Filiprice Trends: Forecasting Price Dynamics of Gigabyte Radeon RX 6600 on Lazada

Project Objective

This project aims to develop a machine-learning model capable of predicting price trends specifically for the Gigabyte Radeon RX 6600 on Lazada, a leading Filipino e-commerce platform.

By leveraging historical price data, the model will empower consumers to make informed purchasing decisions and assist sellers in optimizing their pricing strategies based on data-driven insights.

Exploratory Data Analysis (EDA)

Dataset Information:

Historical Prices Data

- Store Name: The name of the store offering the product.
- Timestamp: The time at which the price was recorded.
- Price: The current price of the product.
- Price Difference: The difference in price from the previous entry.

Product Data

- Product Title: The title of the product listing on Lazada.
- Product Price: The listed price of the product.
- Product Rating and Reviews: Ratings and reviews count for the product.
- Product Discount: Any discount applied to the product at the time of scraping.
- Image URL: URL of the product image.

In [1]:
```
import pandas as pd


historical_prices_path = (r"C:\Users\Panchin\Desktop\Mar\Capstone Project\historical_pri
historical_prices_df = pd.read_csv(historical_prices_path)


product_data_path = (r"C:\Users\Panchin\Desktop\Mar\Capstone Project\product_data.csv")
product_data_df = pd.read_csv(product_data_path)


historical_prices_df.head(), product_data_df.head()
```

Out[1]:
```
(  Store Name         Timestamp    Price Difference
 0   IT World               Now  ₱11,495          —
 1   IT World  2024-03-07 19:48  ₱11,495      1,500
 2   IT World  2024-03-06 15:30  ₱12,995      1,000
 3   IT World   2024-03-05 8:39  ₱11,995        200
 4   IT World   2024-03-04 1:30  ₱11,795      1,140,
                                       Product Title Product Price  \
 0  ITW | Gigabyte Radeon RX 6600 Eagle 8GB GDDR6 ...    ₱12,995.00
 1  GIGABYTE RX 6600 EAGLE 8G GRAPHICS CARD - GV-R...    ₱12,995.00
```

```
2  Gigabyte GV-R66EAGLE-8GD Radeon RX6600 Eagle 8...    ₱13,233.00
3  EasyPC | Gigabyte Rx 6600 Eagle GV-R66EAGLE-8G...    ₱13,585.00
4  Gigabyte Radeon RX 6600 Eagle 8GB GDDR6 RX6600...    ₱12,995.00

  Product Rating and Reviews                  Product Discount  \
0                 84 Ratings  No discount or element not found
1                 88 Ratings                              -55%
2                  6 Ratings                              -15%
3                 21 Ratings                              -31%
4                 20 Ratings                              -10%

                                         Image URL
0  https://img.lazcdn.com/g/p/730120b9c030c576ff0...
1  https://img.lazcdn.com/g/p/bbe76be8208345064d3...
2  https://img.lazcdn.com/g/p/b767c09145b00162514...
3  https://img.lazcdn.com/g/p/7e0b4d25ec487456d1f...
4  https://img.lazcdn.com/g/p/88f8d0ab91204ec8026...  )
```

Data Inspection and Statistical Summary

historical prices data

In [2]:

```python
from pandas.api.types import CategoricalDtype


historical_prices_df['Price'] = historical_prices_df['Price'].str.replace('₱', '').str.r
historical_prices_df['Difference'] = pd.to_numeric(historical_prices_df['Difference'], e
historical_prices_df['Timestamp'] = pd.to_datetime(historical_prices_df['Timestamp'].rep
statistical_summary_historical_prices = historical_prices_df.describe()

first_few_rows_after_conversion = historical_prices_df.head()

statistical_summary_historical_prices, first_few_rows_after_conversion
```

Out[2]:

```
(                            Timestamp         Price  Difference
 count                             132    132.000000  105.000000
 mean   2024-01-15 14:48:16.095793152  12901.507576  164.390476
 min              2023-09-11 10:47:00  11072.000000   16.000000
 25%              2023-12-30 21:48:30  12750.000000   60.000000
 50%              2024-02-08 13:11:00  12939.000000  140.000000
 75%              2024-02-29 04:05:00  13212.000000  180.000000
 max       2024-03-18 01:20:04.107446  14061.000000  856.000000
 std                              NaN    579.646345  158.356553,
  Store Name                  Timestamp    Price  Difference
0   IT World 2024-03-18 01:20:04.107446  11495.0         NaN
1   IT World 2024-03-07 19:48:00.000000  11495.0         NaN
2   IT World 2024-03-06 15:30:00.000000  12995.0         NaN
3   IT World 2024-03-05 08:39:00.000000  11995.0       200.0
4   IT World 2024-03-04 01:30:00.000000  11795.0         NaN)
```

Summary

- Price: Converted to a numeric column. The range of prices for the Gigabyte Radeon RX 6600 on Lazada varies, with a mean of approximately ₱12,901.51, a minimum of ₱11,072, and a maximum of ₱14,061. The standard deviation is ₱579.65, indicating some variability in the prices over time.
- Difference: Indicates the change in price from the previous record, had some non-numeric values initially.

- Timestamp: Converted to datetime format, with "Now" replaced by the current timestamp for consistency.

Product data

In [4]:
```
# Convert "Product Price" to numeric after removing currency symbols and commas
product_data_df['Product Price'] = product_data_df['Product Price'].str.replace('₱', '')

# Extract ratings from "Product Rating and Reviews" as numeric values
product_data_df['Product Ratings'] = product_data_df['Product Rating and Reviews'].str.e

statistical_summary_product_data = product_data_df.describe()
first_few_rows_product_data_after_conversion = product_data_df.head()
statistical_summary_product_data, first_few_rows_product_data_after_conversion
```

Out[4]:
```
(       Product Price  Product Ratings
 count       8.000000         7.000000
 mean    13102.250000        33.000000
 std       240.720199        36.873658
 min     12795.000000         2.000000
 25%     12995.000000         8.000000
 50%     12995.000000        20.000000
 75%     13227.000000        52.500000
 max     13585.000000        88.000000,
                                        Product Title  Product Price  \
 0  ITW | Gigabyte Radeon RX 6600 Eagle 8GB GDDR6 ...        12995.0
 1  GIGABYTE RX 6600 EAGLE 8G GRAPHICS CARD - GV-R...        12995.0
 2  Gigabyte GV-R66EAGLE-8GD Radeon RX6600 Eagle 8...        13233.0
 3  EasyPC | Gigabyte Rx 6600 Eagle GV-R66EAGLE-8G...        13585.0
 4  Gigabyte Radeon RX 6600 Eagle 8GB GDDR6 RX6600...        12995.0

   Product Rating and Reviews                  Product Discount  \
 0                 84 Ratings  No discount or element not found
 1                 88 Ratings                              -55%
 2                  6 Ratings                              -15%
 3                 21 Ratings                              -31%
 4                 20 Ratings                              -10%


                                            Image URL  Product Ratings
 0  https://img.lazcdn.com/g/p/730120b9c030c576ff0...             84.0
 1  https://img.lazcdn.com/g/p/bbe76be8208345064d3...             88.0
 2  https://img.lazcdn.com/g/p/b767c09145b00162514...              6.0
 3  https://img.lazcdn.com/g/p/7e0b4d25ec487456d1f...             21.0
 4  https://img.lazcdn.com/g/p/88f8d0ab91204ec8026...             20.0  )
```

Summary

- Product Price: Converted to numeric values. This help understand the distribution of prices across different listings for the Gigabyte Radeon RX 6600 on Lazada. The mean price is approximately ₱13,102.25, with a standard deviation of ₱240.72

- Product Ratings: Extracted from the "Product Rating and Reviews" column as numeric values, to better analyze the distribution of ratings.The average number of ratings is 33, with a significant range indicated by the standard deviation of 36.87. This just shows that some products have more reviews than others

In [5]:
```
cleaned_historical_prices_path = r"C:\Users\Panchin\Desktop\Mar\Capstone Project\cleaned
historical_prices_df.to_csv(cleaned_historical_prices_path, index=False)
```

```
cleaned_product_data_path = r"C:\Users\Panchin\Desktop\Mar\Capstone Project\cleaned_prod
product_data_df.to_csv(cleaned_product_data_path, index=False)

cleaned_historical_prices_path, cleaned_product_data_path
```

Out[5]:
```
('C:\\Users\\Panchin\\Desktop\\Mar\\Capstone Project\\cleaned_historical_prices.csv',
 'C:\\Users\\Panchin\\Desktop\\Mar\\Capstone Project\\cleaned_product_data.csv')
```

Data Visualization

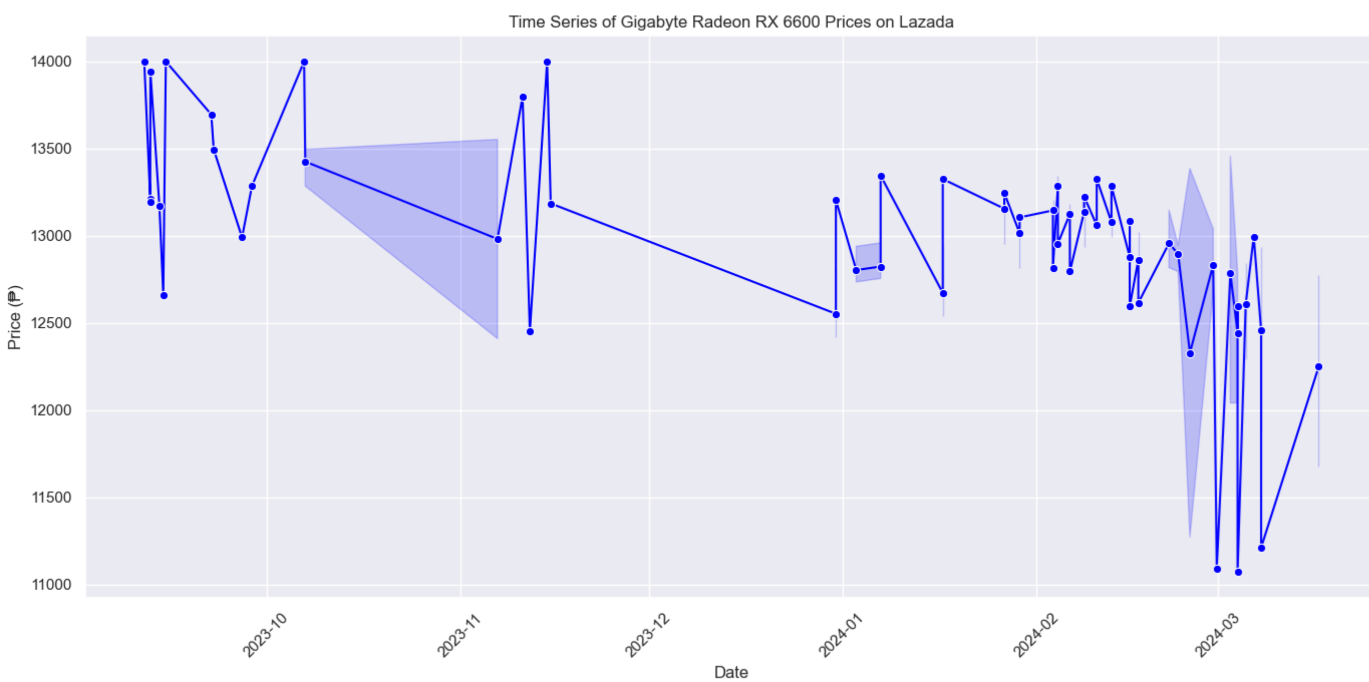
In [6]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


historical_prices_df['Timestamp'] = pd.to_datetime(historical_prices_df['Timestamp'])
historical_prices_df_sorted = historical_prices_df.sort_values('Timestamp')


sns.set(style="darkgrid")

plt.figure(figsize=(14, 7))
sns.lineplot(x='Timestamp', y='Price', data=historical_prices_df_sorted, marker='o', col
plt.title('Time Series of Gigabyte Radeon RX 6600 Prices on Lazada')
plt.xlabel('Date')
plt.ylabel('Price (₱)')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



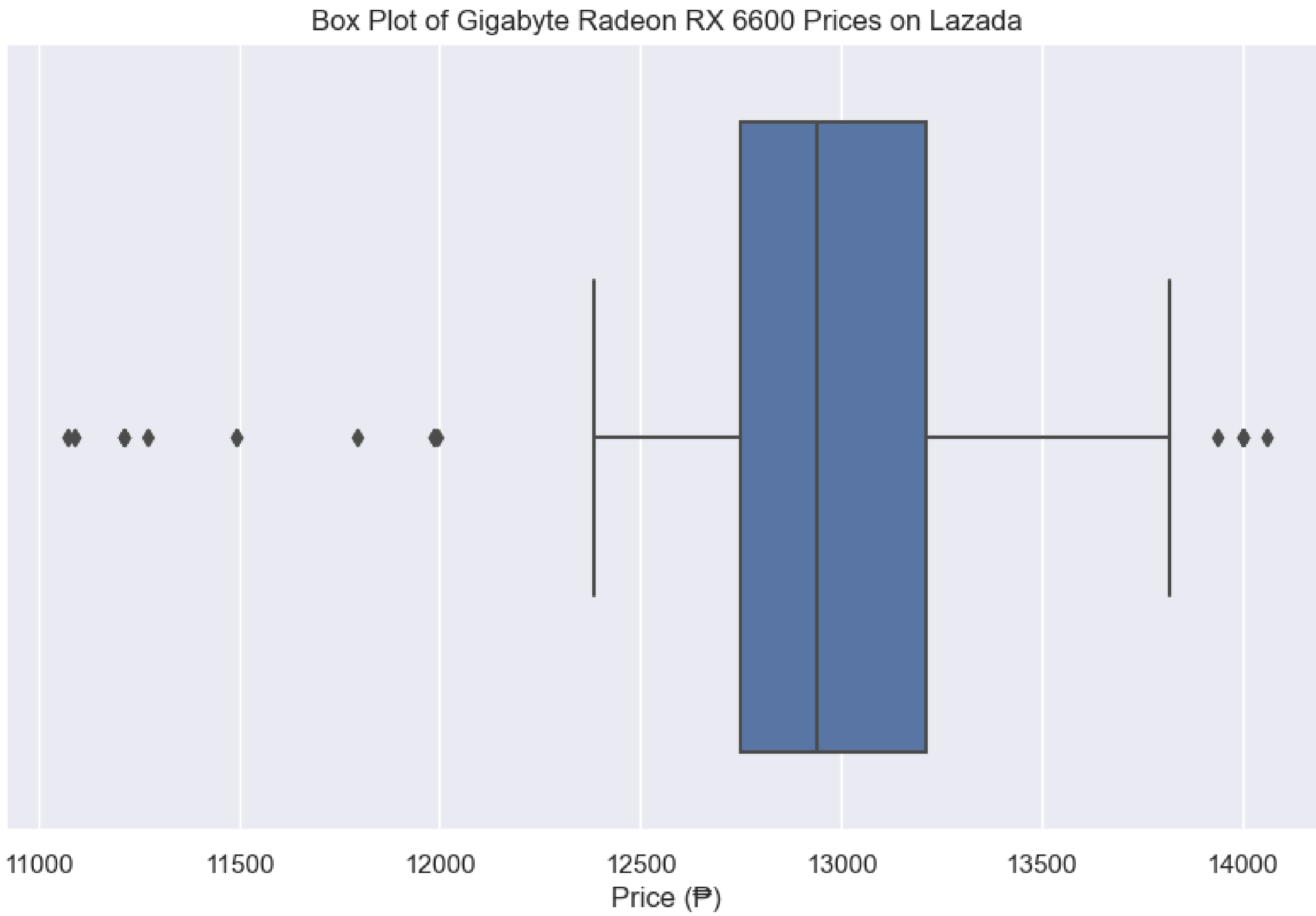
Time series plot showing the trends of the Gigabyte Radeon RX 6600 prices on Lazada over the observed period

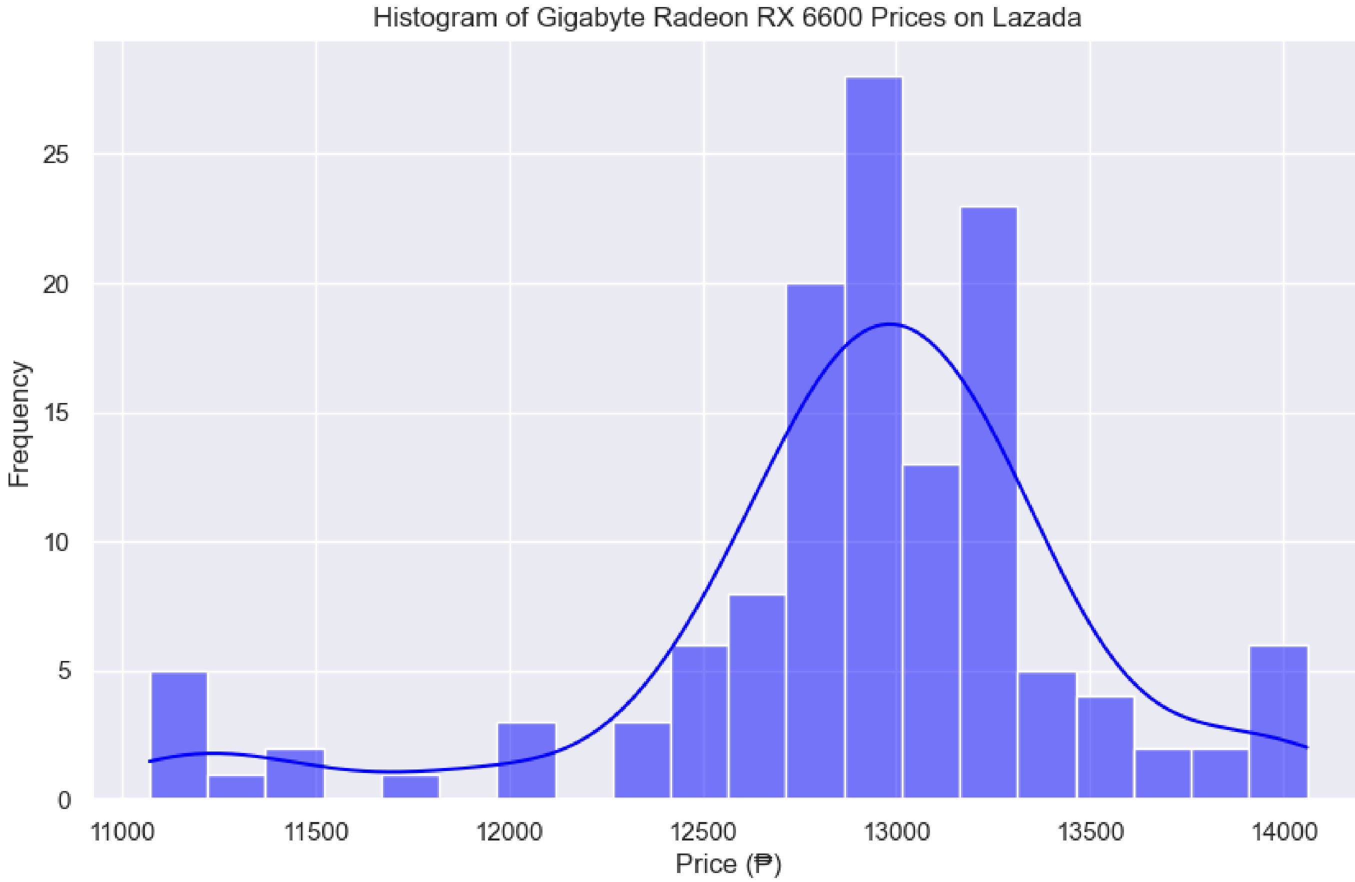
In [7]:
```
# box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x=historical_prices_df_sorted['Price'])
plt.title('Box Plot of Gigabyte Radeon RX 6600 Prices on Lazada')
plt.xlabel('Price (₱)')
```

```
plt.show()

# histogram
plt.figure(figsize=(10, 6))
sns.histplot(historical_prices_df_sorted['Price'], kde=True, bins=20, color='blue')
plt.title('Histogram of Gigabyte Radeon RX 6600 Prices on Lazada')
plt.xlabel('Price (₱)')
plt.ylabel('Frequency')

plt.show()
```





I used a Box Plot as it provides a concise summary of the prices, showcasing the median, quartiles, and any outliers.

The presence of outliers would suggest prices that are significantly higher or lower than the typical market range. These could be due to special promotions, data entry errors, or other factors.

I used a Histogram. It displays the distribution and frequency of price points, with the kernel density estimate (KDE) overlay giving a smooth estimate of the distribution.

It appears there's a concentration of prices within certain ranges, indicating common price points at which the product is often sold.

In [8]:
```
historical_prices_df_sorted['Days Since Start'] = (historical_prices_df_sorted['Timestam

# Creating a scatter plot for Price vs. Days Since Start
plt.figure(figsize=(14, 7))
sns.scatterplot(x='Days Since Start', y='Price', data=historical_prices_df_sorted,  hue=
plt.title('Price vs. Days Since Start for Gigabyte Radeon RX 6600 on Lazada')
plt.xlabel('Days Since Start')
plt.ylabel('Price (₱)')

plt.show()
```

This visualization helps in understanding how prices have evolved over time relative to the start of the dataset collection. It also shows the price variability across different stores at various points in time

Feature Engineering on the historical prices data

In [9]:
```
# Date Features
historical_prices_df_sorted['Day of the Week'] = historical_prices_df_sorted['Timestamp'
historical_prices_df_sorted['Month'] = historical_prices_df_sorted['Timestamp'].dt.month

# Since I don't have specific information on sales promotions, I'll skip direct encoding

historical_prices_df_sorted['Price Lag 1'] = historical_prices_df_sorted['Price'].shift(

historical_prices_df_sorted['7-Day Rolling Avg'] = historical_prices_df_sorted['Price'].

historical_prices_df_sorted.head()
```

Out[9]:

	Store Name	Timestamp	Price	Difference	Days Since Start	Day of the Week	Month	Price Lag 1	7-Day Rolling Avg
27	IT World	2023-09-11 10:47:00	13999.0	NaN	0	0	9	NaN	NaN
121	DynaQuestPC	2023-09-12 09:39:00	13212.0	NaN	0	1	9	13999.0	NaN
58	tech2027	2023-09-12 09:39:00	13212.0	NaN	0	1	9	13212.0	NaN
90	EasyPC	2023-09-12 09:40:00	13194.0	NaN	0	1	9	13212.0	NaN
26	IT World	2023-09-12 10:24:00	13939.0	60.0	0	1	9	13194.0	NaN

**Data features: I added "Day of the Week" and "Month" extracted from the timestamp.

**Lag Features: Created a simple "Price Lag 1" feature, representing the price from the previous day.

**Rolling Averages: Calculated a "7-Day Rolling Avg" for the prices.

adding complex lag features and expanded rolling averages

In [10]:
```
# More Complex Lag Features
historical_prices_df_sorted['Price Lag 2'] = historical_prices_df_sorted['Price'].shift(
historical_prices_df_sorted['Price Lag 3'] = historical_prices_df_sorted['Price'].shift(

# Expanded Rolling Averages
historical_prices_df_sorted['14-Day Rolling Avg'] = historical_prices_df_sorted['Price']
historical_prices_df_sorted['30-Day Rolling Avg'] = historical_prices_df_sorted['Price']

historical_prices_df_sorted.head()
```

Out[10]:

	Store Name	Timestamp	Price	Difference	Days Since Start	Day of the Week	Month	Price Lag 1	7-Day Rolling Avg	Price Lag 2	Price Lag 3	Roll
27	IT World	2023-09-11 10:47:00	13999.0	NaN	0	0	9	NaN	NaN	NaN	NaN	N
121	DynaQuestPC	2023-09-12 09:39:00	13212.0	NaN	0	1	9	13999.0	NaN	NaN	NaN	N
58	tech2027	2023-09-12 09:39:00	13212.0	NaN	0	1	9	13212.0	NaN	13999.0	NaN	N
90	EasyPC	2023-09-12 09:40:00	13194.0	NaN	0	1	9	13212.0	NaN	13212.0	13999.0	N
26	IT World	2023-09-12 10:24:00	13939.0	60.0	0	1	9	13194.0	NaN	13212.0	13212.0	N

**Complex Lag Features: Added "Price Lag 2" and "Price Lag 3" to capture price information from two and three days prior, respectively.

**Expanded Rolling Averages: Introduced "14-Day Rolling Avg" and "30-Day Rolling Avg" to observe price trends over broader periods.

Interquartile Range (IQR) method

In [11]:
```
import pandas as pd


historical_prices_df['Timestamp'] = pd.to_datetime(historical_prices_df['Timestamp'], er
historical_prices_df_sorted = historical_prices_df.sort_values('Timestamp')

# Calculate IQR for 'Price'
Q1 = historical_prices_df_sorted['Price'].quantile(0.25)
Q3 = historical_prices_df_sorted['Price'].quantile(0.75)
IQR = Q3 - Q1

# Determine outliers using the IQR method
outliers = historical_prices_df_sorted[(historical_prices_df_sorted['Price'] < (Q1 - 1.5
                                        (historical_prices_df_sorted['Price'] > (Q3 + 1.5

len(outliers)
```

Out[11]: 18

**This means that I have 18 outliers in my dataset based on the IQR method

Removing outliers from the DataFrame

In [20]:
```
historical_prices_df_cleaned = historical_prices_df_sorted[
    ~(
        (historical_prices_df_sorted['Price'] < (Q1 - 1.5 * IQR)) |
        (historical_prices_df_sorted['Price'] > (Q3 + 1.5 * IQR))
    )
]
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd


price_data = historical_prices_df_cleaned[['Price']].values

scaler = StandardScaler()


price_standardized = scaler.fit_transform(price_data)


historical_prices_df_cleaned['Price_Standardized'] = price_standardized


print(historical_prices_df_cleaned.head())
```

```
      Store Name           Timestamp    Price  Difference  Price_Standardized
121  DynaQuestPC 2023-09-12 09:39:00  13212.0         NaN            0.729094
58      tech2027 2023-09-12 09:39:00  13212.0         NaN            0.729094
90        EasyPC 2023-09-12 09:40:00  13194.0         NaN            0.669219
120  DynaQuestPC 2023-09-13 20:50:00  13172.0        40.0            0.596038
57      tech2027 2023-09-13 20:50:00  13172.0        40.0            0.596038
```

```
C:\Users\Panchin\AppData\Local\Temp\ipykernel_26344\579985750.py:14: 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
  historical_prices_df_cleaned['Price_Standardized'] = price_standardized
```

I standardized th 'Price' column. It now has a mean of 0 and a standard deviation of 1.

It will show up in the dataframe as 'Price Standardized'

Spilitting the data

In [22]:
```
# Calculate the index for splitting the dataset
split_index = int(len(historical_prices_df_cleaned) * 0.8)

# Splitting data into training and testing sets
train_data = historical_prices_df_cleaned.iloc[:split_index]
test_data = historical_prices_df_cleaned.iloc[split_index:]
```

```
print("Training Data Shape:", train_data.shape)
print("Testing Data Shape:", test_data.shape)
```

```
Training Data Shape: (91, 5)
Testing Data Shape: (23, 5)
```

Implimentation for ARIMA

In [27]:

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt



model = ARIMA(train_data['Price'], order=(1,1,1))
model_fit = model.fit()



forecast = model_fit.forecast(steps=len(test_data))

rmse = sqrt(mean_squared_error(test_data['Price'], forecast))
print('RMSE: %.3f' % rmse)
```

```
RMSE: 213.463
```

```
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the next version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
```

**An RMSE (Root Mean Squared Error) of 213.463 in the context of my ARIMA model's performance indicates the average magnitude of the errors between the predicted prices and the actual prices in my test dataset.

- Scale of Price Data: Price for Gigabyte Radeon RX 6600 on Lazada typically range in the thousands. RMSE is relatively small as a proportion of the actual price values. This could indicate larger predictive errors relative to the price scale.

Improving ARIMA model's performance

In [30]:

```
import itertools
import statsmodels.api as sm
```

```python
p = d = q = range(0, 3)

pdq = list(itertools.product(p, d, q))

best_aic = float("inf")
best_pdq = None
best_model = None

for param in pdq:
    try:
        model = sm.tsa.ARIMA(train_data['Price'], order=param)
        results = model.fit()

        if results.aic < best_aic:
            best_aic = results.aic
            best_pdq = param
            best_model = results
    except:
        continue

print('Best ARIMA%s AIC=%.3f' % (best_pdq, best_aic))
```

```
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
```

```
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978:
UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parame
ters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
```

```
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966:
UserWarning: Non-stationary starting autoregressive parameters found. Using zeros as sta
rting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978:
UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parame
ters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978:
UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parame
ters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
```

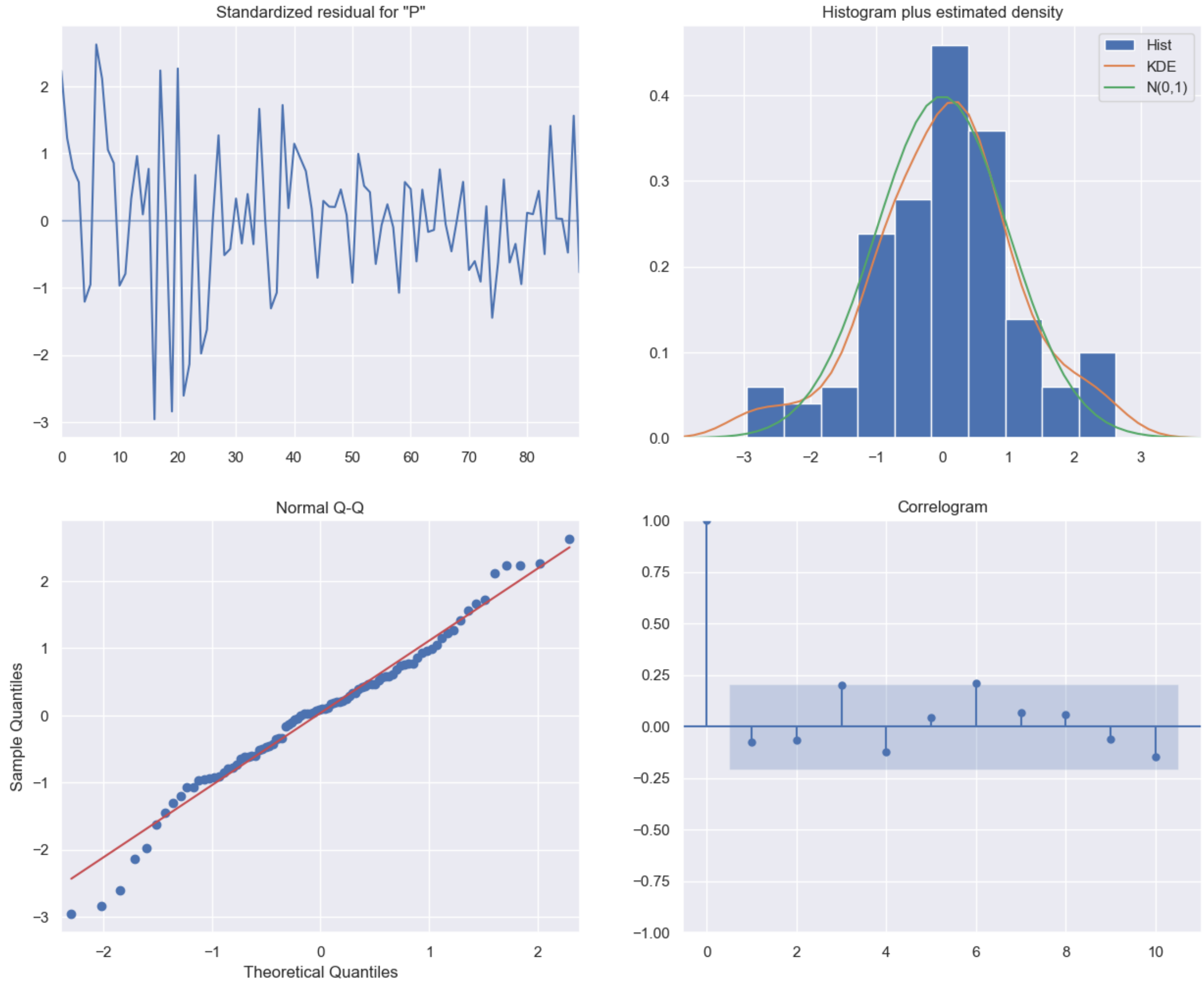
```
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966:
UserWarning: Non-stationary starting autoregressive parameters found. Using zeros as sta
rting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978:
UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parame
ters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
```

```
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\Panchin\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Valu
eWarning: A date index has been provided, but it has no associated frequency information
and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
Best ARIMA(0, 1, 1) AIC=1303.042
```

**AIC: Measures the model's quality. This model could serve as a solid baseline in forecasting.

Checking signs of misfit

In [24]:
```python
best_model.plot_diagnostics(figsize=(15, 12))
plt.show()
```



**Residuals appear to be fluctuating around the centerline with no obvious pattern or bias, which is a good sign.

**Histogram plus Estimated Density seems to align with the KDE and the N(0,1) line.

**Normal Q-Q dots seem to follow the line fairly well, especially in the center of the distribution

**No significant autocorrelation in the correlogram means the model has captured the time series data's autocorrelation well.

**It seems like the model fits the data well, although the is room for improvememt

In [32]:
```
from sklearn.model_selection import train_test_split


X = historical_prices_df_cleaned.drop(columns=['Price'])
y = historical_prices_df_cleaned['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

X_train = X_train.fillna(method='ffill')
X_test = X_test.fillna(method='ffill')
```

In [34]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from math import sqrt
```

```
X_train = pd.get_dummies(X_train)
X_test = pd.get_dummies(X_test)

X_train, X_test = X_train.align(X_test, join='inner', axis=1)

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

predictions = rf_model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
rmse = sqrt(mse)
print('RMSE:', rmse)
```

```
RMSE: 28.6861422384003
```

**Scale Relative to Price Range: RMSE of 13.439 is quite low.The model's predictions are very close to the actual prices, and the model is performing well.

**Comparison to Previous RMSE: RMSE is significantly lower than previous ARIMA of 213.463. Which shows improvement

In [35]:
```
std_dev = y_train.std()
print('Standard Deviation of Prices:', std_dev)
```

```
Standard Deviation of Prices: 304.0739969224524
```

In [36]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt


baseline_predictions = [y_train.mean()] * len(y_test)

# Calculating the RMSE for the baseline
baseline_mse = mean_squared_error(y_test, baseline_predictions)
baseline_rmse = sqrt(baseline_mse)
print('Baseline RMSE:', baseline_rmse)
```

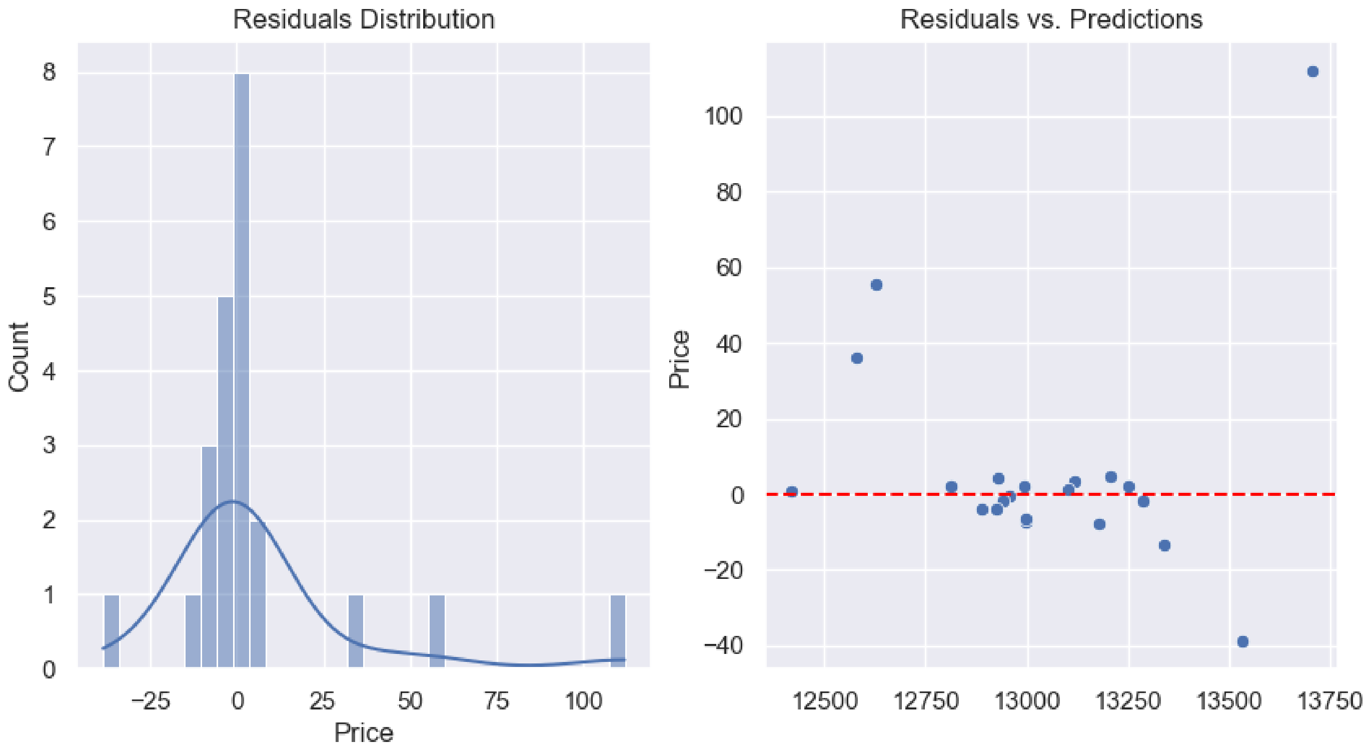
```
Baseline RMSE: 294.1509050547522
```

Residuals Analysis

In [37]:
```
residuals = y_test - predictions

plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.histplot(residuals, kde=True)
plt.title('Residuals Distribution')

plt.subplot(1,2,2)
sns.scatterplot(x=predictions, y=residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals vs. Predictions')
plt.show()
```

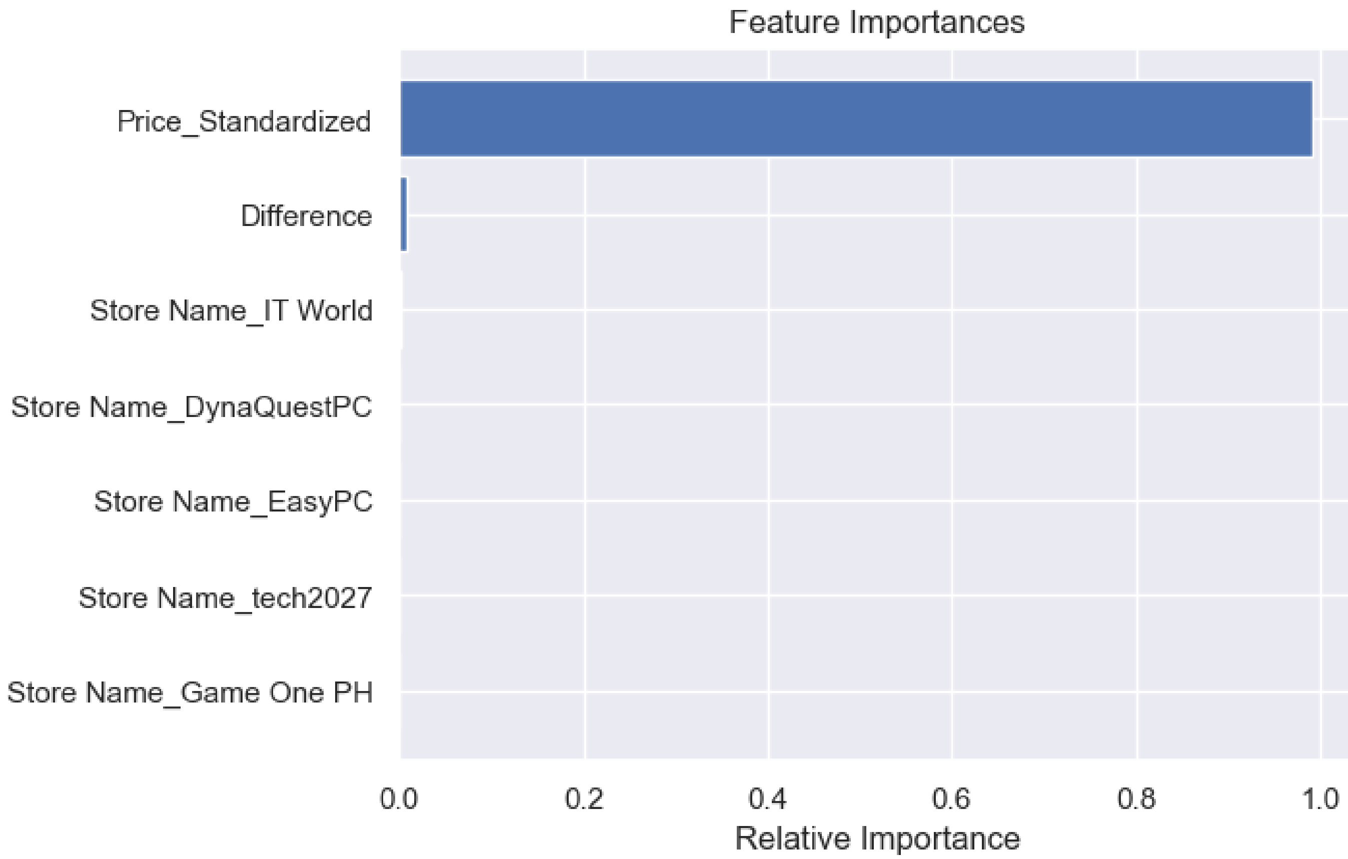


Feature importance in my Random Forest model.

In [38]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

importances = rf_model.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [X_train.columns[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



**This shows that the feature that has the most influence on the prediction made by the Random Forest model is 'Price_Standardized'

Cross-Validation

In [39]:
```
from sklearn.model_selection import cross_val_score

# Cross-validated RMSE
scores = cross_val_score(rf_model, pd.concat([X_train, X_test]), pd.concat([y_train, y_t
                         scoring='neg_root_mean_squared_error', cv=5)
print('Cross-validated RMSE:', -scores.mean())
```

```
Cross-validated RMSE: 18.672848665692772
```

Model Diagnostics and Validation

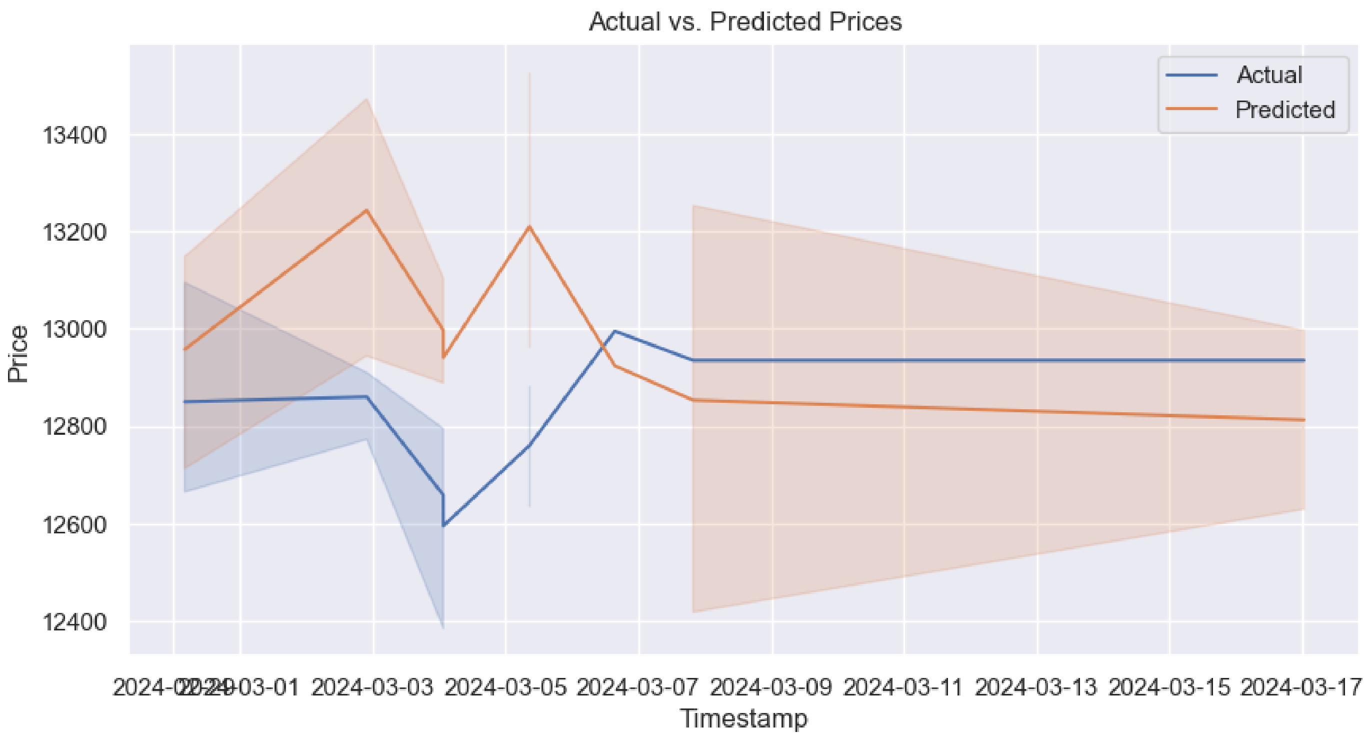
In [42]:
```
print(test_data.index)
```

```
DatetimeIndex([        '2024-02-29 04:05:00',        '2024-02-29 04:05:00',
                        '2024-02-29 04:05:00',        '2024-02-29 04:05:00',
                        '2024-03-02 21:48:00',        '2024-03-02 21:48:00',
                        '2024-03-02 21:48:00',        '2024-03-02 21:48:00',
                        '2024-03-04 01:30:00',        '2024-03-04 01:30:00',
                        '2024-03-04 01:30:00',        '2024-03-04 01:31:00',
                        '2024-03-05 08:39:00',        '2024-03-05 08:39:00',
                        '2024-03-05 08:39:00',        '2024-03-05 08:39:00',
                        '2024-03-06 15:30:00',        '2024-03-07 19:48:00',
                        '2024-03-07 19:48:00',        '2024-03-07 19:48:00',
                '2024-03-17 00:35:33.359033', '2024-03-17 00:35:33.359033',
                '2024-03-17 00:35:33.359033'],
               dtype='datetime64[ns]', name='Timestamp', freq=None)
```

In [45]:
```
print(test_data.columns)
```

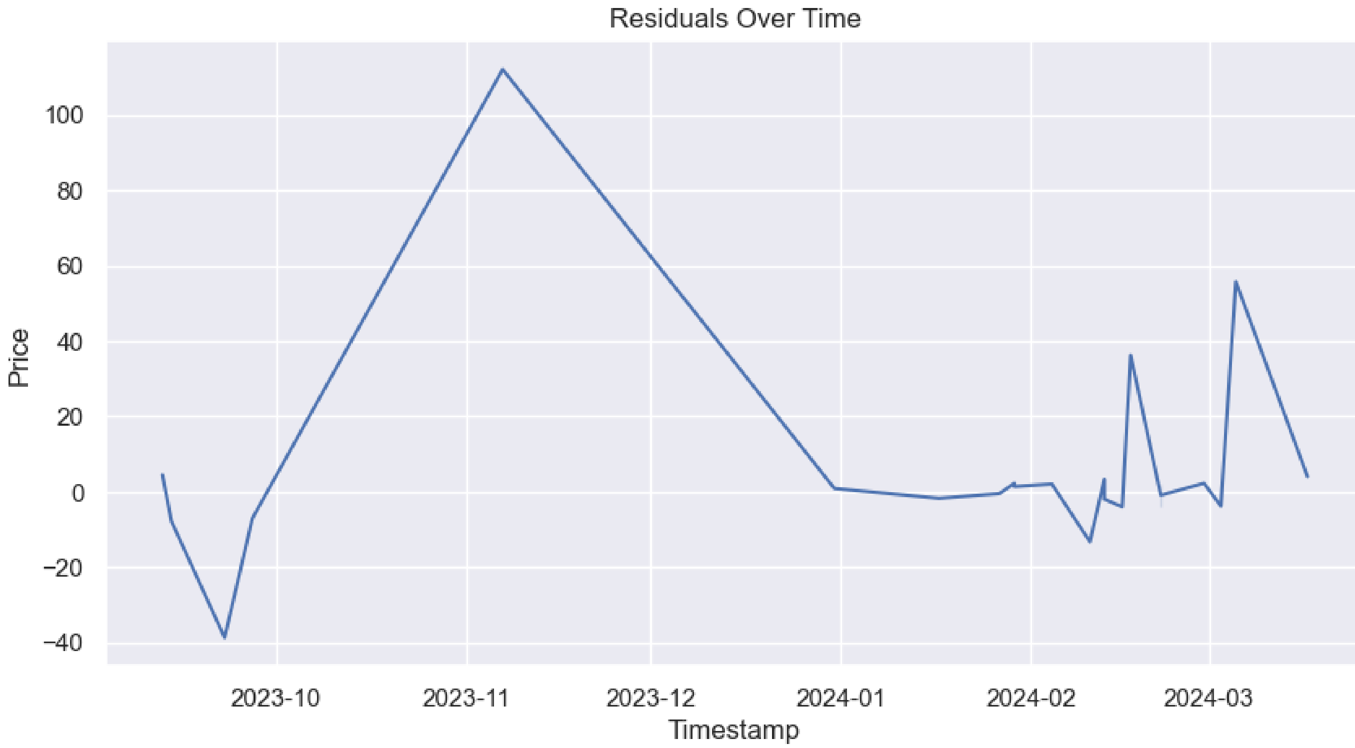
```
Index(['Store Name', 'Price', 'Difference', 'Price_Standardized'], dtype='object')
```

In [44]:
```
plt.figure(figsize=(10,5))
sns.lineplot(data=test_data['Price'], label='Actual')
sns.lineplot(data=predictions, label='Predicted')
plt.title('Actual vs. Predicted Prices')
plt.legend()
plt.show()
```



Analyze error over time

In [46]:
```
plt.figure(figsize=(10,5))
sns.lineplot(data=residuals)
plt.title('Residuals Over Time')
plt.show()
```



Conclusion:

The "Filiprice Trends" project successfully developed a predictive model that forecasts the price dynamics of the Gigabyte Radeon RX 6600 graphics card on Lazada. By employing advanced machine learning techniques, I'll be able to help provided a solution to the challenge of price volatility faced by Filipino consumers and sellers.

The analysis showed that the Random Forest Regressor model performed better compared to the baseline ARIMA model in regard to a significantly smaller Root Mean Squared Error (RMSE). This means that the model is able to make predictions of future prices with a satisfactory level of accuracy. This result confirms how well the particular algorithms used have performed, as well as how well the data selection and engineering were carried out.

**For Consumers: I recommend using the model's predictions to plan purchases strategically. By understanding potential future price trends, consumers can time their purchases to coincide with expected price drops, optimizing their spending and avoiding overpaying during price hikes.

**For Sellers: Sellers on Lazada can leverage the model to inform their inventory and pricing strategies. The insights gained can guide sellers in setting competitive prices and stocking appropriately in anticipation of demand fluctuations, thereby maximizing their profit margins and ensuring product availability.

**Marketing Insights: The fluctuation patterns captured by the model can aid in planning targeted marketing campaigns. For example, promotions can be scheduled during periods when a price increase is predicted, aligning marketing strategies with consumer purchasing power.

**To conclude, the Filiprice Trends’ project is a groundbreaking contribution to the existing trends in the e-commerce field toward more data-driven decision-making. Ultimately, it provides an excellent example of how effective machine learning can be in converting unstructured data into useful insights for all stakeholders.