In [1]:

```python
import pandas as pd
from openpyxl.workbook import Workbook

#reading excel file
df = pd.read_excel('C:/Users/svasudev/OneDrive - IGMFinancial/Documents/Predictive Analyti
                   engine='openpyxl',sheet_name=0,header=0,index_col=False,keep_default_na=
```

In [2]:

```python
#Top 5 rows of the Excel file
df.head()
```

Out[2]:

| | Date | Ottawa | Toronto West/Ouest | Toronto East/Est | Windsor | London | Peterborough | St. Catharine's | Sudbury | Sault Saint Marie | Thunder Bay | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1990-01-03 | 55.9 | 49.1 | 48.7 | 45.2 | 50.1 | 0.0 | 0.0 | 56.4 | 54.8 | 56.6 | |
| **1** | 1990-01-10 | 55.9 | 47.7 | 46.8 | 49.7 | 47.6 | 0.0 | 0.0 | 56.4 | 54.9 | 56.8 | |
| **2** | 1990-01-17 | 55.9 | 53.2 | 53.2 | 49.6 | 53.7 | 0.0 | 0.0 | 55.8 | 54.9 | 56.8 | |
| **3** | 1990-01-24 | 55.9 | 53.2 | 53.5 | 49.0 | 52.1 | 0.0 | 0.0 | 55.7 | 54.9 | 56.8 | |
| **4** | 1990-01-31 | 55.9 | 51.9 | 52.6 | 48.6 | 49.1 | 0.0 | 0.0 | 55.6 | 54.8 | 56.8 | |

In [3]:

```python
#Datatype and Null Information about the columns in the Excel file
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9962 entries, 0 to 9961
Data columns (total 20 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Date                                        9962 non-null   datetime64[ns]
 1   Ottawa                                      9962 non-null   float64
 2   Toronto West/Ouest                          9962 non-null   float64
 3   Toronto East/Est                            9962 non-null   float64
 4   Windsor                                     9962 non-null   float64
 5   London                                      9962 non-null   float64
 6   Peterborough                                9962 non-null   float64
 7   St. Catharine's                             9962 non-null   float64
 8   Sudbury                                     9962 non-null   float64
 9   Sault Saint Marie                           9962 non-null   float64
 10  Thunder Bay                                 9962 non-null   float64
 11  North Bay                                   9962 non-null   float64
 12  Timmins                                     9962 non-null   float64
 13  Kenora                                      9962 non-null   float64
 14  Parry Sound                                 9962 non-null   float64
 15  Ontario Average/Moyenne provinciale         9962 non-null   float64
 16  Southern Average/Moyenne du sud de l'Ontario  9962 non-null   float64
 17  Northern Average/Moyenne du nord de l'Ontario 9962 non-null   float64
```

```
 18  Fuel Type                                       9962 non-null   object
 19  Type de carburant                               9962 non-null   object
dtypes: datetime64[ns](1), float64(17), object(2)
memory usage: 1.5+ MB
```

In [4]:

```python
#Creating a new dataframe
df2 = pd.DataFrame()
```

In [5]:

```python
#Choosing rows from the original dataframe with Fuel Type "Regular Unleaded gasoline" and
df2 = df.loc[df['Fuel Type']=='Regular Unleaded Gasoline']
```

In [6]:

```python
#Datatype and Null Information about the columns in the new dataframe
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1675 entries, 0 to 1674
Data columns (total 20 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Date                                            1675 non-null   datetime64[ns]
 1   Ottawa                                          1675 non-null   float64
 2   Toronto West/Ouest                              1675 non-null   float64
 3   Toronto East/Est                                1675 non-null   float64
 4   Windsor                                         1675 non-null   float64
 5   London                                          1675 non-null   float64
 6   Peterborough                                    1675 non-null   float64
 7   St. Catharine's                                 1675 non-null   float64
 8   Sudbury                                         1675 non-null   float64
 9   Sault Saint Marie                               1675 non-null   float64
 10  Thunder Bay                                     1675 non-null   float64
 11  North Bay                                       1675 non-null   float64
 12  Timmins                                         1675 non-null   float64
 13  Kenora                                          1675 non-null   float64
 14  Parry Sound                                     1675 non-null   float64
 15  Ontario Average/Moyenne provinciale             1675 non-null   float64
 16  Southern Average/Moyenne du sud de l'Ontario    1675 non-null   float64
 17  Northern Average/Moyenne du nord de l'Ontario   1675 non-null   float64
 18  Fuel Type                                       1675 non-null   object
 19  Type de carburant                               1675 non-null   object
dtypes: datetime64[ns](1), float64(17), object(2)
memory usage: 274.8+ KB
```

In [7]:

```python
df2.head()
```

Out[7]:

| | Date | Ottawa | Toronto West/Ouest | Toronto East/Est | Windsor | London | Peterborough | St. Catharine's | Sudbury | Sault Saint Marie | Thunder Bay | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1990-01-03 | 55.9 | 49.1 | 48.7 | 45.2 | 50.1 | 0.0 | 0.0 | 56.4 | 54.8 | 56.6 | |
| **1** | 1990-01-10 | 55.9 | 47.7 | 46.8 | 49.7 | 47.6 | 0.0 | 0.0 | 56.4 | 54.9 | 56.8 | |
| **2** | 1990-01-17 | 55.9 | 53.2 | 53.2 | 49.6 | 53.7 | 0.0 | 0.0 | 55.8 | 54.9 | 56.8 | |

| | Date | Ottawa | Toronto West/Ouest | Toronto East/Est | Windsor | London | Peterborough | St. Catharine's | Sudbury | Sault Saint Marie | Thunder Bay | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 1990-01-24 | 55.9 | 53.2 | 53.5 | 49.0 | 52.1 | 0.0 | 0.0 | 55.7 | 54.9 | 56.8 | |
| **4** | 1990-01-31 | 55.9 | 51.9 | 52.6 | 48.6 | 49.1 | 0.0 | 0.0 | 55.6 | 54.8 | 56.8 | |

In [8]:

```
#Creating a new dataframe
df3 = pd.DataFrame()
```

In [9]:

```
#Analyzing 'Regular Unleaded Gasoline' price for Toronto West/Ouest Region only and pastir
df3.insert(0,"Date",df2['Date'],True)
df3.insert(1,"Toronto West/Ouest",df2['Toronto West/Ouest'],True)
```

In [10]:

```
df3.head()
```

Out[10]:

| | Date | Toronto West/Ouest |
|---|---|---|
| **0** | 1990-01-03 | 49.1 |
| **1** | 1990-01-10 | 47.7 |
| **2** | 1990-01-17 | 53.2 |
| **3** | 1990-01-24 | 53.2 |
| **4** | 1990-01-31 | 51.9 |

In [11]:

```
#Date column only had weekly values. Sometimes the week started on Wednesday and sometimes
#this inconistency, converting the date column into daily values and assigning the previou
#daily values.
df3.set_index('Date', inplace=True)
df3 = df3.resample('D').ffill().reset_index()
```

In [12]:

```
#Last 10 values from the dataset
df3.tail(10)
```

Out[12]:

| | Date | Toronto West/Ouest |
|---|---|---|
| **11707** | 2022-01-22 | 146.3 |
| **11708** | 2022-01-23 | 146.3 |
| **11709** | 2022-01-24 | 148.0 |
| **11710** | 2022-01-25 | 148.0 |
| **11711** | 2022-01-26 | 148.0 |
| **11712** | 2022-01-27 | 148.0 |
| **11713** | 2022-01-28 | 148.0 |
| **11714** | 2022-01-29 | 148.0 |

| | Date | Toronto West/Ouest |
|---|---|---|
| **11715** | 2022-01-30 | 148.0 |
| **11716** | 2022-01-31 | 153.1 |

In [13]:

```
#Checking for null values
df3.isnull().values.any()
```

Out[13]:

```
False
```

In [14]:

```
#Splitting the "Date" column into year, month and week to explore trends
df3['Year']=df3['Date'].dt.year
df3['Month']=df3['Date'].dt.month
df3['Week']=df3['Date'].dt.isocalendar().week
```

In [15]:

```
df3.head(10)
```

Out[15]:

| | Date | Toronto West/Ouest | Year | Month | Week |
|---|---|---|---|---|---|
| **0** | 1990-01-03 | 49.1 | 1990 | 1 | 1 |
| **1** | 1990-01-04 | 49.1 | 1990 | 1 | 1 |
| **2** | 1990-01-05 | 49.1 | 1990 | 1 | 1 |
| **3** | 1990-01-06 | 49.1 | 1990 | 1 | 1 |
| **4** | 1990-01-07 | 49.1 | 1990 | 1 | 1 |
| **5** | 1990-01-08 | 49.1 | 1990 | 1 | 2 |
| **6** | 1990-01-09 | 49.1 | 1990 | 1 | 2 |
| **7** | 1990-01-10 | 47.7 | 1990 | 1 | 2 |
| **8** | 1990-01-11 | 47.7 | 1990 | 1 | 2 |
| **9** | 1990-01-12 | 47.7 | 1990 | 1 | 2 |

In [16]:

```
#Splitting the dataset in Train and Test
#Train from Year 1990 to Year 2019
#Test from Year 2020

train = df3[(df3['Date'] > '1990-01-01') & (df3['Date'] <= '2019-12-31')]
test = df3[df3['Date'] >= '2020-01-01']
```

## Yearly Price Visualization on Train and Test Dataset

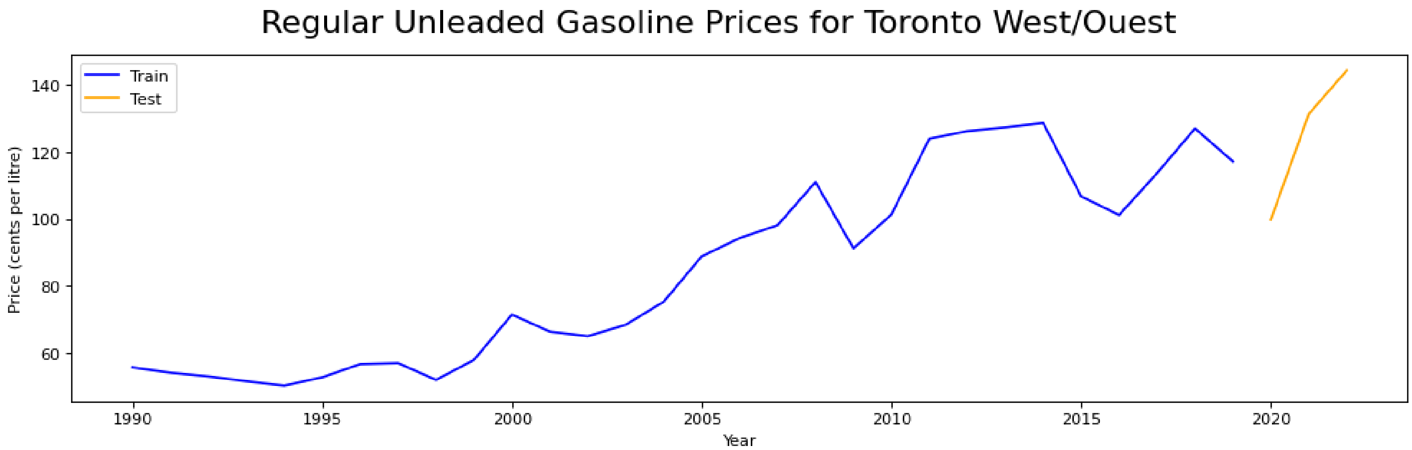
In [17]:

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

yearly_train_Price = train.groupby(['Year'])['Toronto West/Ouest'].mean()
yearly_test_Price = test.groupby(['Year'])['Toronto West/Ouest'].mean()

figure(figsize=(15, 4), dpi=80)
plt.plot(yearly_train_Price, label='Train',c='blue')
plt.plot(yearly_test_Price, label='Test',c='orange')
plt.legend(loc='best')
```

```
plt.suptitle('Regular Unleaded Gasoline Prices for Toronto West/Ouest', fontsize=20)
plt.xlabel('Year')
plt.ylabel('Price (cents per litre)')
plt.show()
```




## DataPrep for Time Series

In [18]:

```
train.index = pd.DatetimeIndex(train['Date'])
#Changing the frequency of the index to Daily
train.index = train.asfreq('d').index

test.index = pd.DatetimeIndex(test['Date'])
#Changing the frequency of the index to Daily
test.index = test.asfreq('d').index
```

## Train and Time Series Dataset

In [19]:

```
train_time_series = pd.DataFrame()
train_time_series.index = train.index
train_time_series.insert(0,"Toronto West/Ouest Gas Price Train",train['Toronto West/Ouest'

test_time_series = pd.DataFrame()
test_time_series.index = test.index
test_time_series.insert(0,"Toronto West/Ouest Gas Price Test",test['Toronto West/Ouest'],T


print(train_time_series.tail())
print(test_time_series.tail())
```

```
            Toronto West/Ouest Gas Price Train
Date
2019-12-27                               117.8
2019-12-28                               117.8
2019-12-29                               117.8
2019-12-30                               119.3
2019-12-31                               119.3
            Toronto West/Ouest Gas Price Test
Date
2022-01-27                              148.0
2022-01-28                              148.0
2022-01-29                              148.0
2022-01-30                              148.0
2022-01-31                              153.1
```

## ARIMA Model

To predict time series with ARIMA, we need to set the values of three parameters (p,d,q):

p: The order of the auto-regressive (AR) model (i.e., the number of lag observations)

d: The degree of differencing.

q: The order of the moving average (MA) model.

### Checking if data is stationary - We can see that it is not based on the P-value - Augmented Dickey Fuller Test

In [20]:

```
from statsmodels.tsa.stattools import adfuller
results = adfuller(train_time_series['Toronto West/Ouest Gas Price Train'])
print('ADF Statistic: ',results[0])
print('p-value: ',results[1])
print('Critical Values', results[4])
```

```
ADF Statistic:  -1.8963580272097427
p-value:  0.33377634463682804
Critical Values {'1%': -3.4309487036285113, '5%': -2.8618045938569296, '10%': -2.566910838
1800188}
```

### Taking First difference - P value is < 0.05. We can stop at the First Difference; d = 1

In [21]:

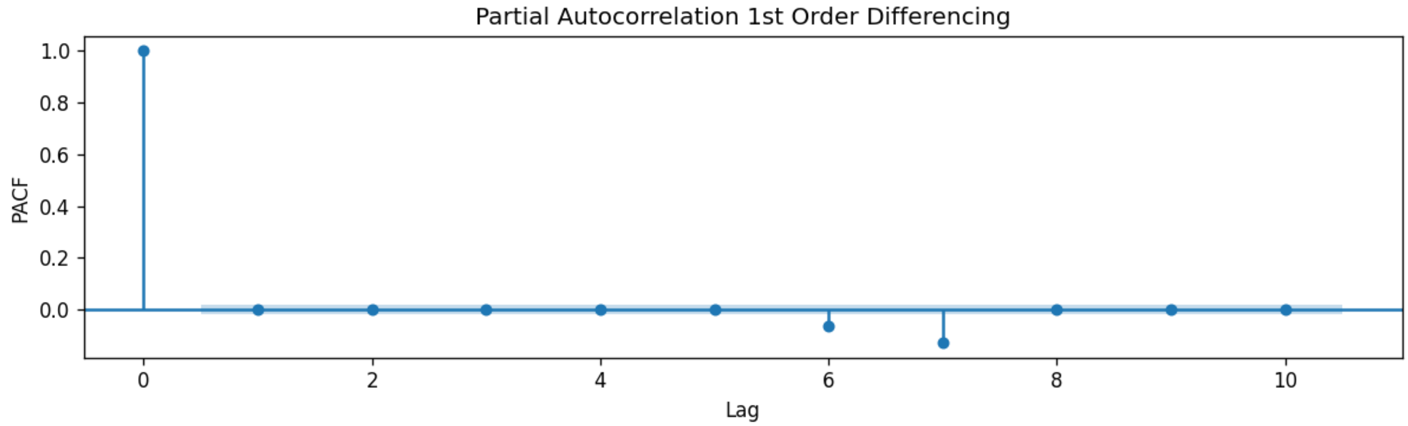
```
train_time_series_stationary1 = train_time_series.diff().dropna()
results1 = adfuller(train_time_series_stationary1['Toronto West/Ouest Gas Price Train'])
print('ADF Statistic: ',results1[0])
print('p-value: ',results1[1])
print('Critical Values', results1[4])
```

```
ADF Statistic:  -17.955762110833664
p-value:  2.8282691844425224e-30
Critical Values {'1%': -3.4309487036285113, '5%': -2.8618045938569296, '10%': -2.566910838
1800188}
```

### The Order of Autoregressive Term p; p = 0

In [22]:

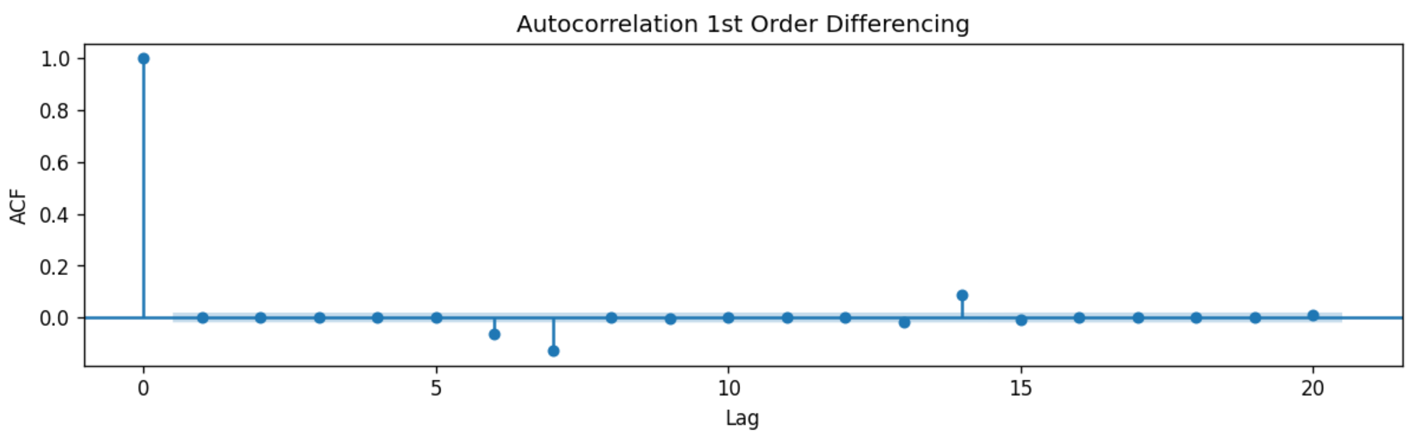
```
plt.rcParams.update({'figure.figsize':(12,3), 'figure.dpi':120})
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(train_time_series_stationary1, lags=10,  title="Partial Autocorrelation 1st Orde
plt.xlabel('Lag')
plt.ylabel('PACF')
plt.show()
```




### The order of the Moving Average term q; q = 0

In [23]:

```
plt.rcParams.update({'figure.figsize':(12,3), 'figure.dpi':120})
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(train_time_series_stationary1, lags=20,  title="Autocorrelation 1st Order Differe
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.show()
```




### Check the above p,d,q parameters with Auto Arima - Best Model ARIMA (0,1,0)

In [24]:

```
import pmdarima as pm
from pmdarima.model_selection import train_test_split
import numpy as np

model1 = pm.auto_arima(train_time_series, trace=True, error_action='ignore', suppress_warn
model1.fit(train_time_series)

forecast1 = model1.predict(n_periods=len(test_time_series))
forecast1 = pd.DataFrame(forecast1,index = test_time_series.index,columns=['Prediction'])
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=33542.140, Time=1.45 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=33534.140, Time=0.22 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=33536.140, Time=0.64 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=33536.140, Time=0.76 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=33532.500, Time=0.13 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=33538.140, Time=0.99 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 4.225 seconds
```

## Model Summary

In [25]:
```
import statsmodels.api as sm
model = sm.tsa.arima.ARIMA(train_time_series, order=(0,1,0))
model_result = model.fit()
print(model_result.summary())
```

```
                                   SARIMAX Results

==========================================================================================
====
Dep. Variable:     Toronto West/Ouest Gas Price Train   No. Observations:                 1
0955
Model:                                 ARIMA(0, 1, 0)   Log Likelihood                -1676
5.250
Date:                                Sun, 06 Mar 2022   AIC                            3353
2.500
Time:                                        17:17:09   BIC                            3353
9.801
Sample:                                    01-03-1990   HQIC                           3353
4.960
                                         - 12-31-2019

Covariance Type:                                  opg

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         1.2501      0.002    517.593      0.000       1.245       1.255
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            4192412.30
Prob(Q):                              1.00   Prob(JB):                          0.00
Heteroskedasticity (H):               1.58   Skew:                              1.02
Prob(H) (two-sided):                  0.00   Kurtosis:                         98.82
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

## Model Prediction

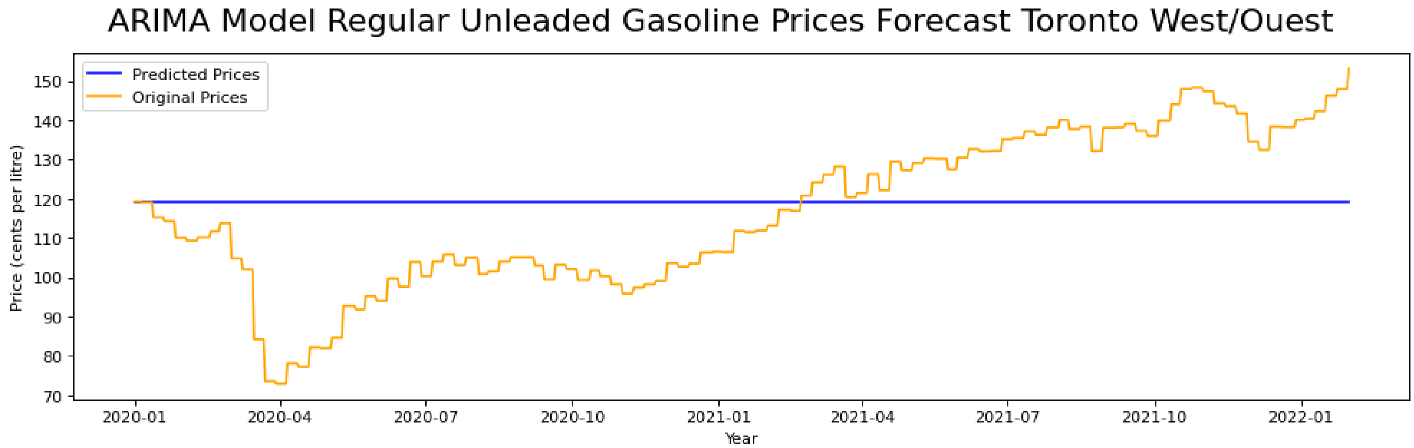
In [26]:
```
import warnings
warnings.filterwarnings('ignore')
ARIMA_Predict = model_result.predict(start='1/1/2020', end='1/31/2022')
ARIMA_Predict_df = pd.DataFrame(ARIMA_Predict)
```

In [27]:
```
ARIMA_Predict_df.tail()
```

Out[27]:

| | predicted_mean |
|---|---|
| **2022-01-27** | 119.3 |
| **2022-01-28** | 119.3 |
| **2022-01-29** | 119.3 |
| **2022-01-30** | 119.3 |
| **2022-01-31** | 119.3 |

In [28]:
```
figure(figsize=(15, 4), dpi=80)
plt.plot(ARIMA_Predict_df, label='Predicted Prices',c='blue')
plt.plot(test_time_series, label='Original Prices',c='orange')
plt.legend(loc='best')
plt.suptitle('ARIMA Model Regular Unleaded Gasoline Prices Forecast Toronto West/Ouest', f
plt.xlabel('Year')
plt.ylabel('Price (cents per litre)')
plt.show()
```




### Evaluation of the Model

### Mean Absolute Error (MAE) ARIMA

In [29]:
```
from sklearn.metrics import mean_absolute_error
maeARIMA=mean_absolute_error(test_time_series['Toronto West/Ouest Gas Price Test'],ARIMA_F
print('Mean Absolute Error ARIMA = {}'.format(round(maeARIMA, 2)))
```

```
Mean Absolute Error ARIMA = 17.1
```

### Mean squared error (MSE) ARIMA

In [30]:
```
from sklearn.metrics import mean_squared_error
mseARIMA=mean_squared_error(test_time_series['Toronto West/Ouest Gas Price Test'],ARIMA_Pr
print('The Mean Squared Error ARIMA = {}'.format(round(mseARIMA, 2)))
```

```
The Mean Squared Error ARIMA = 381.97
```

### Root mean squared error (RMSE) ARIMA

In [31]:
```
from numpy import sqrt
rmseARIMA = sqrt(mseARIMA)
print('The Root Mean Squared Error ARIMA = {}'.format(round(rmseARIMA, 2)))
```

```
The Root Mean Squared Error ARIMA = 19.54
```

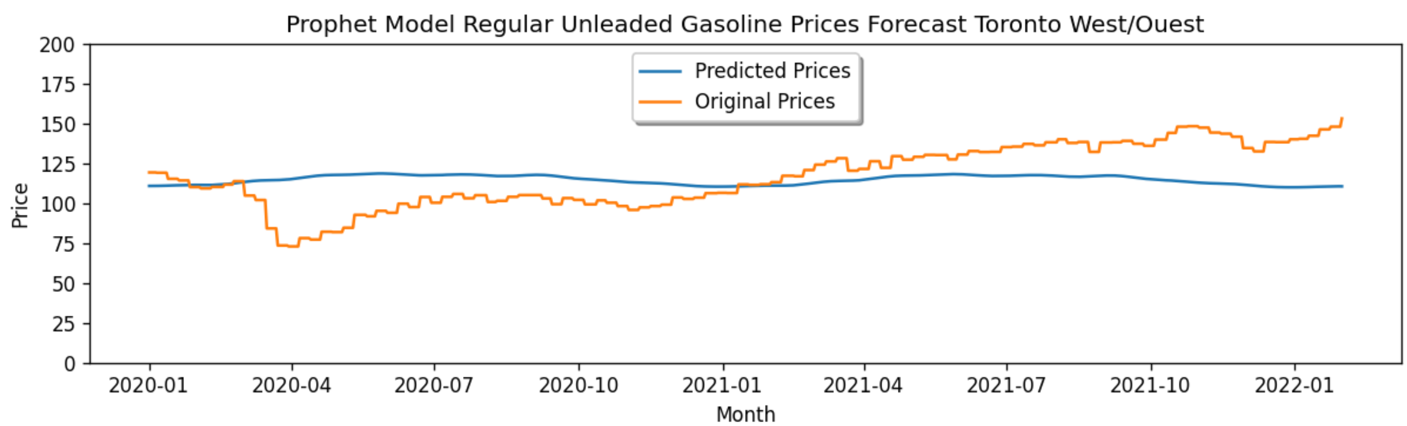
### Prophet Model

In [32]:
```
from fbprophet import Prophet
d={'ds':train['Date'],'y':train['Toronto West/Ouest']}
df_pred=pd.DataFrame(data=d)
```

```
model_prophet = Prophet(daily_seasonality=False)
model_prophet_result = model_prophet.fit(df_pred)
```

In [33]:
```
future = model_prophet.make_future_dataframe(periods=765)
forecast = model_prophet.predict(future)
forecast = forecast[(forecast['ds' ] >= '2020-01-01') & (forecast['ds' ] <= '2022-01-31')]
```

In [34]:
```
fig, ax = plt.subplots()
ax.plot(forecast['ds'], forecast['yhat'], label='Predicted Prices')
ax.plot(test['Date'], test['Toronto West/Ouest'], label='Original Prices')
plt.ylim([0,200])
legend = ax.legend(loc='upper center', shadow=True)
plt.title('Prophet Model Regular Unleaded Gasoline Prices Forecast Toronto West/Ouest')
plt.xlabel('Month')
plt.ylabel('Price')
plt.show()
```




## Mean Absolute Error (MAE) Prophet

In [35]:
```
maeProphet=mean_absolute_error(test['Toronto West/Ouest'],forecast['yhat'])
print('Mean Absolute Error Prophet = {}'.format(round(maeProphet, 2)))
```

```
Mean Absolute Error Prophet = 17.22
```

## Mean squared error (MSE) Prophet

In [36]:
```
mseProphet = mean_squared_error(test['Toronto West/Ouest'],forecast['yhat'])
print('The Mean Squared Error Prophet = {}'.format(round(mseProphet, 2)))
```

```
The Mean Squared Error Prophet = 407.49
```

## Root mean squared error (RMSE) Prophet

In [37]:
```
rmseProphet = sqrt(mseProphet)
print('The Root Mean Squared Error Prophet = {}'.format(round(rmseProphet, 2)))
```

```
The Root Mean Squared Error Prophet = 20.19
```