In [2]:
```
import numpy as np
import pandas as pd
df = pd.read_csv("HCLTECH.csv")
df
```

Out[2]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliv
0	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	1192200	1.886915e+14	NaN	NaN	
1	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	344850	5.714349e+13	NaN	NaN	
2	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	53000	9.564880e+12	NaN	NaN	
3	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	270950	5.256169e+13	NaN	NaN	
4	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	428800	8.535473e+13	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5193	2020-11-23	HCLTECH	EQ	819.25	825.0	842.00	816.25	838.50	839.20	832.35	7116516	5.923459e+14	139507.0	2607441.0	0
5194	2020-11-24	HCLTECH	EQ	839.20	843.9	857.40	835.35	841.00	840.50	847.95	8465615	7.178406e+14	169850.0	2882146.0	0
5195	2020-11-25	HCLTECH	EQ	840.50	840.5	846.00	822.50	825.00	824.70	829.08	5610232	4.651325e+14	124023.0	2224611.0	0
5196	2020-11-26	HCLTECH	EQ	824.70	824.1	845.00	819.60	841.20	842.05	834.43	8414555	7.021383e+14	138751.0	2752455.0	0
5197	2020-11-27	HCLTECH	EQ	842.05	842.0	847.80	814.35	823.15	822.10	827.29	11723771	9.698927e+14	154427.0	6387431.0	0

5198 rows × 15 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Date                    0
Symbol                  0
Series                  0
Prev Close              0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
VWAP                    0
Volume                  0
Turnover                0
Trades               2844
Deliverable Volume    503
%Deliverble           503
dtype: int64
```

In [20]:
```
import lightgbm as lgb

from matplotlib import pyplot as plt
from pmdarima import auto_arima
from sklearn.metrics import mean_absolute_error, mean_squared_error

myfavouritenumber = 13
seed = myfavouritenumber
np.random.seed(seed)
```

In [5]:
```
df.set_index("Date", drop=False, inplace=True)
df.head()
```

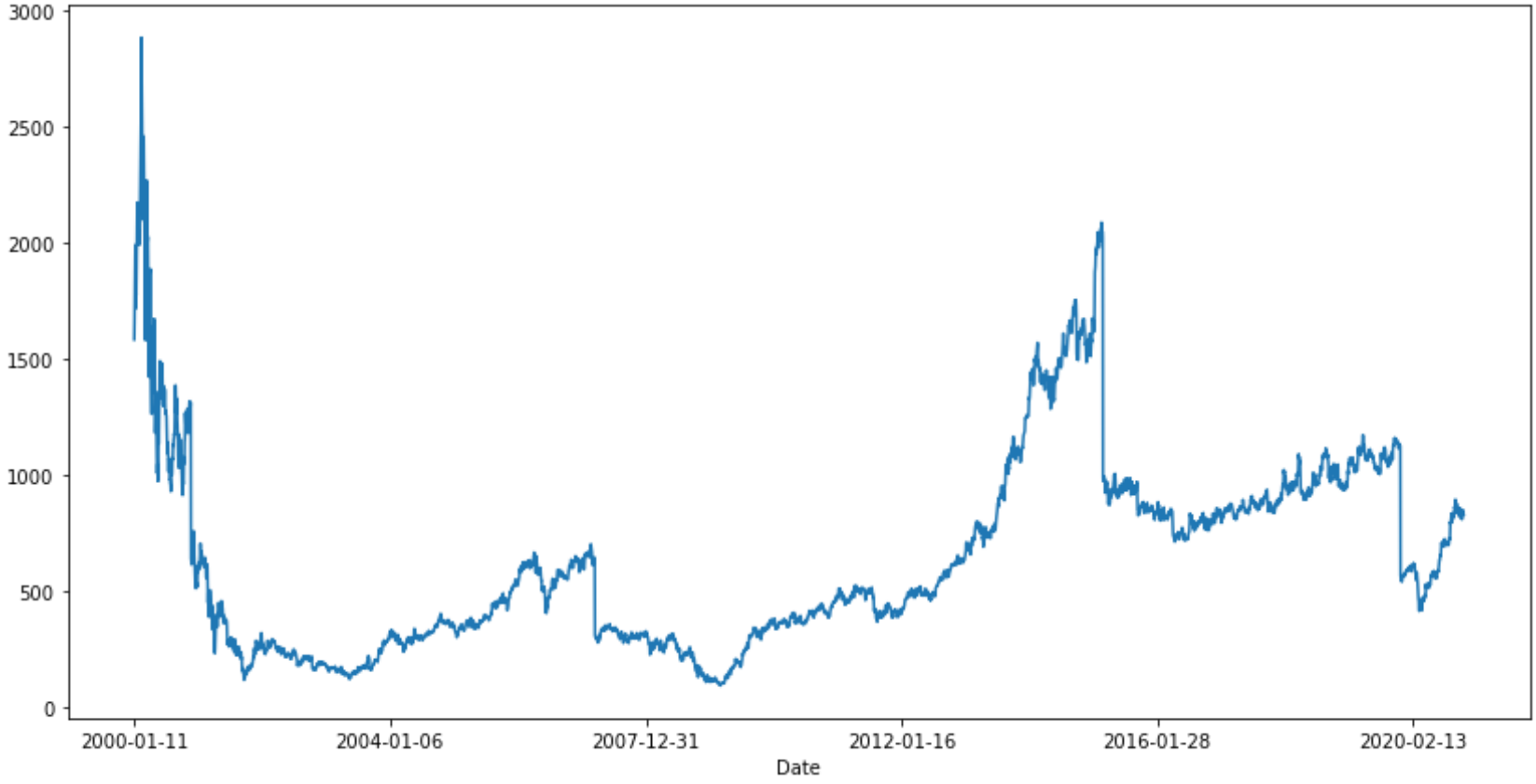
Out[5]:

Date	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverl
2000-01-11	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	1192200	1.886915e+14	NaN	NaN	N
2000-01-12	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	344850	5.714349e+13	NaN	NaN	N
2000-01-13	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	53000	9.564880e+12	NaN	NaN	N

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverl
Date															
2000-01-14	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	270950	5.256169e+13	NaN	NaN	N
2000-01-17	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	428800	8.535473e+13	NaN	NaN	N

In [6]:
```
df.VWAP.plot(figsize=(14, 7))
```

Out[6]: <AxesSubplot:xlabel='Date'>



In [8]:

```
df.reset_index(drop=True, inplace=True)
lag_features = ["High", "Low", "Volume", "Turnover", "Trades"]
window1 = 3
window2 = 7
window3 = 30

df_rolled_3d = df[lag_features].rolling(window=window1, min_periods=0)
df_rolled_7d = df[lag_features].rolling(window=window2, min_periods=0)
df_rolled_30d = df[lag_features].rolling(window=window3, min_periods=0)

df_mean_3d = df_rolled_3d.mean().shift(1).reset_index().astype(np.float32)
df_mean_7d = df_rolled_7d.mean().shift(1).reset_index().astype(np.float32)
df_mean_30d = df_rolled_30d.mean().shift(1).reset_index().astype(np.float32)

df_std_3d = df_rolled_3d.std().shift(1).reset_index().astype(np.float32)
df_std_7d = df_rolled_7d.std().shift(1).reset_index().astype(np.float32)
df_std_30d = df_rolled_30d.std().shift(1).reset_index().astype(np.float32)

for feature in lag_features:
    df[f"{feature}_mean_lag{window1}"] = df_mean_3d[feature]
    df[f"{feature}_mean_lag{window2}"] = df_mean_7d[feature]
    df[f"{feature}_mean_lag{window3}"] = df_mean_30d[feature]

    df[f"{feature}_std_lag{window1}"] = df_std_3d[feature]
    df[f"{feature}_std_lag{window2}"] = df_std_7d[feature]
    df[f"{feature}_std_lag{window3}"] = df_std_30d[feature]

df.fillna(df.mean(), inplace=True)

df.set_index("Date", drop=False, inplace=True)
df.head()
```

Out[8]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	...	Turnover_mean_lag30	Turnover_std_lag3	Turnover
Date														
2000-01-11	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	...	9.527759e+13	3.715819e+13	4.5
2000-01-12	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	...	1.886915e+14	3.715819e+13	4.5

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	...	Turnover_mean_lag30	Turnover_std_lag3	Turnover
Date														
2000-01-13	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	...	1.229175e+14	9.301846e+13	9.3(
2000-01-14	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	...	8.513328e+13	9.278553e+13	9.2
2000-01-17	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	...	7.699038e+13	2.624704e+13	7.7

5 rows × 45 columns

In [9]:
```
df.Date = pd.to_datetime(df.Date, format="%Y-%m-%d")
df["month"] = df.Date.dt.month
df["week"] = df.Date.dt.week
df["day"] = df.Date.dt.day
df["day_of_week"] = df.Date.dt.dayofweek
df.head()
```

```
<ipython-input-9-aaf895c467cb>:3: FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated.  Pleas
e use Series.dt.isocalendar().week instead.
  df["week"] = df.Date.dt.week
```

Out[9]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	...	Trades_mean_lag3	Trades_mean_lag7	Trades_mea
Date														
2000-01-11	2000-01-11	HCLTECH	EQ	580.00	1550.0	1725.00	1492.00	1560.00	1554.45	1582.72	...	61192.984375	61078.179688	6046
2000-01-12	2000-01-12	HCLTECH	EQ	1554.45	1560.0	1678.85	1560.00	1678.85	1678.85	1657.05	...	61192.984375	61078.179688	6046
2000-01-13	2000-01-13	HCLTECH	EQ	1678.85	1790.0	1813.20	1781.00	1813.20	1813.20	1804.69	...	61192.984375	61078.179688	6046
2000-01-14	2000-01-14	HCLTECH	EQ	1813.20	1958.3	1958.30	1835.00	1958.30	1958.30	1939.90	...	61192.984375	61078.179688	6046

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	...	Trades_mean_lag3	Trades_mean_lag7	Trades_mea
Date														
2000-01-17	2000-01-17	HCLTECH	EQ	1958.30	2115.0	2115.00	1801.65	1801.65	1801.65	1990.55	...	61192.984375	61078.179688	6046

5 rows × 49 columns

In [11]:
```
df_train = df[df.Date < "2019"]
df_valid = df[df.Date >= "2019"]

exogenous_features = ["High_mean_lag3", "High_std_lag3", "Low_mean_lag3", "Low_std_lag3",
                      "Volume_mean_lag3", "Volume_std_lag3", "Turnover_mean_lag3",
                      "Turnover_std_lag3", "Trades_mean_lag3", "Trades_std_lag3",
                      "High_mean_lag7", "High_std_lag7", "Low_mean_lag7", "Low_std_lag7",
                      "Volume_mean_lag7", "Volume_std_lag7", "Turnover_mean_lag7",
                      "Turnover_std_lag7", "Trades_mean_lag7", "Trades_std_lag7",
                      "High_mean_lag30", "High_std_lag30", "Low_mean_lag30", "Low_std_lag30",
                      "Volume_mean_lag30", "Volume_std_lag30", "Turnover_mean_lag30",
                      "Turnover_std_lag30", "Trades_mean_lag30", "Trades_std_lag30","month", "week", "day", "day_of_w
```

In [14]:
```
model = auto_arima(df_train.VWAP, exogenous=df_train[exogenous_features], trace=True, error_action="ignore", suppress
model.fit(df_train.VWAP, exogenous=df_train[exogenous_features])

forecast = model.predict(n_periods=len(df_valid), exogenous=df_valid[exogenous_features])
df_valid["Forecast_ARIMAX"] = forecast
```

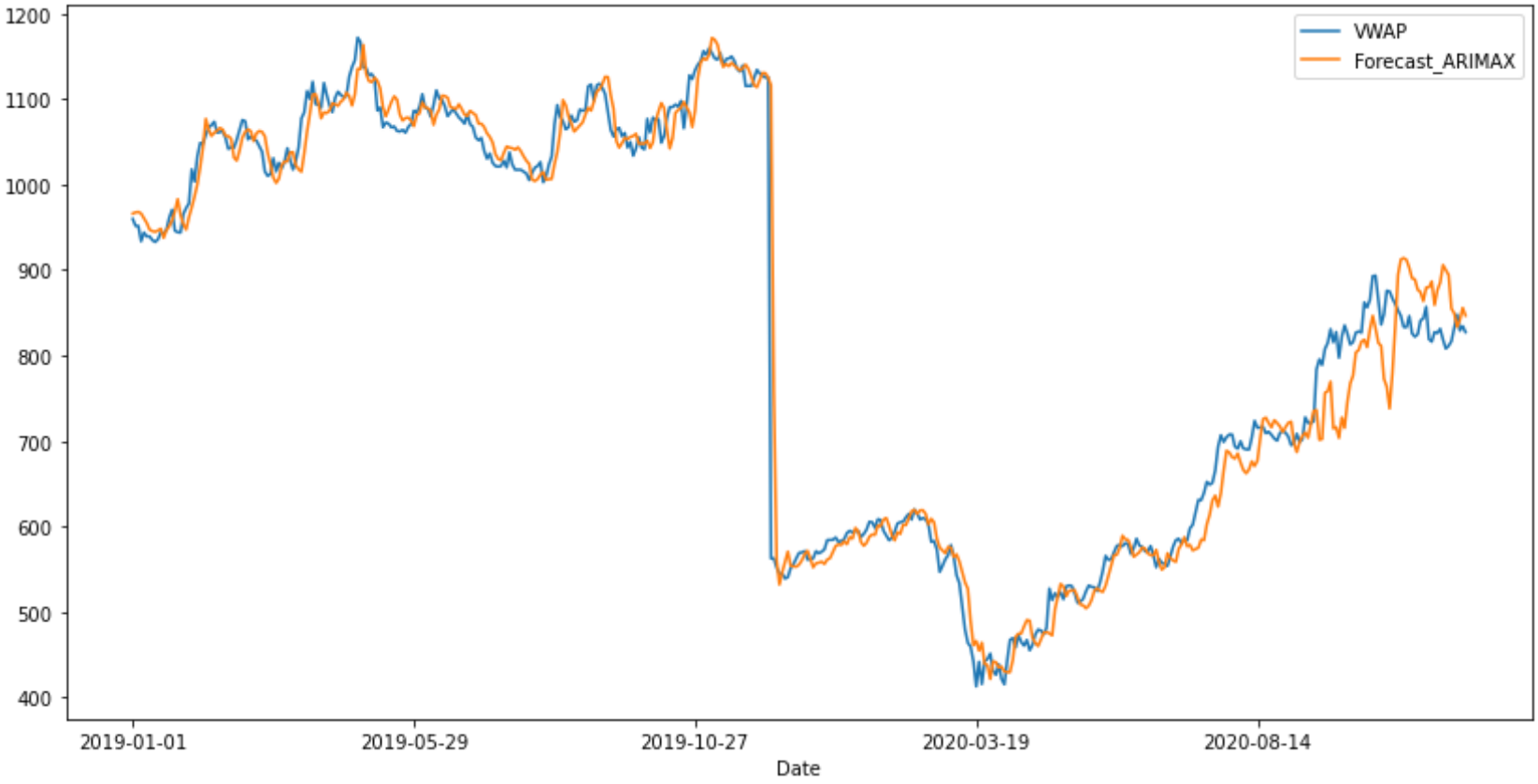
```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=46880.199, Time=26.60 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=48402.658, Time=19.10 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=47058.501, Time=15.14 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=47422.462, Time=17.15 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=83636.203, Time=13.09 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=46916.165, Time=20.28 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=46860.595, Time=20.16 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=46962.930, Time=19.48 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=46851.086, Time=20.33 sec
 ARIMA(3,0,0)(0,0,0)[0] intercept   : AIC=46854.759, Time=22.82 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=46861.201, Time=21.04 sec
```

```
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=46849.049, Time=17.82 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=47056.197, Time=14.54 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=46852.592, Time=23.92 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=46858.324, Time=18.84 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=46960.772, Time=18.42 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=46859.066, Time=22.73 sec

Best model:  ARIMA(2,0,0)(0,0,0)[0]
Total fit time: 334.098 seconds
```

```
C:\Users\U.R Computer\anaconda\lib\site-packages\statsmodels\tsa\base\tsa_model.py:376: ValueWarning: No supported in
dex is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
<ipython-input-14-e46cb50284b6>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_valid["Forecast_ARIMAX"] = forecast
```

In [15]:
```
df_valid[["VWAP", "Forecast_ARIMAX"]].plot(figsize=(14, 7))
```

Out[15]:
```
<AxesSubplot:xlabel='Date'>
```



In [16]:
```
print("RMSE of Auto ARIMAX:", np.sqrt(mean_squared_error(df_valid.VWAP, df_valid.Forecast_ARIMAX)))
print("\nMAE of Auto ARIMAX:", mean_absolute_error(df_valid.VWAP, df_valid.Forecast_ARIMAX))
```

```
RMSE of Auto ARIMAX: 39.92641923127909

MAE of Auto ARIMAX: 22.35014672643549
```

In [18]:
```
params = {"objective": "regression"}

dtrain = lgb.Dataset(df_train[exogenous_features], label=df_train.VWAP.values)
dvalid = lgb.Dataset(df_valid[exogenous_features])

model_lgb = lgb.train(params, train_set=dtrain)
```

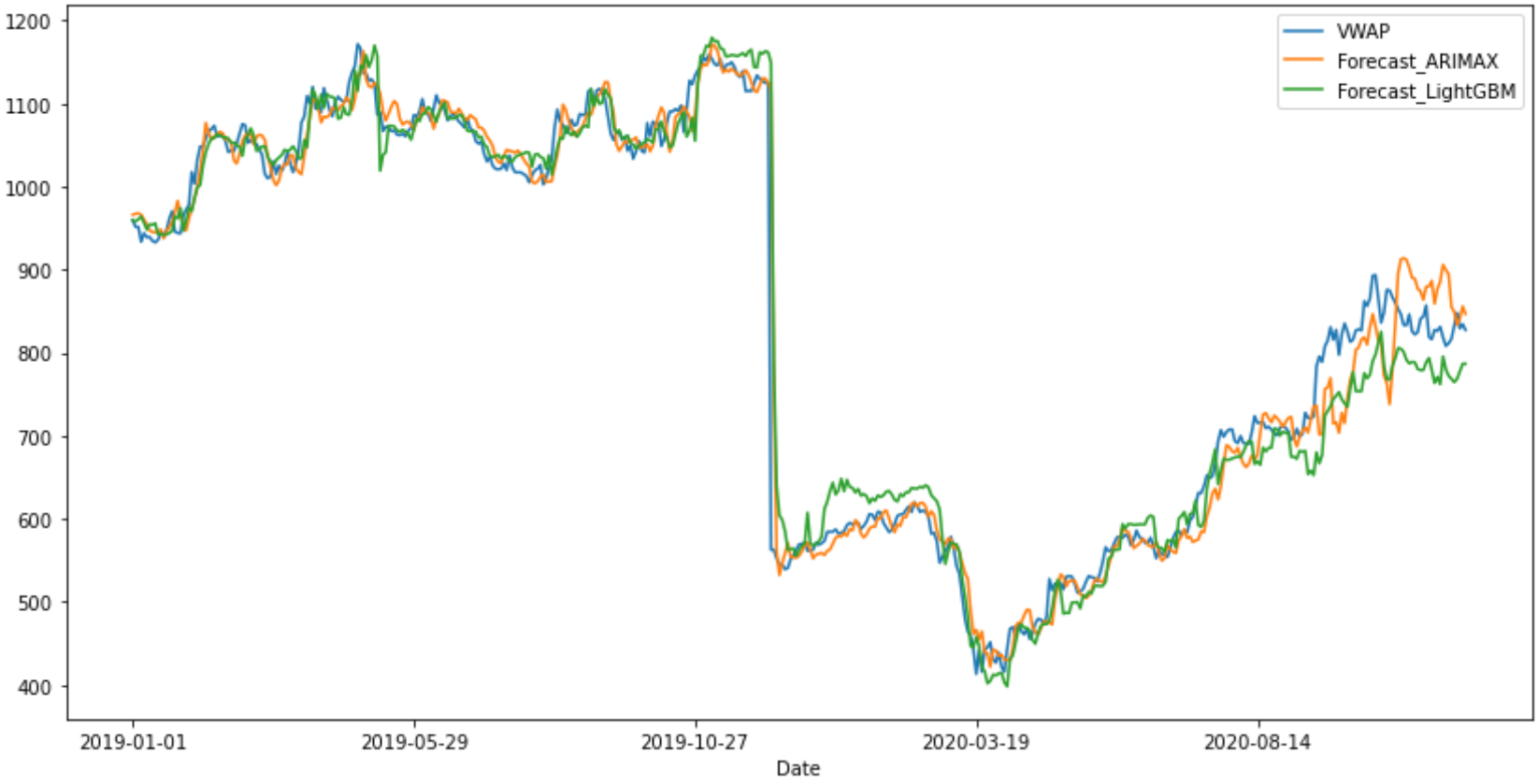
```
forecast = model_lgb.predict(df_valid[exogenous_features])
df_valid["Forecast_LightGBM"] = forecast
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.060965 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 7756
[LightGBM] [Info] Number of data points in the train set: 4723, number of used features: 34
[LightGBM] [Info] Start training from score 615.885064
```

```
<ipython-input-18-307b1df82ff0>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_valid["Forecast_LightGBM"] = forecast
```

In [19]:
```
df_valid[["VWAP", "Forecast_ARIMAX", "Forecast_LightGBM"]].plot(figsize=(14, 7))
```

Out[19]:
```
<AxesSubplot:xlabel='Date'>
```



In [ ]: