In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import pandas as pd
import random
import math
import time
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime
import operator
plt.style.use('seaborn-poster')
%matplotlib inline
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('retina')
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
# getting df
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVI
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-1
# recoveries_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/C
latest_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID
us_medical_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/C
apple_mobility = pd.read_csv("https://covid19-static.cdn-apple.com/covid19-mobili
```

In [5]:
```
latest_data.head()
```

Out[5]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed
0	NaN	NaN	NaN	Afghanistan	2022-02-03 04:21:21	33.93911	67.709953	164190
1	NaN	NaN	NaN	Albania	2022-02-03 04:21:21	41.15330	20.168300	261240
2	NaN	NaN	NaN	Algeria	2022-02-03 04:21:21	28.03390	1.659600	254885
3	NaN	NaN	NaN	Andorra	2022-02-03 04:21:21	42.50630	1.521800	36315
4	NaN	NaN	NaN	Angola	2022-02-03 04:21:21	-11.20270	17.873900	98267

In [6]:
```
confirmed_df.head()
```

Out[6]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0

5 rows × 750 columns

In [7]:
```
us_medical_data.head()
```

Out[7]:

	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovere
0	Alabama	US	2022-02-03 04:32:25	32.3182	-86.9023	1229300	17215	Na
1	Alaska	US	2022-02-03 04:32:25	61.3707	-152.4044	221077	1093	Na
2	American Samoa	US	2022-02-03 04:32:25	-14.2710	-170.1320	18	0	Na
3	Arizona	US	2022-02-03 04:32:25	33.7298	-111.4312	1886541	26369	Na
4	Arkansas	US	2022-02-03 04:32:25	34.9697	-92.3731	786010	9690	Na

In [8]:
```
cols = confirmed_df.keys()
```

In [9]:
```
# Get all the data for ongoing coronavirus
confirmed = confirmed_df.loc[:, cols[4]:cols[-1]]
deaths = deaths_df.loc[:, cols[4]:cols[-1]]
```

In [10]:
```
dates = confirmed.keys()
world_cases = []
total_deaths = []
mortality_rate = []

for i in dates:
    confirmed_sum = confirmed[i].sum()
    death_sum = deaths[i].sum()

    # confirmed, deaths, recovered, and active
    world_cases.append(confirmed_sum)
    total_deaths.append(death_sum)
    # calculate rates
    mortality_rate.append(death_sum/confirmed_sum)
```

In [11]:
```
def daily_increase(data):
    d = []
    for i in range(len(data)):
        if i == 0:
            d.append(data[0])
        else:
            d.append(data[i]-data[i-1])
    return d

def moving_average(data, window_size):
    moving_average = []
    for i in range(len(data)):
        if i + window_size < len(data):
            moving_average.append(np.mean(data[i:i+window_size]))
        else:
            moving_average.append(np.mean(data[i:len(data)]))
    return moving_average

# window size
window = 7

# confirmed cases
world_daily_increase = daily_increase(world_cases)
world_confirmed_avg= moving_average(world_cases, window)
world_daily_increase_avg = moving_average(world_daily_increase, window)

# deaths
world_daily_death = daily_increase(total_deaths)
world_death_avg = moving_average(total_deaths, window)
world_daily_death_avg = moving_average(world_daily_death, window)
```

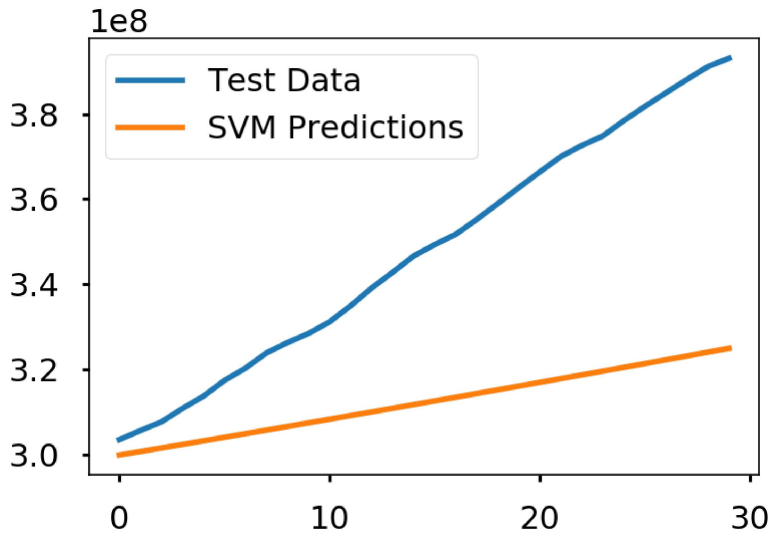
In [12]:
```
days_since_1_22 = np.array([i for i in range(len(dates))]).reshape(-1, 1)
world_cases = np.array(world_cases).reshape(-1, 1)
total_deaths = np.array(total_deaths).reshape(-1, 1)
```

In [13]:
```
# future forecasting
days_in_future = 10
future_forcast = np.array([i for i in range(len(dates)+days_in_future)]).reshape(
adjusted_dates = future_forcast[:-10]
```

In [14]:
```
start = '1/22/2020'
start_date = datetime.datetime.strptime(start, '%m/%d/%Y')
future_forcast_dates = []
for i in range(len(future_forcast)):
    future_forcast_dates.append((start_date + datetime.timedelta(days=i)).strftim

days_to_skip = 376
X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed = train_
```

In [15]:
```
#Model Prediction
svm_confirmed = SVR(shrinking=True, kernel='poly',gamma=0.01, epsilon=1,degree=3,
svm_confirmed.fit(X_train_confirmed, y_train_confirmed)
svm_pred = svm_confirmed.predict(future_forcast)
svm_test_pred = svm_confirmed.predict(X_test_confirmed)
plt.plot(y_test_confirmed)
plt.plot(svm_test_pred)
plt.legend(['Test Data', 'SVM Predictions'])
print('MAE:', mean_absolute_error(svm_test_pred, y_test_confirmed))
print('MSE:',mean_squared_error(svm_test_pred, y_test_confirmed))
```

```
MAE: 35461290.2143829
MSE: 1669114963770873.2
```



In [16]:
```
# transform our data for polynomial regression
poly = PolynomialFeatures(degree=2)
poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)
poly_X_test_confirmed = poly.fit_transform(X_test_confirmed)
poly_future_forcast = poly.fit_transform(future_forcast)

bayesian_poly = PolynomialFeatures(degree=2)
bayesian_poly_X_train_confirmed = bayesian_poly.fit_transform(X_train_confirmed)
bayesian_poly_X_test_confirmed = bayesian_poly.fit_transform(X_test_confirmed)
bayesian_poly_future_forcast = bayesian_poly.fit_transform(future_forcast)
```

In [17]:
```
# polynomial regression
linear_model = LinearRegression(normalize=True, fit_intercept=False)
linear_model.fit(poly_X_train_confirmed, y_train_confirmed)
test_linear_pred = linear_model.predict(poly_X_test_confirmed)
linear_pred = linear_model.predict(poly_future_forcast)
print('MAE:', mean_absolute_error(test_linear_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_linear_pred, y_test_confirmed))
```

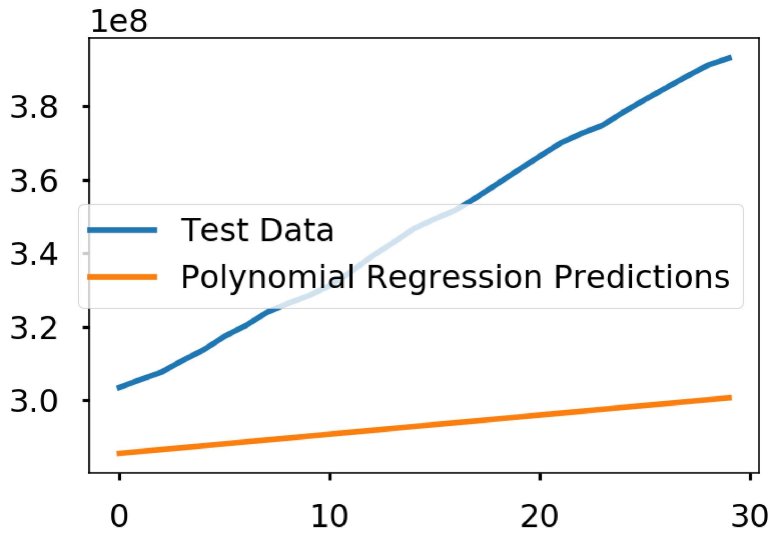
```
MAE: 54576518.135831974
MSE: 3519480447068585.0
```

In [18]:
```
print(linear_model.coef_)
```

```
[[-1.20883877e+08  6.12235653e+05 -6.22314286e+01]]
```

In [19]:
```
plt.plot(y_test_confirmed)
plt.plot(test_linear_pred)
plt.legend(['Test Data', 'Polynomial Regression Predictions'])
```

Out[19]:
```
<matplotlib.legend.Legend at 0x1ea735cf588>
```



In [20]:
```
tol = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2]
alpha_1 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
alpha_2 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
lambda_1 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
lambda_2 = [1e-7, 1e-6, 1e-5, 1e-4, 1e-3]
normalize = [True, False]

bayesian_grid = {'tol': tol, 'alpha_1': alpha_1, 'alpha_2' : alpha_2, 'lambda_1'
                 'normalize' : normalize}

bayesian = BayesianRidge(fit_intercept=False)
bayesian_search = RandomizedSearchCV(bayesian, bayesian_grid, scoring='neg_mean_
bayesian_search.fit(bayesian_poly_X_train_confirmed, y_train_confirmed)
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[20]:
```
RandomizedSearchCV(cv=3, estimator=BayesianRidge(fit_intercept=False),
                   n_iter=40, n_jobs=-1,
                   param_distributions={'alpha_1': [1e-07, 1e-06, 1e-05, 0.000
1,
                                                    0.001],
                                        'alpha_2': [1e-07, 1e-06, 1e-05, 0.000
1,
                                                    0.001],
                                        'lambda_1': [1e-07, 1e-06, 1e-05,
                                                     0.0001, 0.001],
                                        'lambda_2': [1e-07, 1e-06, 1e-05,
                                                     0.0001, 0.001],
                                        'normalize': [True, False],
                                        'tol': [1e-06, 1e-05, 0.0001, 0.001,
                                                0.01]},
                   return_train_score=True, scoring='neg_mean_squared_error',
                   verbose=1)
```

In [21]:
```
bayesian_search.best_params_
```

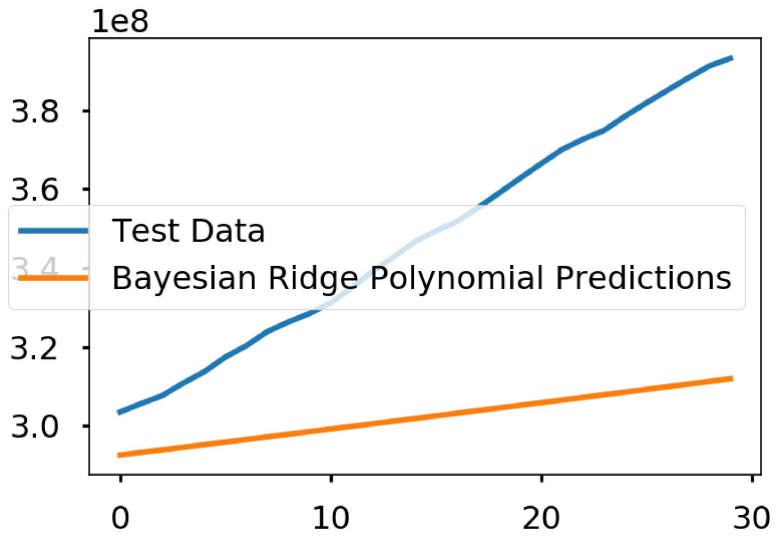
Out[21]:
```
{'tol': 1e-05,
 'normalize': True,
 'lambda_2': 1e-05,
 'lambda_1': 1e-07,
 'alpha_2': 1e-06,
 'alpha_1': 0.001}
```

In [22]:
```
bayesian_confirmed = bayesian_search.best_estimator_
test_bayesian_pred = bayesian_confirmed.predict(bayesian_poly_X_test_confirmed)
bayesian_pred = bayesian_confirmed.predict(bayesian_poly_future_forcast)
print('MAE:', mean_absolute_error(test_bayesian_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_bayesian_pred, y_test_confirmed))
```

```
MAE: 45490816.64220862
MSE: 2553455445929215.5
```

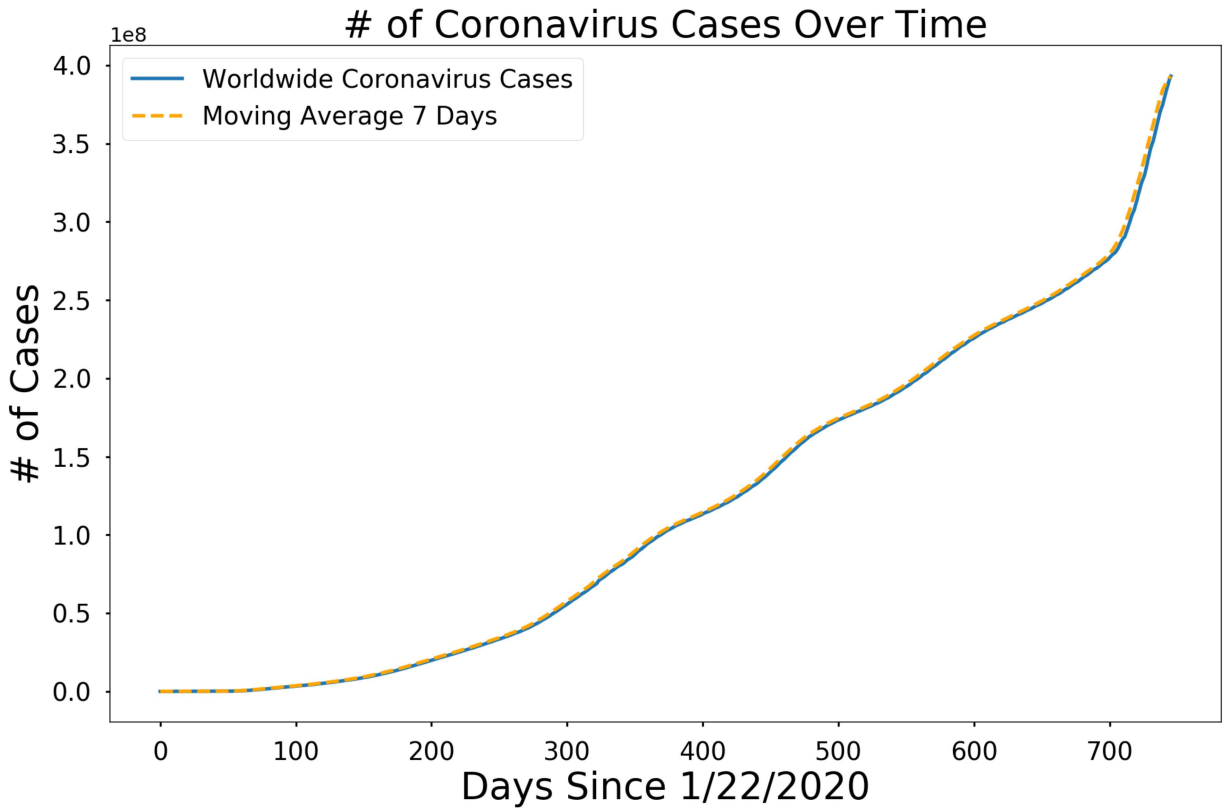
In [23]:
```
plt.plot(y_test_confirmed)
plt.plot(test_bayesian_pred)
plt.legend(['Test Data', 'Bayesian Ridge Polynomial Predictions'])
```

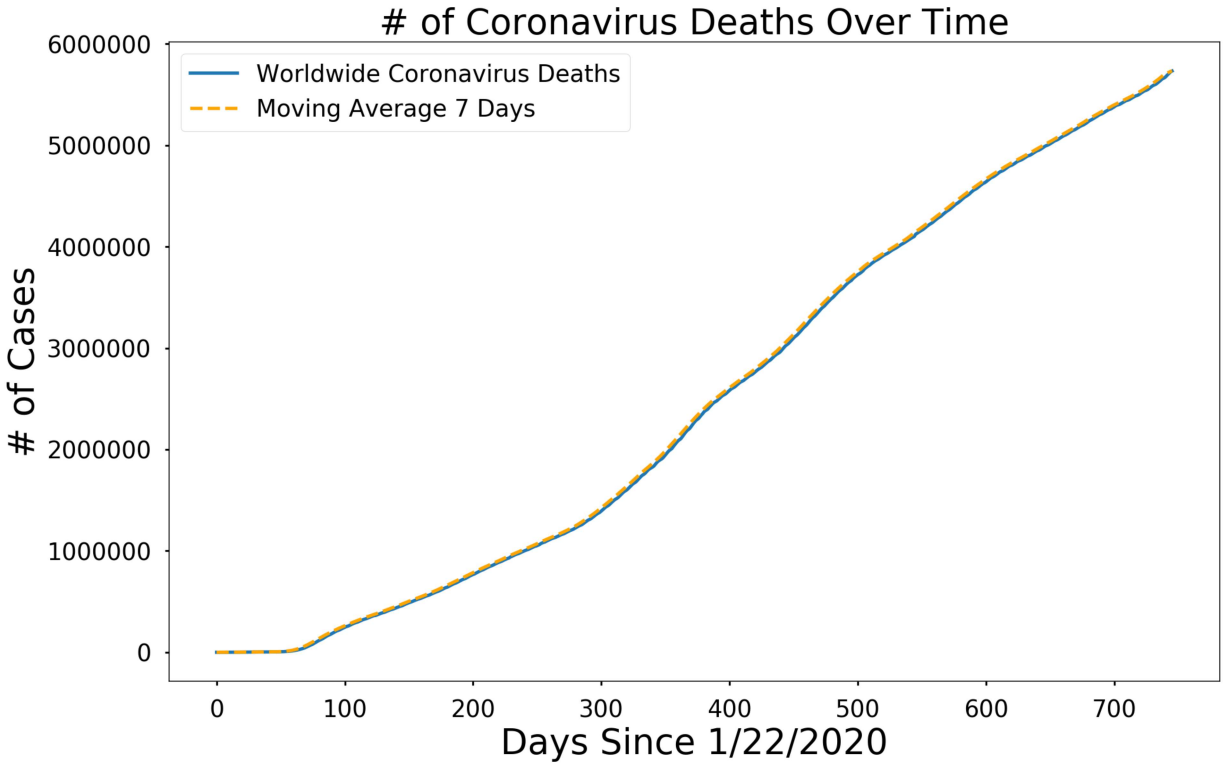
Out[23]:
```
<matplotlib.legend.Legend at 0x1ea73802dd8>
```



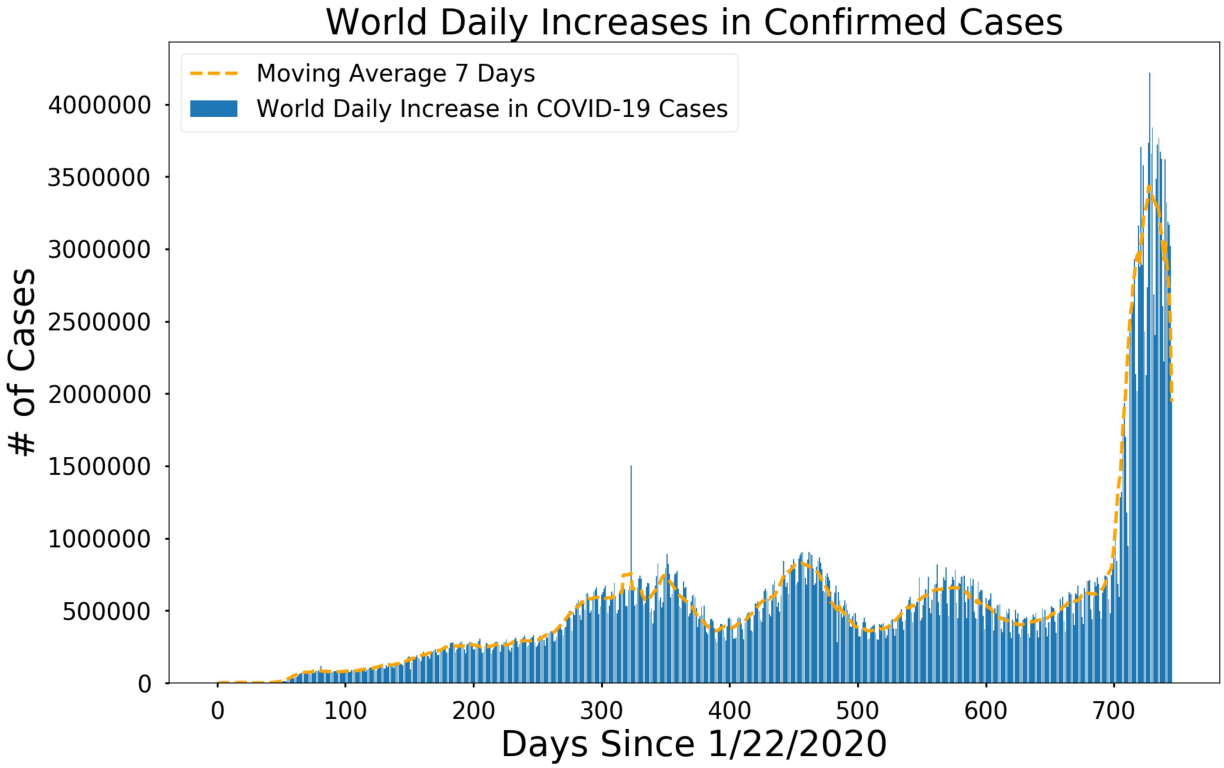
In [24]:
```
#worldwide overview
def flatten(arr):
    a = []
    arr = arr.tolist()
    for i in arr:
        a.append(i[0])
    return a
```

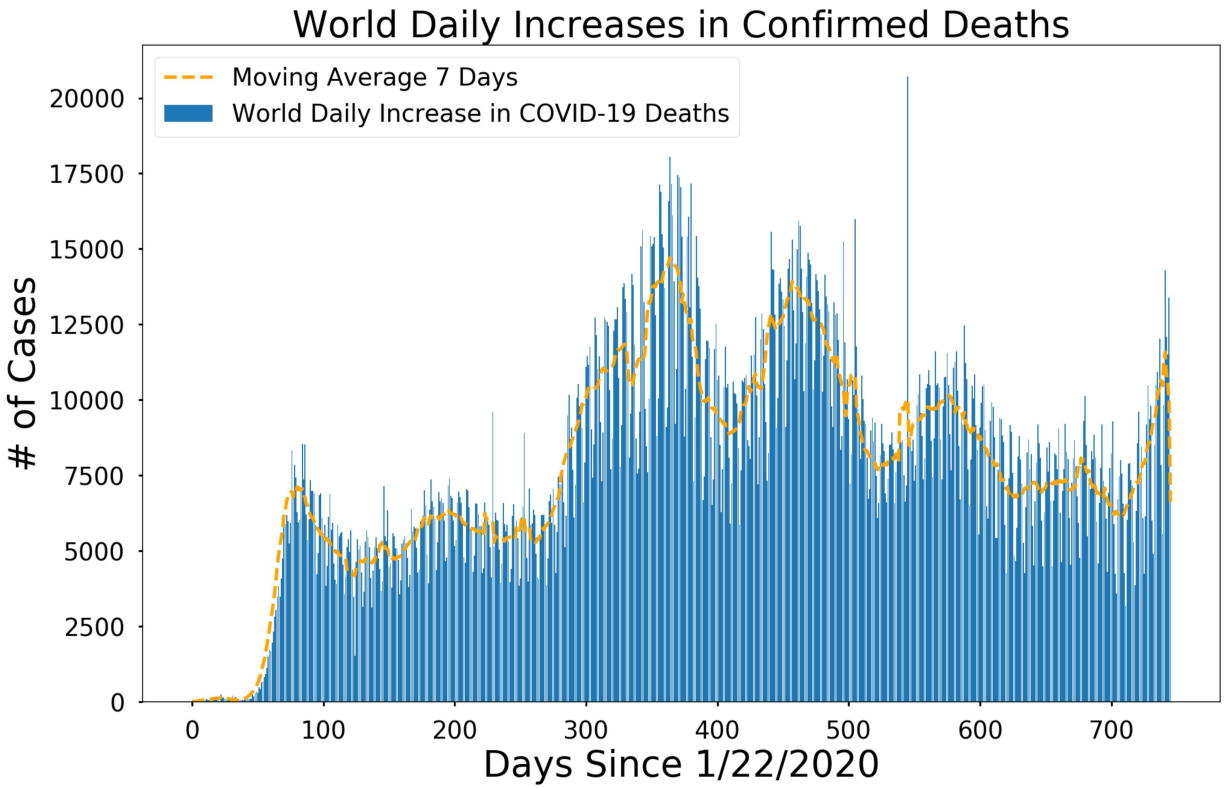
In [25]:
```
adjusted_dates = adjusted_dates.reshape(1, -1)[0]
plt.figure(figsize=(16, 10))
plt.plot(adjusted_dates, world_cases)
plt.plot(adjusted_dates, world_confirmed_avg, linestyle='dashed', color='orange')
plt.title('# of Coronavirus Cases Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['Worldwide Coronavirus Cases', 'Moving Average {} Days'.format(windov
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 10))
plt.plot(adjusted_dates, total_deaths)
plt.plot(adjusted_dates, world_death_avg, linestyle='dashed', color='orange')
plt.title('# of Coronavirus Deaths Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['Worldwide Coronavirus Deaths', 'Moving Average {} Days'.format(windo
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



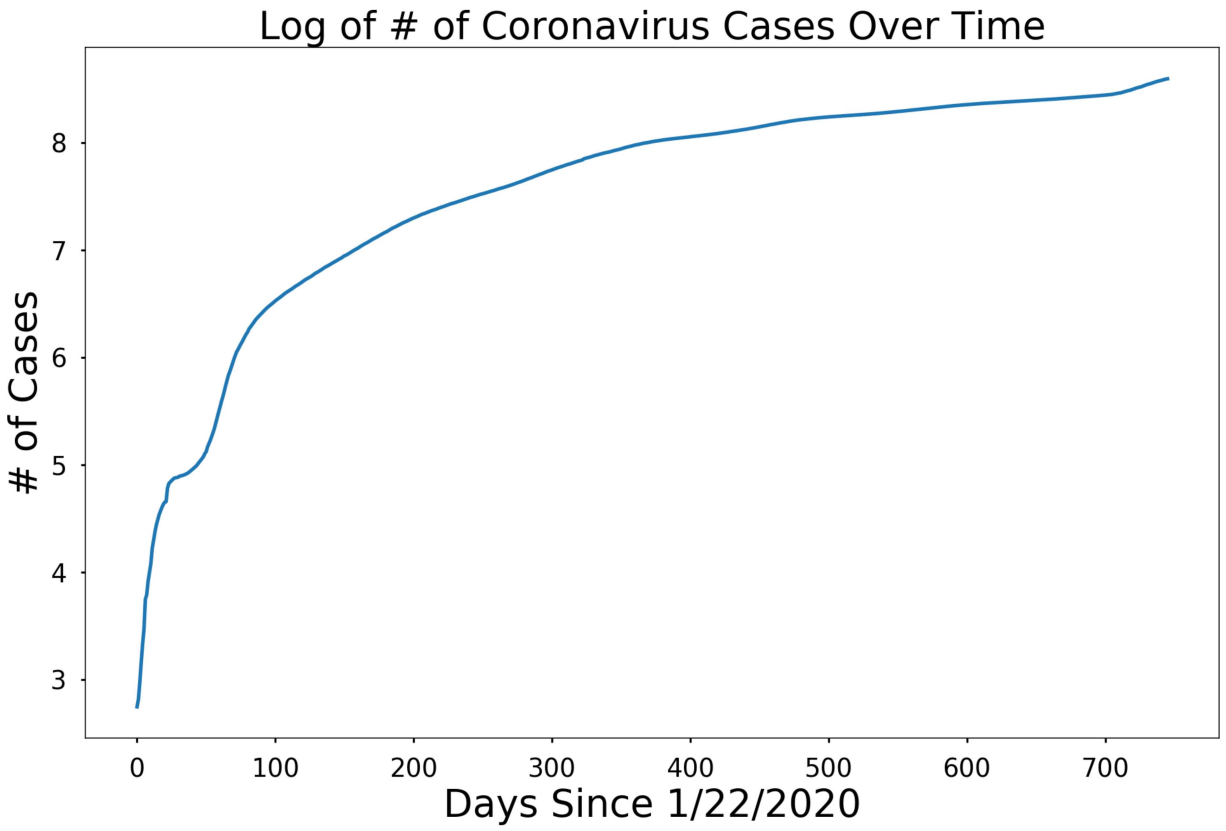

# of Coronavirus Deaths Over Time
Worldwide Coronavirus Deaths
Moving Average 7 Days
# of Cases
6000000
5000000
4000000
3000000
2000000
1000000
0
0
100
200
300
400
500
600
700
Days Since 1/22/2020

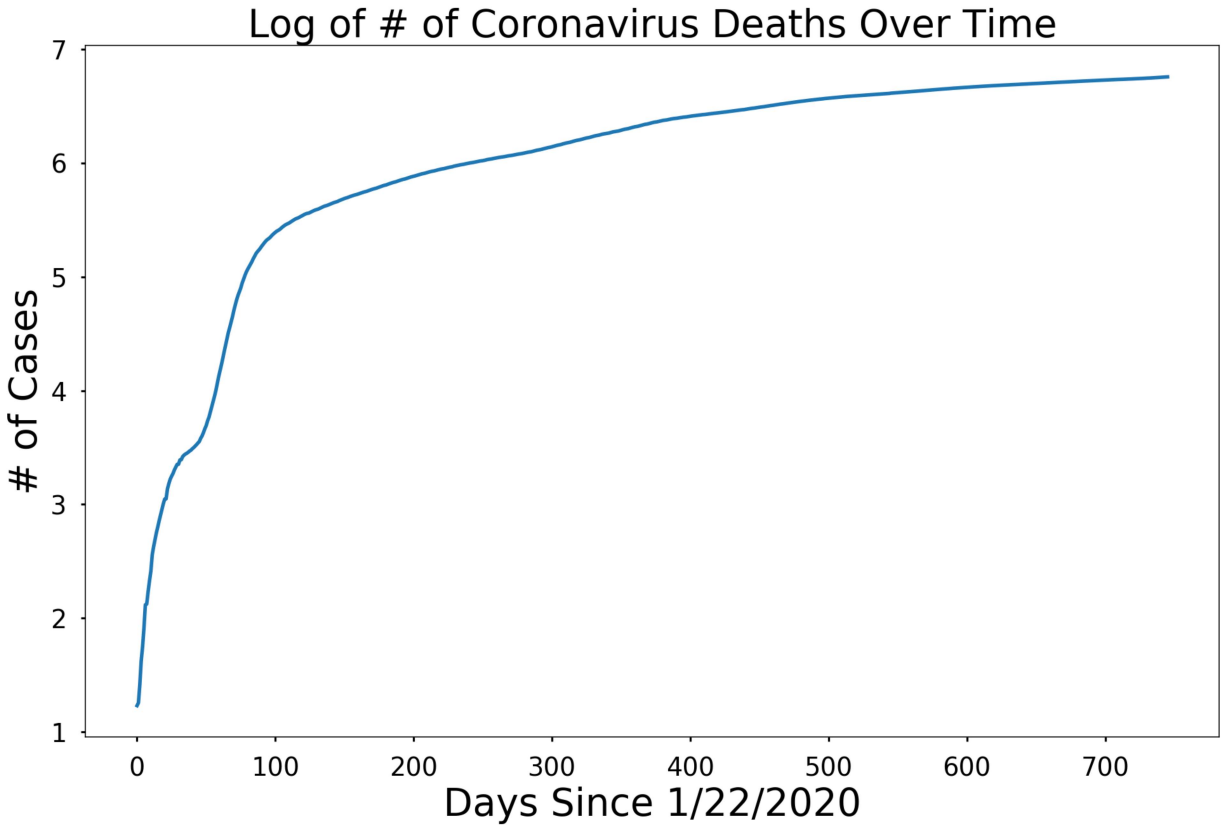
In [26]:
```
plt.figure(figsize=(16, 10))
plt.bar(adjusted_dates, world_daily_increase)
plt.plot(adjusted_dates, world_daily_increase_avg, color='orange', linestyle='das
plt.title('World Daily Increases in Confirmed Cases', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['Moving Average {} Days'.format(window), 'World Daily Increase in COV
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 10))
plt.bar(adjusted_dates, world_daily_death)
plt.plot(adjusted_dates, world_daily_death_avg, color='orange', linestyle='dashed
plt.title('World Daily Increases in Confirmed Deaths', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.legend(['Moving Average {} Days'.format(window), 'World Daily Increase in COV
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```




World Daily Increases in Confirmed Deaths
Moving Average 7 Days
World Daily Increase in COVID-19 Deaths
# of Cases
0
2500
5000
7500
10000
12500
15000
17500
20000
0
100
200
300
400
500
600
700
Days Since 1/22/2020

In [27]:
```
plt.figure(figsize=(16, 10))
plt.plot(adjusted_dates, np.log10(world_cases))
plt.title('Log of # of Coronavirus Cases Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()

plt.figure(figsize=(16, 10))
plt.plot(adjusted_dates, np.log10(total_deaths))
plt.title('Log of # of Coronavirus Deaths Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('# of Cases', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



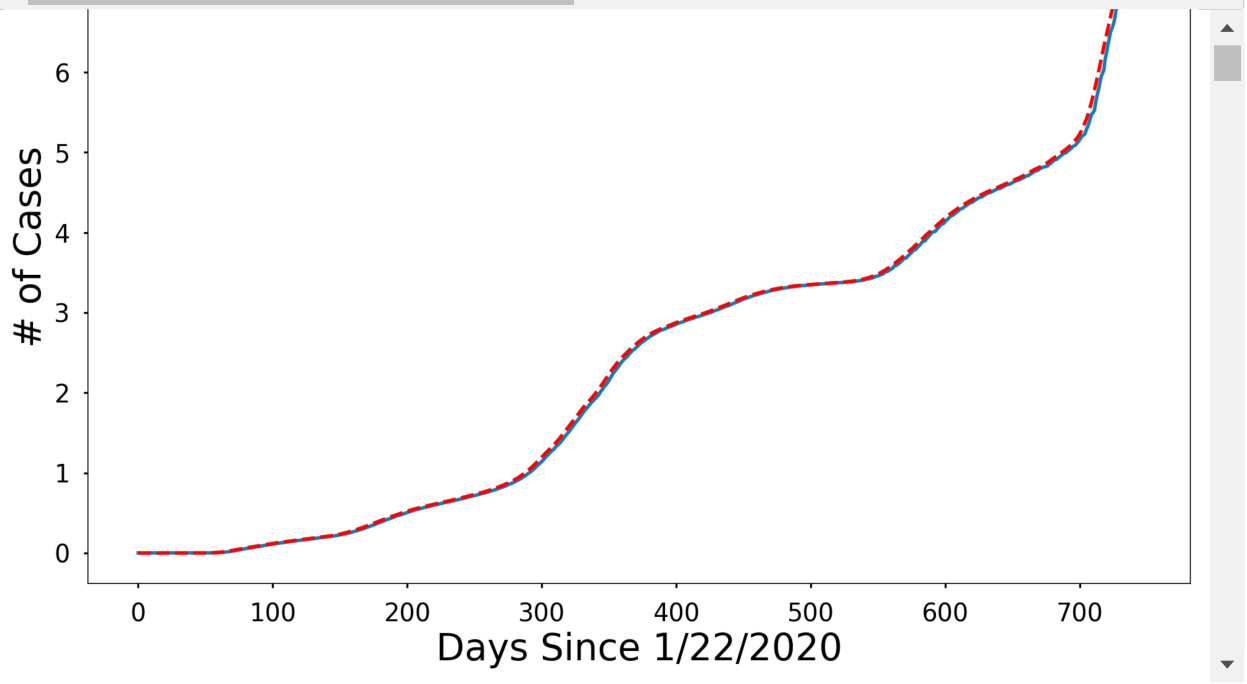

Log of # of Coronavirus Deaths Over Time
# of Cases
Days Since 1/22/2020
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700

In [31]:

```
def country_plot(x, y1, y2, y3, country):
    # window is set as 14 in in the beginning of the notebook
    confirmed_avg = moving_average(y1, window)
    confirmed_increase_avg = moving_average(y2, window)
    death_increase_avg = moving_average(y3, window)
#     recovery_increase_avg = moving_average(y4, window)

    plt.figure(figsize=(16, 10))
    plt.plot(x, y1)
    plt.plot(x, confirmed_avg, color='red', linestyle='dashed')
    plt.legend(['{} Confirmed Cases'.format(country), 'Moving Average {} Days'.fo
    plt.title('{} Confirmed Cases'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

    plt.figure(figsize=(16, 10))
    plt.bar(x, y2)
    plt.plot(x, confirmed_increase_avg, color='red', linestyle='dashed')
    plt.legend(['Moving Average {} Days'.format(window), '{} Daily Increase in Co
    plt.title('{} Daily Increases in Confirmed Cases'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

    plt.figure(figsize=(16, 10))
    plt.bar(x, y3)
    plt.plot(x, death_increase_avg, color='red', linestyle='dashed')
    plt.legend(['Moving Average {} Days'.format(window), '{} Daily Increase in Co
    plt.title('{} Daily Increases in Deaths'.format(country), size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()

def get_country_info(country_name):
    country_cases = []
    country_deaths = []
#     country_recoveries = []

    for i in dates:
        country_cases.append(confirmed_df[confirmed_df['Country/Region']==country
        country_deaths.append(deaths_df[deaths_df['Country/Region']==country_name
#         country_recoveries.append(recoveries_df[recoveries_df['Country/Region']
    return (country_cases, country_deaths)


def country_visualizations(country_name):
    country_info = get_country_info(country_name)
    country_cases = country_info[0]
    country_deaths = country_info[1]
```

```
    country_daily_increase = daily_increase(country_cases)
    country_daily_death = daily_increase(country_deaths)
#     country_daily_recovery = daily_increase(country_recoveries)

    country_plot(adjusted_dates, country_cases, country_daily_increase, country_d
```

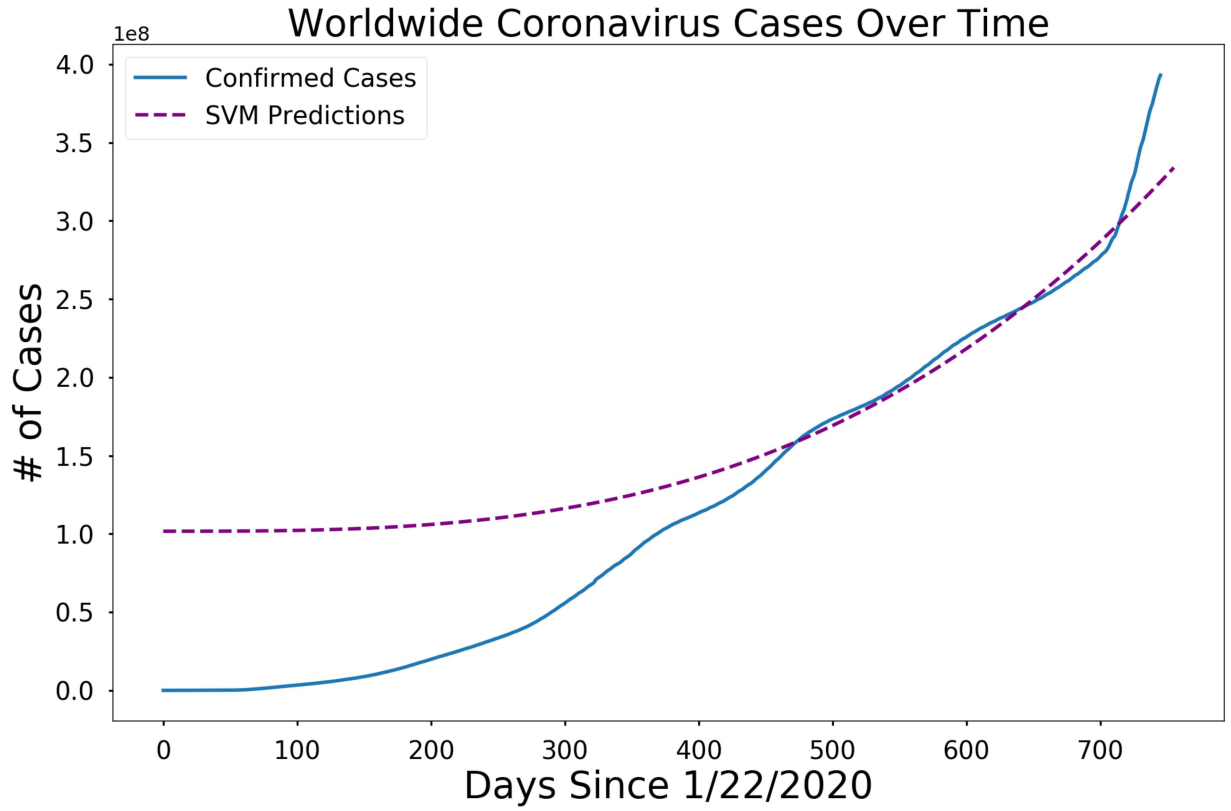
In [32]:
```
countries = ['US', 'Russia', 'India', 'Brazil', 'South Africa', 'China', 'Italy',
             'Germany', 'Spain', 'France', 'United Kingdom', 'Peru', 'Mexico', 'C
            'Pakistan', 'Turkey', 'Philippines', 'Iraq', 'Indonesia', 'Israel',
            'Romania', 'Morocco','Portugal', 'Austria', 'Sweden']

for country in countries:
    country_visualizations(country)
```



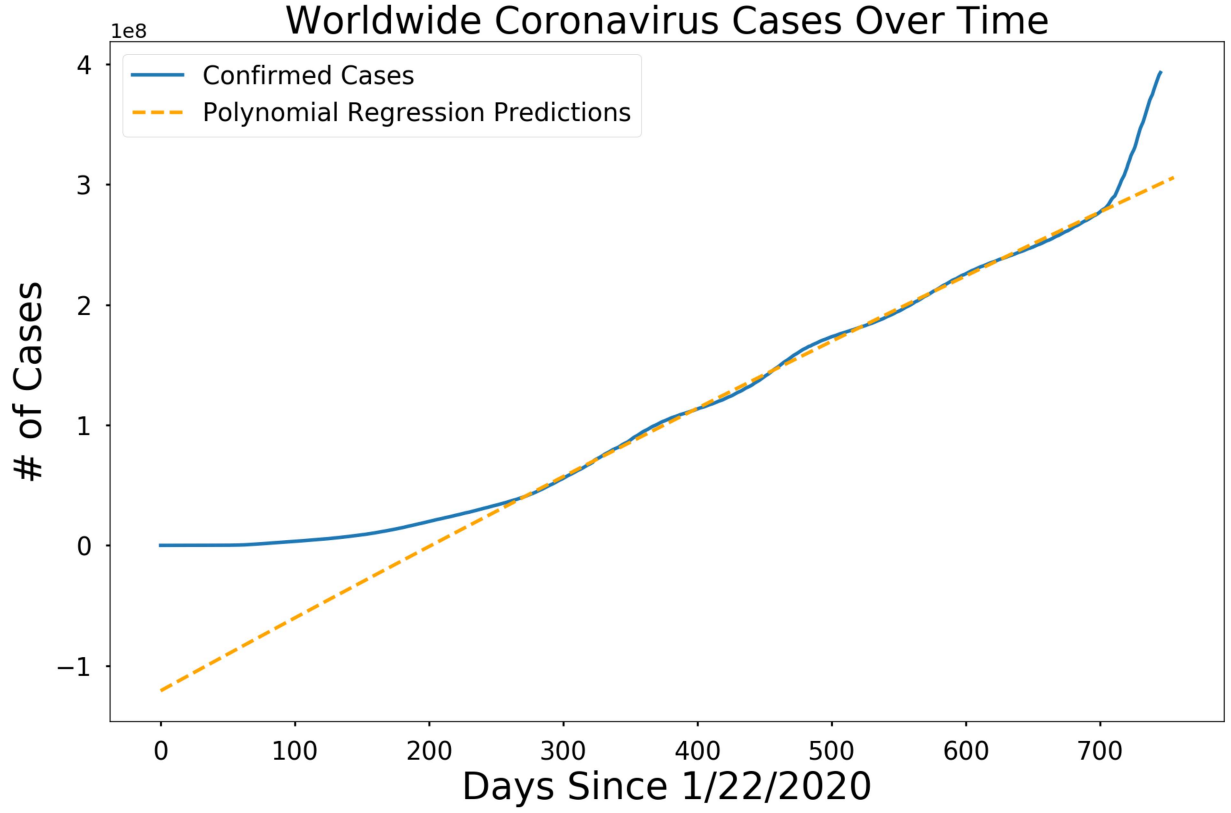
In [33]:
```
def plot_predictions(x, y, pred, algo_name, color):
    plt.figure(figsize=(16, 10))
    plt.plot(x, y)
    plt.plot(future_forcast, pred, linestyle='dashed', color=color)
    plt.title('Worldwide Coronavirus Cases Over Time', size=30)
    plt.xlabel('Days Since 1/22/2020', size=30)
    plt.ylabel('# of Cases', size=30)
    plt.legend(['Confirmed Cases', algo_name], prop={'size': 20})
    plt.xticks(size=20)
    plt.yticks(size=20)
    plt.show()
```

Predictions for confirmed coronovirus cases

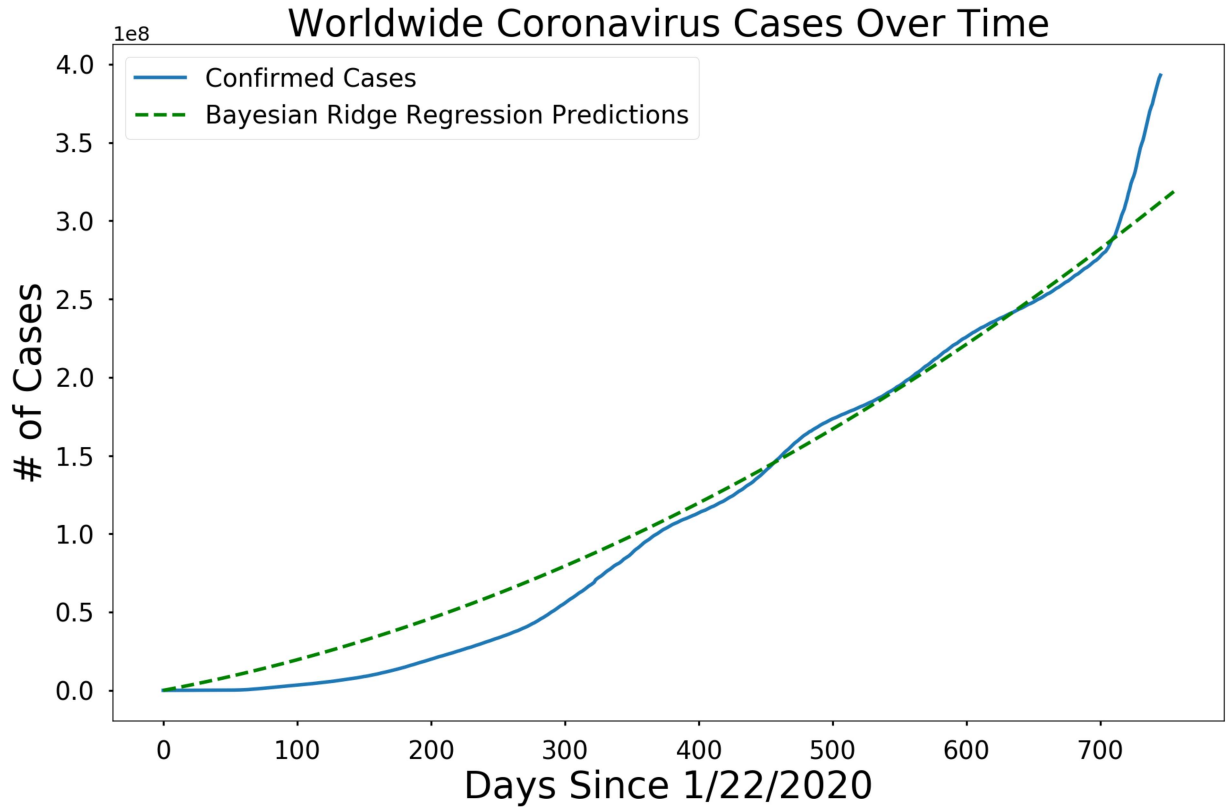
In [34]:
```
plot_predictions(adjusted_dates, world_cases, svm_pred, 'SVM Predictions', 'purpl
```



In [35]:
```
plot_predictions(adjusted_dates, world_cases, linear_pred, 'Polynomial Regressior
```



In [36]:
```
plot_predictions(adjusted_dates, world_cases, bayesian_pred, 'Bayesian Ridge Regr
```



In [37]:
```
# Future predictions using SVM
svm_df = pd.DataFrame({'Date': future_forcast_dates[-10:], 'SVM Predicted # of Co
svm_df.style.background_gradient(cmap='Reds')
```

Out[37]:

	Date	SVM Predicted # of Confirmed Cases Worldwide
0	02/06/2022	325852026.000000
1	02/07/2022	326754700.000000
2	02/08/2022	327659795.000000
3	02/09/2022	328567313.000000
4	02/10/2022	329477258.000000
5	02/11/2022	330389633.000000
6	02/12/2022	331304440.000000
7	02/13/2022	332221684.000000
8	02/14/2022	333141367.000000
9	02/15/2022	334063493.000000

In [38]:
```
# Future predictions using polynomial regression
linear_pred = linear_pred.reshape(1,-1)[0]
linear_df = pd.DataFrame({'Date': future_forcast_dates[-10:], 'Polynomial Predict
linear_df.style.background_gradient(cmap='Reds')
```

Out[38]:

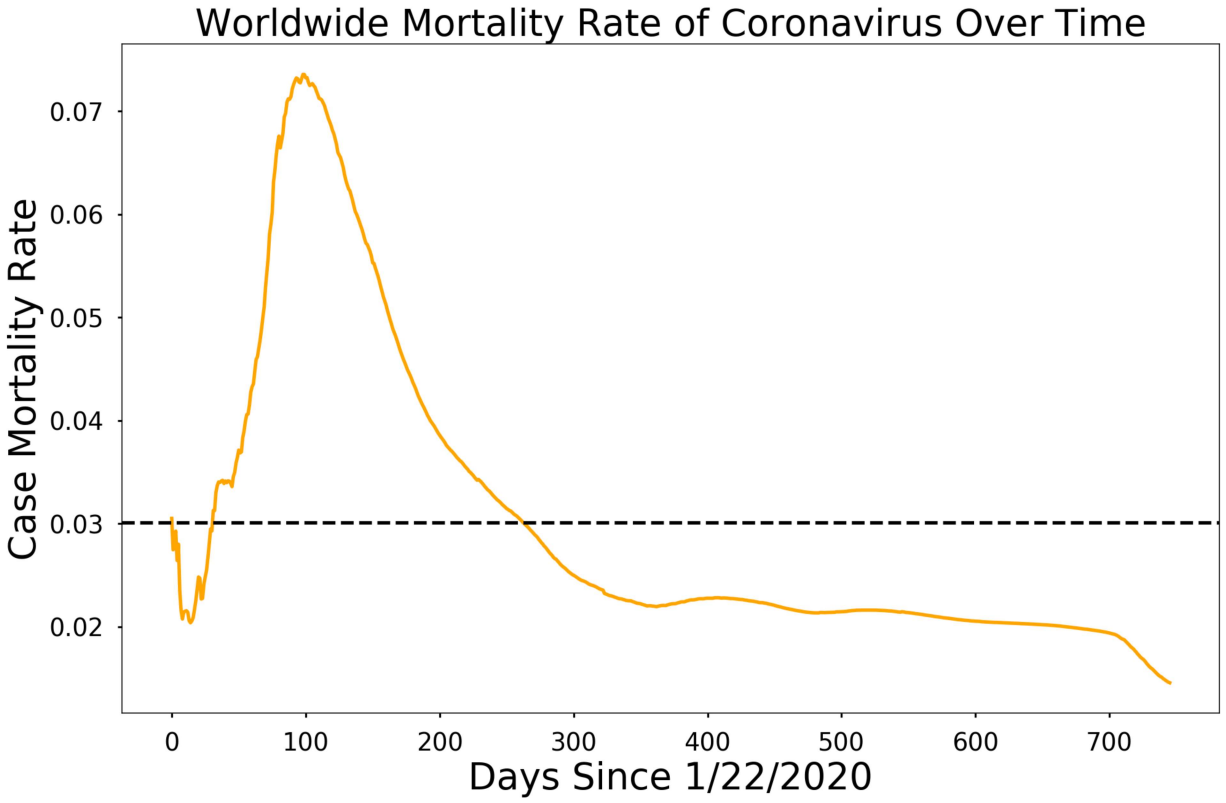
	Date	Polynomial Predicted # of Confirmed Cases Worldwide
0	02/06/2022	301211134.000000
1	02/07/2022	301730458.000000
2	02/08/2022	302249658.000000
3	02/09/2022	302768733.000000
4	02/10/2022	303287684.000000
5	02/11/2022	303806510.000000
6	02/12/2022	304325212.000000
7	02/13/2022	304843789.000000
8	02/14/2022	305362242.000000
9	02/15/2022	305880571.000000

In [39]:
```
# Future predictions using Bayesian Ridge
bayesian_df = pd.DataFrame({'Date': future_forcast_dates[-10:], 'Bayesian Ridge P
bayesian_df.style.background_gradient(cmap='Reds')
```

Out[39]:

	Date	Bayesian Ridge Predicted # of Confirmed Cases Worldwide
0	02/06/2022	312614636.000000
1	02/07/2022	313292205.000000
2	02/08/2022	313970467.000000
3	02/09/2022	314649420.000000
4	02/10/2022	315329065.000000
5	02/11/2022	316009403.000000
6	02/12/2022	316690433.000000
7	02/13/2022	317372154.000000
8	02/14/2022	318054568.000000
9	02/15/2022	318737674.000000

In [40]:
```
mean_mortality_rate = np.mean(mortality_rate)
plt.figure(figsize=(16, 10))
plt.plot(adjusted_dates, mortality_rate, color='orange')
plt.axhline(y = mean_mortality_rate,linestyle='--', color='black')
plt.title('Worldwide Mortality Rate of Coronavirus Over Time', size=30)
plt.xlabel('Days Since 1/22/2020', size=30)
plt.ylabel('Case Mortality Rate', size=30)
plt.xticks(size=20)
plt.yticks(size=20)
plt.show()
```



In [ ]: