         mode='lines+markers', name="MA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["AR
IMA Model Prediction"],
                         mode='lines+markers', name="ARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["Li
near Regression Prediction"],
                         mode='lines+markers', name="Linear Regression Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster0["Dates"], y=model_predictions_cluster0["Po
lynomial Regression Prediction"],
                         mode='lines+markers', name="Polynomial Regression Model Prediction"))

fig.update_layout(title="Compare Prediction by the best model in Cluster0",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=-1, traceorder="normal"))

fig.show()
```

In [331]:

```
datewise = datewise0
model_predictions = model_predictions_cluster0
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=(datewise["Confirmed"]-datewise["Confirmed"].shift
().fillna(0)),
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=(model_predictions["Holt's Winter Model
Prediction"]-model_predictions["Holt's Winter Model Prediction"].shift()),
                         mode='markers', name="Holt's Winter Model Prediction"))

fig.update_layout(title="Amount of Change by a day",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [ ]:

```

```

Case3 : Cluster2

In [529]:

```
model_predictions_cluster2 = model_predictions
model_predictions_cluster2
```

Out[529]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Mod Predictic
0	2020-06-30	1427482.232025	1234325.771486	1359857.729658	1283631.243518	1354997.5370
1	2020-07-01	1438232.302353	1223235.073952	1368418.260405	1283635.689278	1363171.0080
2	2020-07-02	1448982.372680	1210625.133176	1376978.791151	1283634.247256	1371397.5337
3	2020-07-03	1459732.443008	1196505.902317	1385539.321898	1283634.433940	1379679.5322
4	2020-07-04	1470482.513336	1180905.269617	1394099.852644	1283636.456252	1388012.7820
...	...	...	...	...	...	
94	2020-10-02	2437988.842823	314642237.799825	2164547.619828	1283627.558735	2351528.5632
95	2020-10-03	2448738.913151	334071672.734841	2173108.150575	1283630.099566	2364607.3887
96	2020-10-04	2459488.983479	354507944.869660	2181668.681321	1283630.606759	2377738.3701
97	2020-10-05	2470239.053806	375993948.622174	2190229.212068	1283631.419019	2390921.5072
98	2020-10-06	2480989.124134	398574043.815988	2198789.742814	1283630.748242	2404156.8000

99 rows × 9 columns

In [534]:

```
model_names=["Linear Regression", "Polynomial Regression","Holt's Linear","Holt's Winter Model",
             "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Mode
l"]
model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model Name","Root Mean Square
d Error"]).sort_values(["Root Mean Squared Error"])
model_summary
```

Out[534]:

	Model Name	Root Mean Squared Error
4	Auto Regressive Model (AR)	2298.004181
6	ARIMA Model	2987.212972
7	SARIMA Model	2987.212972
2	Holt's Linear	4870.127846
5	Moving Average Model (MA)	7313.003092
3	Holt's Winter Model	37779.753229
1	Polynomial Regression	60825.623238
0	Linear Regression	62290.676479

In [ ]:

```

```

In [ ]:

```

```

In [535]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise2.index, y=datewise2["Confirmed"],
                         mode='lines+markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["Ho
lt's Winter Model Prediction"],
                         mode='lines+markers', name="Holt's Winter Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["SA
RIMA Model Prediction"],
                         mode='lines+markers', name="SARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["Ho
lt's Linear Model Prediction"],
                         mode='lines+markers', name="Holt's Linear Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["AR
Model Prediction"],
                         mode='lines+markers', name="AR Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["MA
Model Prediction"],
                         mode='lines+markers', name="MA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["AR
IMA Model Prediction"],
                         mode='lines+markers', name="ARIMA Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["Li
near Regression Prediction"],
                         mode='lines+markers', name="Linear Regression Model Prediction"))
fig.add_trace(go.Scatter(x=model_predictions_cluster2["Dates"], y=model_predictions_cluster2["Po
lynomial Regression Prediction"],
                         mode='lines+markers', name="Polynomial Regression Model Prediction"))

fig.update_layout(title="Compare Prediction by the best model in Cluster2",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=-1, traceorder="normal"))

fig.show()
```

In [538]:

```
datewise = datewise2
model_predictions = model_predictions_cluster2
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=(datewise["Confirmed"]-datewise["Confirmed"].shift
().fillna(0)),
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=(model_predictions["AR Model Predictio
n"]-model_predictions["AR Model Prediction"].shift()),
                         mode='markers', name="AR Model Prediction"))

fig.update_layout(title="Amount of Change by a day",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [ ]:

```

```

In [ ]:

```

```

Compare Each 4 Best Model

In [339]:

```
model_predictions = pd.read_csv('model_predictions.csv')
```

In [340]:

```
model_predictions = model_predictions.drop(columns=["Unnamed: 0"])
model_predictions
```

Out[340]:

	Dates	Linear Regression Prediction	Polynomial Regression Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction	AR Pre
0	2020-06-30	7380259.723910	9798236.241235	10086804.349516	10573460.804313	10083394.
1	2020-07-01	7438229.365146	9847282.240619	10224167.230159	10761314.342326	10229086.
2	2020-07-02	7496199.006382	9881310.459843	10361530.110801	10953361.048453	10375078.
3	2020-07-03	7554168.647618	9898732.751112	10498892.991443	11147808.718753	10522468.
4	2020-07-04	7612138.288853	9897865.109540	10636255.872085	11345481.962612	10669943.
...	...	...	...	...	...	
94	2020-10-02	12829406.000087	-737621895.380941	22998915.129886	47802323.311642	27748386.
95	2020-10-03	12887375.641323	-768602963.472418	23136278.010528	48532856.405936	27980091.
96	2020-10-04	12945345.282559	-800575250.229044	23273640.891170	49274231.846084	28212718.
97	2020-10-05	13003314.923795	-833560667.977087	23411003.771812	50026595.166598	28446267.
98	2020-10-06	13061284.565031	-867581334.739752	23548366.652455	50790049.273035	28680737.

99 rows × 9 columns

In [345]:

```
model_predictions_cluster = model_predictions_cluster0["Holt's Winter Model Prediction"] + model
_predictions_cluster1["ARIMA Model Prediction"] + model_predictions_cluster1["Holt's Winter Mode
l Prediction"]
model_predictions_cluster
```

Out[345]:

```
0      9126764.733856
1      9277022.375453
2      9432221.631571
3      9591523.375934
4      9750182.740445
            ...
94    46745487.206533
95    47590951.437716
96    48451906.665152
97    49328685.534404
98    50221222.213023
Length: 99, dtype: float64
```

In [349]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                         mode='markers', name="Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions["Holt's Winter Model
 Prediction"],
                         mode='markers', name="Prediction all over the world"))
fig.add_trace(go.Scatter(x=model_predictions["Dates"], y=model_predictions_cluster,
                         mode='markers', name="Sum of Prediction in Clusters"))

fig.update_layout(title="Compare Prediction among Each 4 Best Model",
                  xaxis_title="Date", yaxis_title="Confirmed Cases",
                  legend=dict(x=0, y=1, traceorder="normal"))

fig.show()
```

In [ ]:

```

```