Pick one of the 5 days of SPY data that we began exploring in the previous step. Let's try to build up a narrative of what happened over that day. For each question below, do some analyses and present some graphics to (at least partially) address the question for your chosen day.

1.What happened to price over the course of the day?

a) How much did it change overall, and in what direction?

b) When was it changing fastest?

c) When was holding mostly steady (if at all)?

1. What happened to sizes of individual trades over the course of the day? Does trade size seem correlated with time of day at all in your opinion?

1. What happened to the rate of trading throughout the day?

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

Since in the previous step (3) of the project we determined that day 4 was not a regular day, I'd like to learn more about what happened there.

Also, on that day the range of stock price was significantly higher compared to any other day, which I failed to formally recognize in the previous step, see below:

In [2]:
```
# loading without parsing dates for faster execution
df1 = pd.read_csv('Day1_Trades_SPY.csv')#, parse_dates = ['Time'], index_col = 'Time')
df2 = pd.read_csv('Day2_Trades_SPY.csv')#, parse_dates = ['Time'], index_col = 'Time')
df3 = pd.read_csv('Day3_Trades_SPY.csv')#, parse_dates = ['Time'], index_col = 'Time')
df4 = pd.read_csv('Day4_Trades_SPY.csv')#, parse_dates = ['Time'], index_col = 'Time')
df5 = pd.read_csv('Day5_Trades_SPY.csv')#, parse_dates = ['Time'], index_col = 'Time')

# create a list of all dataframes
dfs = [df1,df2,df3,df4,df5]
```

```
# print print range of price for each day
for num, d in enumerate(dfs,1):
    print(f"Day#{num} Price Range: {round(d['PRICE'].max() - d['PRICE'].min(),2): 6}")
```

```
Day#1 Price Range:   3.83
Day#2 Price Range:   4.12
Day#3 Price Range:   2.57
Day#4 Price Range:  20.68
Day#5 Price Range:   2.34
```

QUESTION 1. What happened to price over the course of the day?

In [3]:
```
# loading with index on Time column so that it is easier to resample the data by larger periods of time
df4 = pd.read_csv('Day4_Trades_SPY.csv', parse_dates = ['Time'], index_col = 'Time')
```

In [4]:
```
# resampling the data at 1 min frequency and calculating several metrics
df = df4['PRICE'].resample('1min').agg({'min_price':min, 'max_price': max, 'mean_price': np.mean, 'median_price'
# checking the result
# df.head()

# adding a column for hours
df['hour'] = df.index.hour.astype(int)
# adding a column for minutes
df['minute'] = df.index.minute.astype(int)
# adding a column for seconds
df['second'] = df.index.second.astype(int)

# adding count of minutes since midnight
df['minute_num'] = df.hour * 60 + df.minute
# adding count of minutes since midnight
df['second_num'] = df.hour * 60 + df.minute_num * 60 * 60 + df.second

#adding a label as a string for time series charts in 'HH:MM' format
df['label'] = df.hour.astype(str) + ':' + df.index.strftime('%M')

#resetting the index
df.reset_index(inplace = True)

# checking the result
df.head()
```

Out[4]:

	Time	min_price	max_price	mean_price	median_price	hour	minute	second	minute_num	second_num	label
0	2022-12-01 09:30:00	236.01	239.23	237.652030	237.89	9	30	0	570	2052540	9:30
1	2022-12-01 09:31:00	236.03	240.68	239.453419	239.74	9	31	0	571	2056140	9:31
2	2022-12-01 09:32:00	238.67	240.52	239.495180	239.46	9	32	0	572	2059740	9:32
3	2022-12-01 09:33:00	239.44	243.54	241.525011	241.62	9	33	0	573	2063340	9:33
4	2022-12-01 09:34:00	239.99	242.36	241.074947	241.02	9	34	0	574	2066940	9:34

In [5]:
```
# checking for missing data (after we uncovered the gap in Step3)
df.isna().sum()
```

Out[5]:
```
Time            0
min_price      14
max_price      14
mean_price     14
median_price   14
hour            0
minute          0
second          0
minute_num      0
second_num      0
label           0
dtype: int64
```

In [6]:
```
# filling missing values with zeros for all metrics (in order to visualize the gap)
df.fillna(0, inplace = True)

# get the indeces for the start and end of the gap in the data
gap_start_ind = df[df['mean_price'] == 0].index.min()
gap_end_ind = df[df['mean_price'] == 0].index.max()

# checking the section with the missing data, plus one row on each side for comparison
df[gap_start_ind-1:gap_end_ind+2]
```

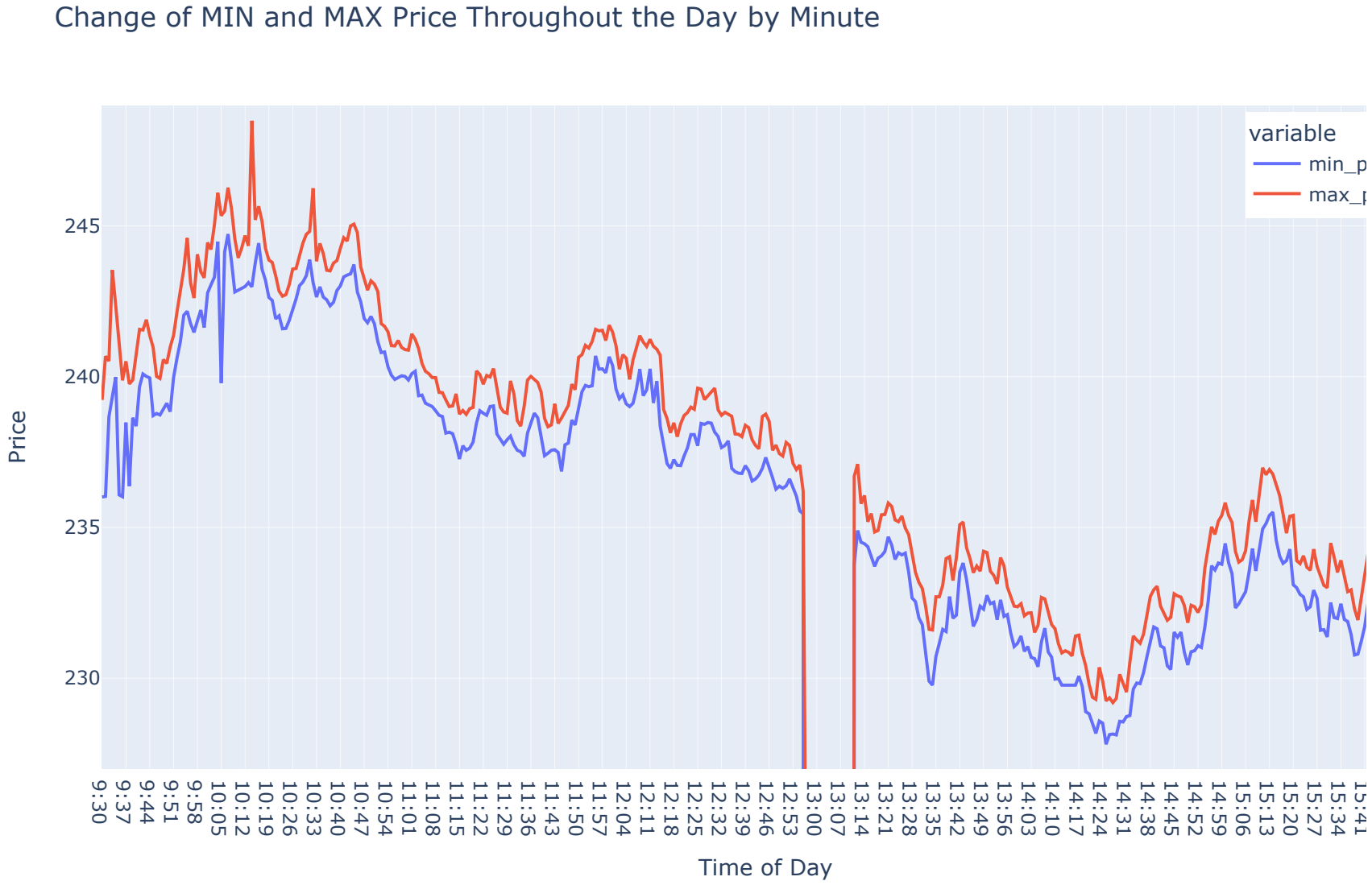
Out[6]:

	Time	min_price	max_price	mean_price	median_price	hour	minute	second	minute_num	second_num	label
206	2022-12-01 12:56:00	235.45	236.2	235.824439	235.83	12	56	0	776	2794320	12:56
207	2022-12-01 12:57:00	0.00	0.0	0.000000	0.00	12	57	0	777	2797920	12:57
208	2022-12-01 12:58:00	0.00	0.0	0.000000	0.00	12	58	0	778	2801520	12:58
209	2022-12-01 12:59:00	0.00	0.0	0.000000	0.00	12	59	0	779	2805120	12:59
210	2022-12-01 13:00:00	0.00	0.0	0.000000	0.00	13	0	0	780	2808780	13:00
211	2022-12-01 13:01:00	0.00	0.0	0.000000	0.00	13	1	0	781	2812380	13:01
212	2022-12-01 13:02:00	0.00	0.0	0.000000	0.00	13	2	0	782	2815980	13:02
213	2022-12-01 13:03:00	0.00	0.0	0.000000	0.00	13	3	0	783	2819580	13:03
214	2022-12-01 13:04:00	0.00	0.0	0.000000	0.00	13	4	0	784	2823180	13:04
215	2022-12-01 13:05:00	0.00	0.0	0.000000	0.00	13	5	0	785	2826780	13:05
216	2022-12-01 13:06:00	0.00	0.0	0.000000	0.00	13	6	0	786	2830380	13:06
217	2022-12-01 13:07:00	0.00	0.0	0.000000	0.00	13	7	0	787	2833980	13:07
218	2022-12-01 13:08:00	0.00	0.0	0.000000	0.00	13	8	0	788	2837580	13:08
219	2022-12-01 13:09:00	0.00	0.0	0.000000	0.00	13	9	0	789	2841180	13:09
220	2022-12-01 13:10:00	0.00	0.0	0.000000	0.00	13	10	0	790	2844780	13:10
221	2022-12-01 13:11:00	233.77	236.7	235.262775	235.27	13	11	0	791	2848380	13:11

In [7]:
```
# # trying to plot the result using seaborn
```

```
# fig, ax = plt.subplots(1, figsize = (15, 6), dpi=80)

# sns.lineplot(data=df, x="label", y="min_price")
# sns.lineplot(data=df, x="label", y="max_price")

# ax.set_title (f'Minimum and Maximum Price by Minute on Day 4', fontsize = 18)
# # plt.setp(ax.get_xticklabels(), visible=False)
# # setting the limit for y axis within the range of daily price change for better vizualization
# ax.set(ylim=(227, 249))
# plt.xlabel("Time of Day")
# plt.ylabel("Price")

# # only plot labels once for every 60 min worth of data
# plt.xticks([x for i,x in enumerate(df.label.to_list()) if i % 60 == 0])

# plt.show()
```

In [8]:
```
# plot min vs max price using plotly for interactive visualizations
fig = px.line(df,
              x='label',
              y=[df["min_price"], df['max_price']],
             title = "Change of MIN and MAX Price Throughout the Day by Minute",
             width=1000, height=600)

# fix the axis closer to the range of acual data (chopping off the period when trading was stopped)
fig.update_layout(yaxis_range=[227, 249])

# name axes
fig.update_xaxes(title_text = "Time of Day", title_font = {"size": 14})
fig.update_yaxes(title_text = "Price", title_font = {"size": 14})

# position the legend inside of the chart
fig.update_layout(legend=dict(
    yanchor="top",
    y=0.99,
    xanchor="right",
    x=0.99
))
fig.show()
```



OBSERVATIONS:

- From the beginning of trading until 10:04AM the price was growing
- At 10:05AM it dropped by almost 2% but quickly rebounded and continued growing until approximately 10:15AM

- At 10:16AM we've seen the first fluctuations of the min price and then it was in freefall until 11:15AM

- At that point we started seeing clearer 15 min cycles of price fluctuation. See min price below:

- At approximately 12:12PM it continued falling again and lost 1.53% by 12:57PM when the trading was halted

- The trading resumed at 13:11PM however, the price continued to fall until approximately 14:25

- From 14:25 the price started recovering and by the end of the day it approximately returned to the position where it was at the beginning of the day

QUESTION 1a: How much did the price change overall, and in what direction?

In [9]:
```
# We will calculate several additional metrics that should help us find the answer to the above question

# make a copy of the dataframe with the columns of interest
df_ind = df[['label','hour','minute','mean_price','min_price', 'max_price']].copy()

# Moving forward we will be using mean price as the main metric and will base all calculations off of it
# for simplicity. It is possible to run a similar analysis using any other metric (e.g. min or max price):

# replace zero means with the first value of the day to leave the gap in vizualization
# instead of the chart dropping all the way to zero (so that it looks better)
df_ind['mean_price'].mask(df_ind['mean_price']==0, df_ind['mean_price'][0], inplace = True)

# calculate minute over minute difference in price
df_ind['diff_mom'] = df_ind['mean_price'] - df_ind.shift(1)['mean_price']

# calculate minute over minute PERCENT difference in price
df_ind['diff_mom_pct'] = df_ind['diff_mom']/df_ind.shift(1)['mean_price'] * 100.00

# calculate change since the opening of the market
df_ind['change_since_open'] = df_ind['mean_price'] - df_ind['mean_price'][0]

# create price index using price at 9:30 AM as zero
```

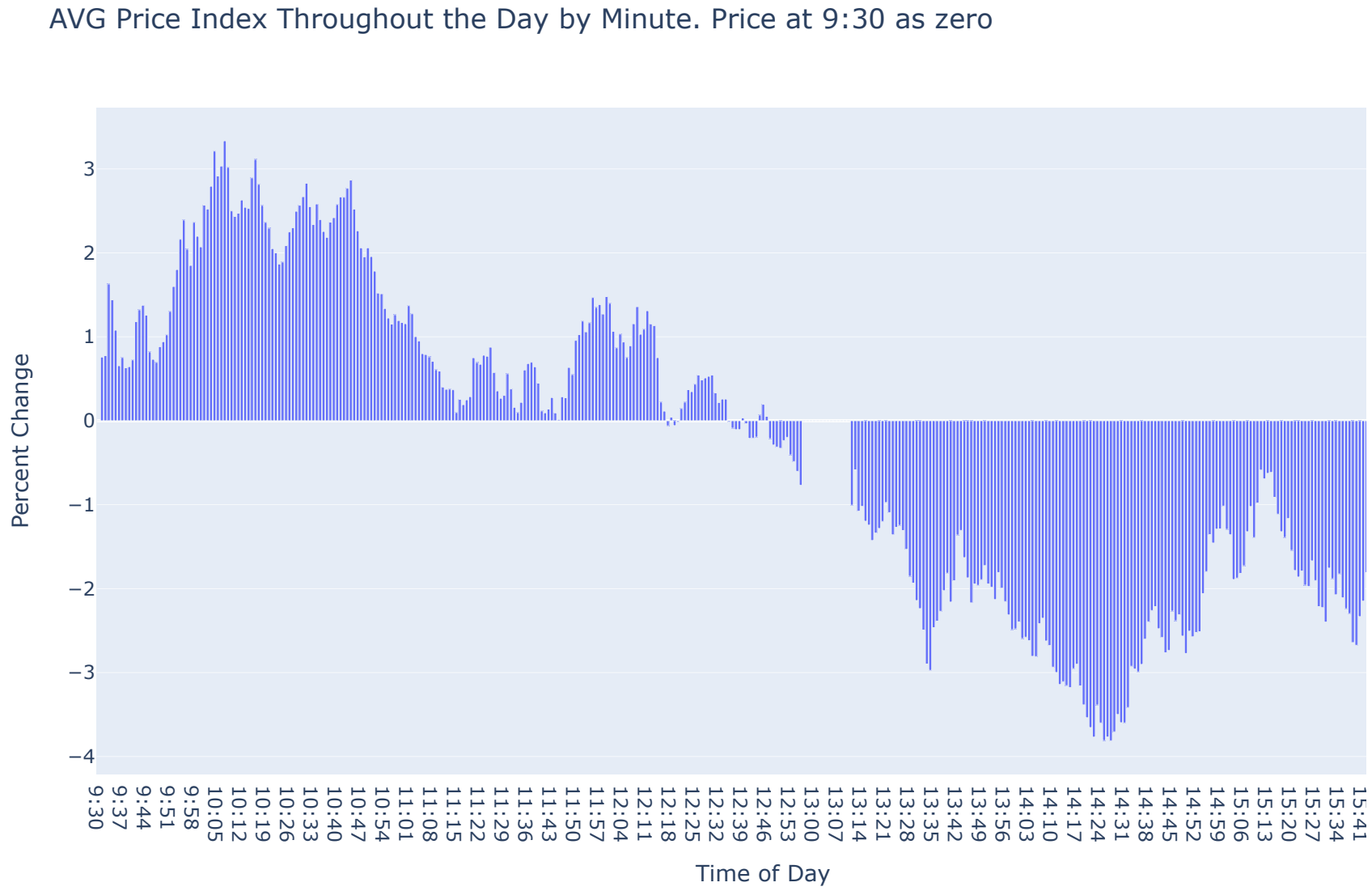
```
df_ind['price_index'] = df_ind['mean_price'] / df_ind['mean_price'][0] * 100.00 - 100

# calculate delta between min and max price for each minute
df_ind['delta_min_max'] = df_ind['max_price'] - df_ind['min_price']

# check the result
df_ind.head()
```

Out[9]:

	label	hour	minute	mean_price	min_price	max_price	diff_mom	diff_mom_pct	change_since_open	price_index	delta_min_max
0	9:30	9	30	237.652030	236.01	239.23	NaN	NaN	0.000000	0.000000	3.22
1	9:31	9	31	239.453419	236.03	240.68	1.801388	0.757994	1.801388	0.757994	4.65
2	9:32	9	32	239.495180	238.67	240.52	0.041762	0.017440	1.843150	0.775567	1.85
3	9:33	9	33	241.525011	239.44	243.54	2.029831	0.847545	3.872981	1.629685	4.10
4	9:34	9	34	241.074947	239.99	242.36	-0.450064	-0.186343	3.422916	1.440306	2.37

In [10]:

```
# Creating a visualization for the AVG price index
fig = px.bar(df_ind,
             x='label',
             y="price_index",
             title = "AVG Price Index Throughout the Day by Minute. Price at 9:30 as zero",
             width=1000, height=600)

fig.update_xaxes(title_text = "Time of Day",
                 title_font = {"size": 14})
fig.update_yaxes(title_text = "Percent Change",
                 title_font = {"size": 14})

fig.show()
```



ANSWER 1a: How much did the price change overall, and in what direction?

The price experienced significant fluctuations during the day, with periods of growth changing to the rapid fall stages, and only holding still while for several minutes on top of 15-min cycles we mentioned above (see details above in #1 observations).
Even though **the price was generally falling for the biggest part of the day**, it ended up rebounding and even gained almost 1.5% compared to the beginning of the trading day, see below for exact numbers:

In [11]:
```
# get the highest price index of the day
df_ind[df_ind['price_index'] == df_ind['price_index'].max()][['label','price_index']]

# by 10:07AM avg price (since we've been using that metric) grew by 3.33%
# however, as we could see on the chart above, the price started rapidly falling shortly after
```

Out[11]:

	label	price_index
37	10:07	3.334767

In [12]:
```
# get the lowest price index of the day
df_ind[df_ind['price_index'] == df_ind['price_index'].min()][['label','price_index']]

# and by 14:27, after almost 4 hours of falling, the price cratered to -3.81% compared to the opening price
```

Out[12]:

	label	price_index
297	14:27	-3.812291

Please note that the above metrics were calculated using the average price. If we were to use actual price in it momentary fluctuations the result will be even more pronounced with growth being even higher (5.16%) and the fall being lower (-5.55%), and a slightly shifted time (uncomment the below cell for the calculation):

In [13]:
```
# same metrics calculated on the original (non-resampled) dataset:
# calculate first and last price of the day
first = df4['PRICE'].iloc[0]
last = df4['PRICE'].iloc[-1]
# max growth
print(df4[df4['PRICE'] == df4['PRICE'].max()]['PRICE'] / first * 100 - 100)
# deepest drop
print(df4[df4['PRICE'] == df4['PRICE'].min()]['PRICE'] / last * 100 - 100)
```

```
Time
2022-12-01 10:14:08.845695    5.167598
Name: PRICE, dtype: float64
Time
2022-12-01 14:25:11.252546   -5.55141
Name: PRICE, dtype: float64
```

In [14]:
```
# aside from the above demonstration that the original data had higher momentary fluctuations,
# we will stick with the mean price as our main metric
```

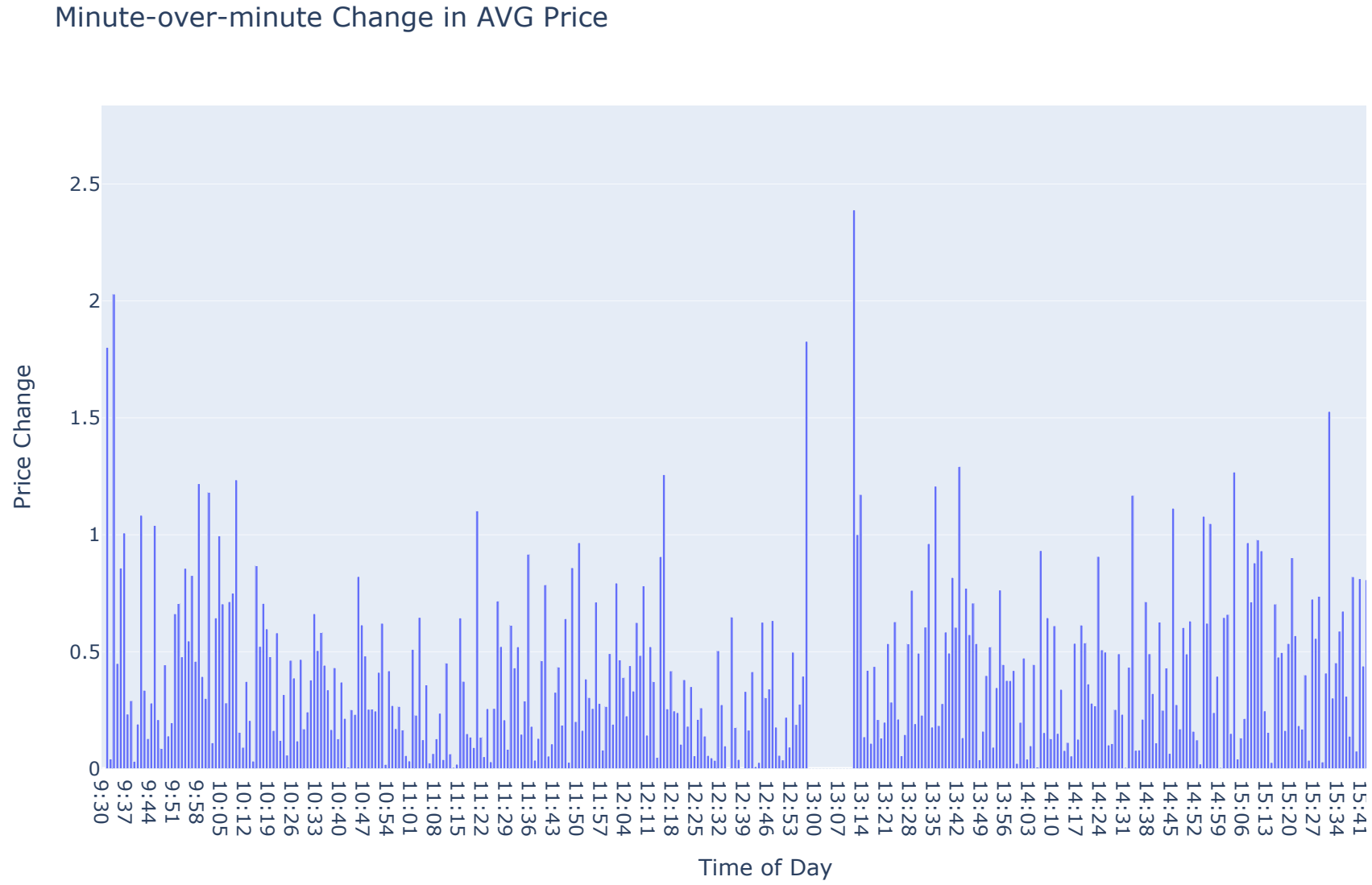
```
# show the percent change between the first and the last avg price of the day
first = df_ind['mean_price'].iloc[0]
last = df_ind['mean_price'].iloc[-1]

print(f'The average price increased by {last/first * 100 - 100:.2f}% at the end of the trading day')
```

```
The average price increased by 1.49% at the end of the trading day
```

Overall the average price increased by 1.49% at the end of the trading day

QUESTION 1b: When was the price changing fastest?

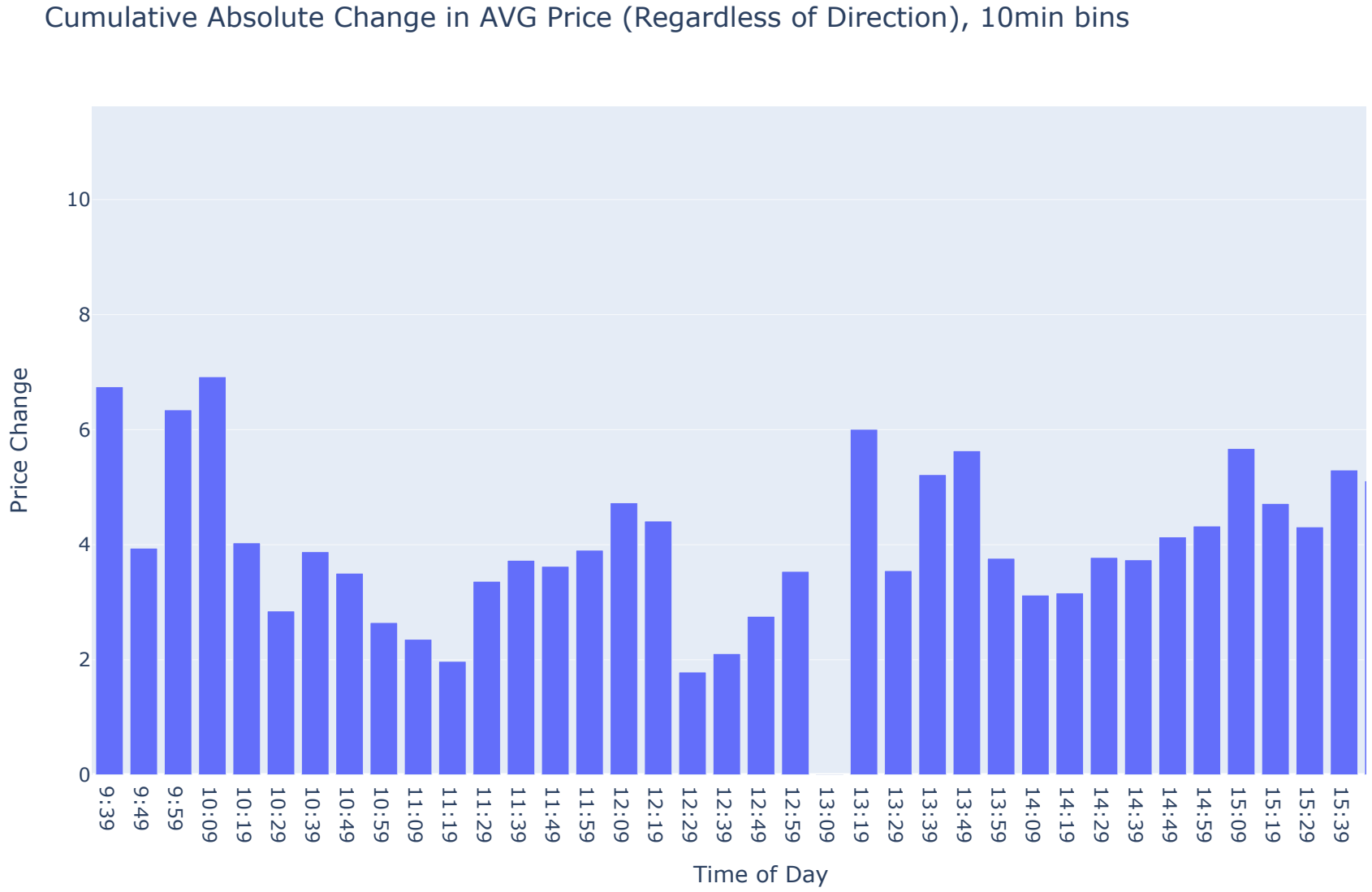
In [15]:
```
# make a copy of the dataframe with the below columns
# diff_mom: minute over minute difference in price
# diff_mom_pct: minute over minute PERCENT difference in price
# delta_min_max: delta between min and max price for each minute

fast = df_ind[['label','diff_mom','diff_mom_pct', 'delta_min_max']].copy()

# since we are only interested in the speed of change and not in direction,
# let's add the diff metrics as absolute values:
fast['diff_mom_abs'] = abs(fast['diff_mom'])
fast['diff_mom_pct_abs'] = abs(fast['diff_mom_pct'])
# fast.head()

# plot Minute over minute (absolute) change in price
fig = px.bar(fast,
             x='label',
             y="diff_mom_abs",
             title = "Minute-over-minute Change in AVG Price",
             width=1000, height=600)

fig.update_xaxes(title_text = "Time of Day",
                 title_font = {"size": 14})
fig.update_yaxes(title_text = "Price Change",
                 title_font = {"size": 14})

fig.show()
```



We could identify several areas of rapid change in price using the above graph but let's try resampling the data to a larger interval in order to remove some of the noise:

In [16]:
```
# make a copy
fast_10min = fast[['label','diff_mom','diff_mom_abs']].copy()

# get every 10th label starting from 10th
```

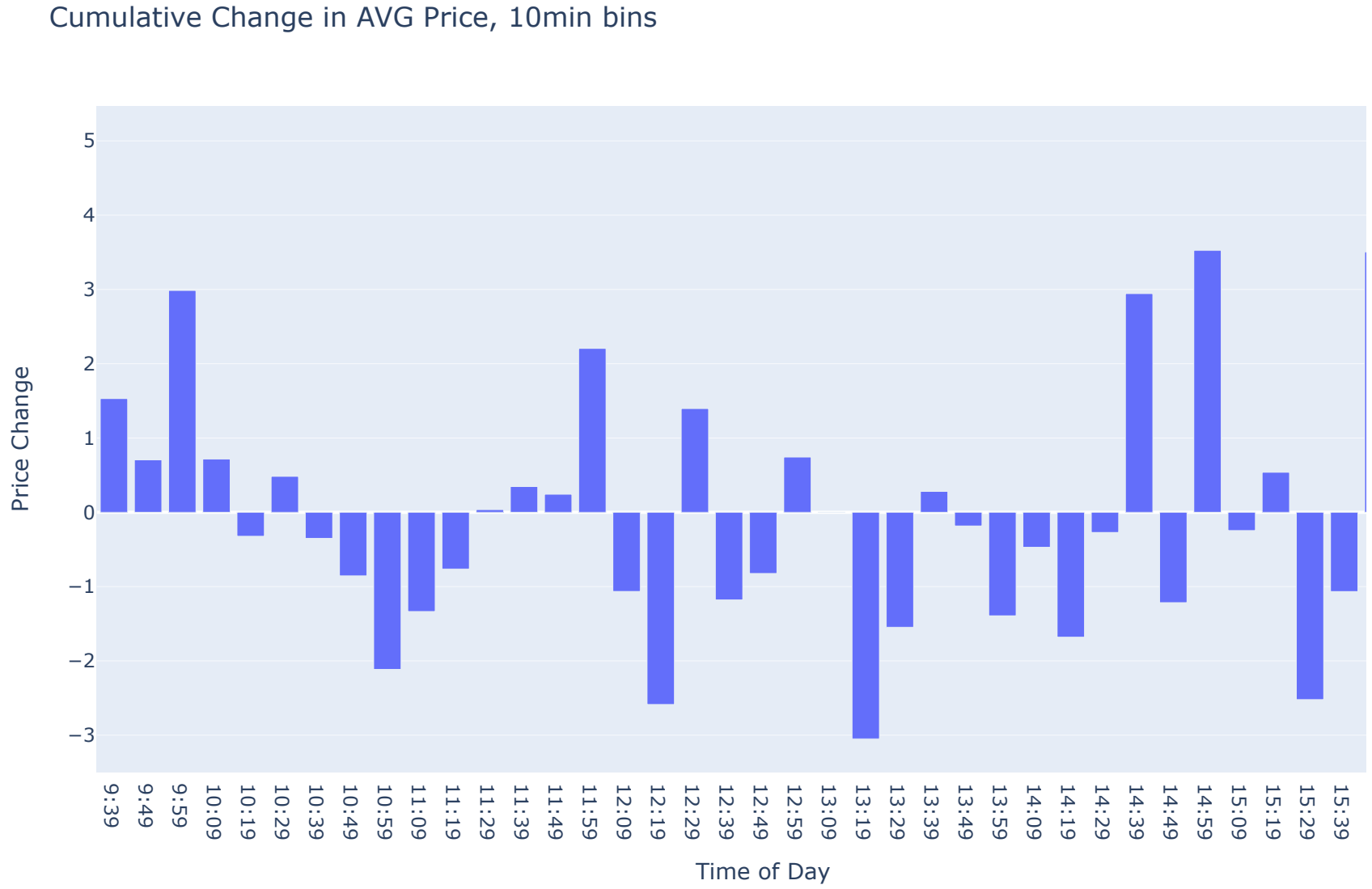
```
labels = fast_10min.iloc[9::10, 0].reset_index(drop = True)

# get rolling sum over 10min interval
fast_10min = fast_10min.groupby(fast_10min.index // 10).sum()

# concatenate two dataframes
fast_10min = pd.concat([labels, fast_10min], axis=1)
# fig = px.bar(fast_10min, x='label', y="diff_mom")
# fig.show()

# plot the result
fig = px.bar(fast_10min,
             x='label',
             y="diff_mom_abs",
             title = "Cumulative Absolute Change in AVG Price (Regardless of Direction), 10min bins",
             width=1000, height=600)

fig.update_xaxes(title_text = "Time of Day",
                 title_font = {"size": 14})
fig.update_yaxes(title_text = "Price Change",
                 title_font = {"size": 14})
fig.show()
```



ANSWER 1b

In the above chart we could see that in the past 10 minutes of the trading day the average price had changed the most when 11.04 cumulative change (in both directions) was recorded. Even if we were to try looking at a change in average price that takes into account direction of that change, last minutes of the day will still be leading, see below version of the same chart:

In [17]:
```
# Trying a similar plot but with cumulative change instead of absolute change
fig = px.bar(fast_10min,
             x='label',
             y="diff_mom",
             title = "Cumulative Change in AVG Price, 10min bins",
             width=1000, height=600)

fig.update_xaxes(title_text = "Time of Day",
                 title_font = {"size": 14})
fig.update_yaxes(title_text = "Price Change",
                 title_font = {"size": 14})
fig.show()
```

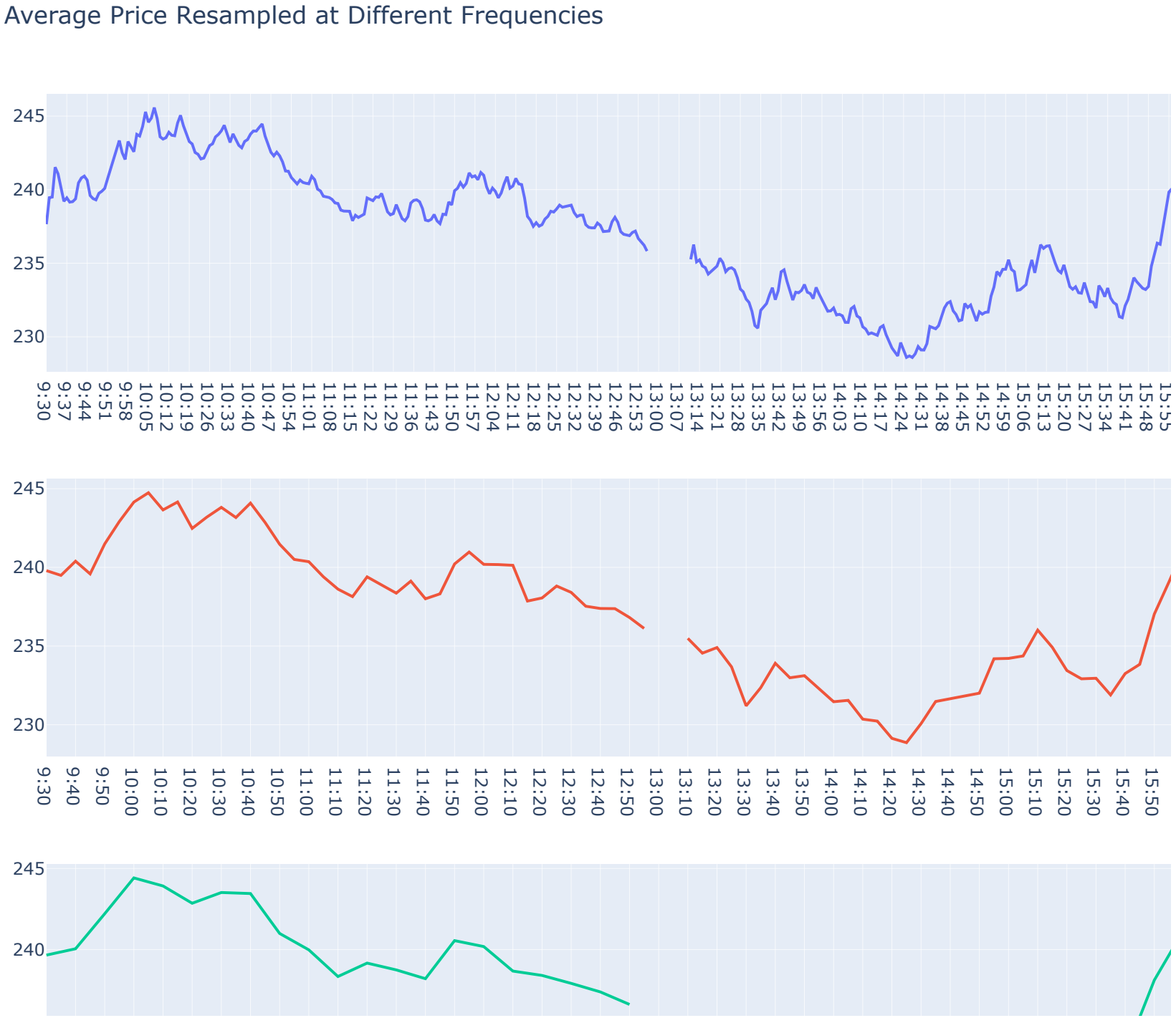


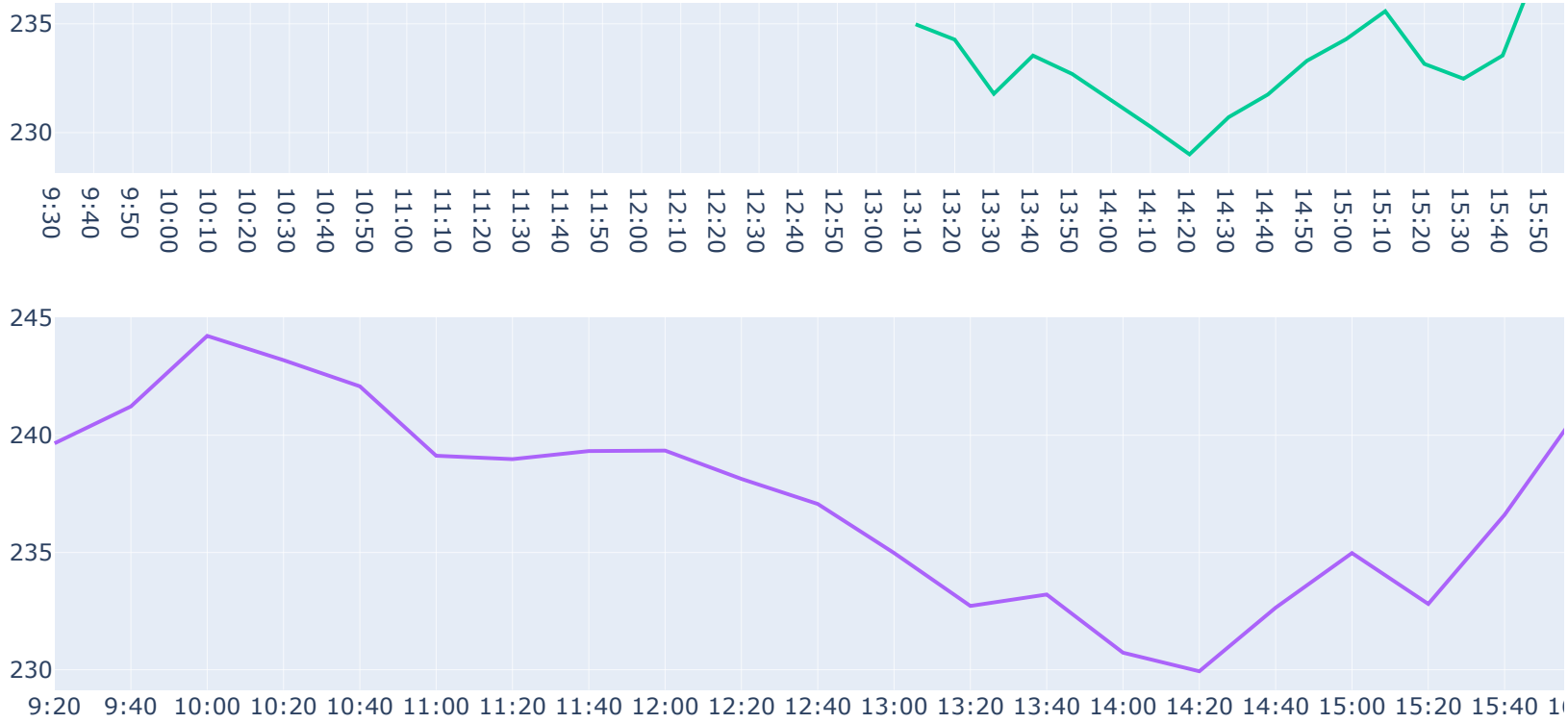
In the above version we could also see that most of the change (minute-over-minute difference) in average price happened at the end of the day (growth in this case).

QUESTION 1c: When was holding mostly steady (if at all)?

We will resample the data at different frequencies to make the fluctuations smoother:

In [18]:
```
fig = make_subplots(rows=4, cols=1)

# resample at 1 min
d = df4['PRICE'].resample('1min').agg({'mean_price': np.mean})
d['label'] = d.index.hour.astype(str) + ':' + d.index.strftime('%M')
d.reset_index(inplace = True, drop = True)

fig.add_trace(
    go.Scatter(x=d.label, y=d.mean_price, name = '1min'),
    row=1, col=1
)

# resample at 5 min
d = df4['PRICE'].resample('5min').agg({'mean_price': np.mean})
d['label'] = d.index.hour.astype(str) + ':' + d.index.strftime('%M')
d.reset_index(inplace = True, drop = True)

fig.add_trace(
    go.Scatter(x=d.label, y=d.mean_price, name = '5min'),
    row=2, col=1
)

# resample at 10 min
d = df4['PRICE'].resample('10min').agg({'mean_price': np.mean})
d['label'] = d.index.hour.astype(str) + ':' + d.index.strftime('%M')
d.reset_index(inplace = True, drop = True)

fig.add_trace(
    go.Scatter(x=d.label, y=d.mean_price, name = '10min'),
    row=3, col=1
)

# resample at 20 min
d = df4['PRICE'].resample('20min').agg({'mean_price': np.mean})
d['label'] = d.index.hour.astype(str) + ':' + d.index.strftime('%M')
d.reset_index(inplace = True, drop = True)

fig.add_trace(
    go.Scatter(x=d.label, y=d.mean_price, name = '20min'),
    row=4, col=1
)

fig.update_layout(height=1200, width=1000, title_text="Average Price Resampled at Different Frequencies")

fig.show()
```

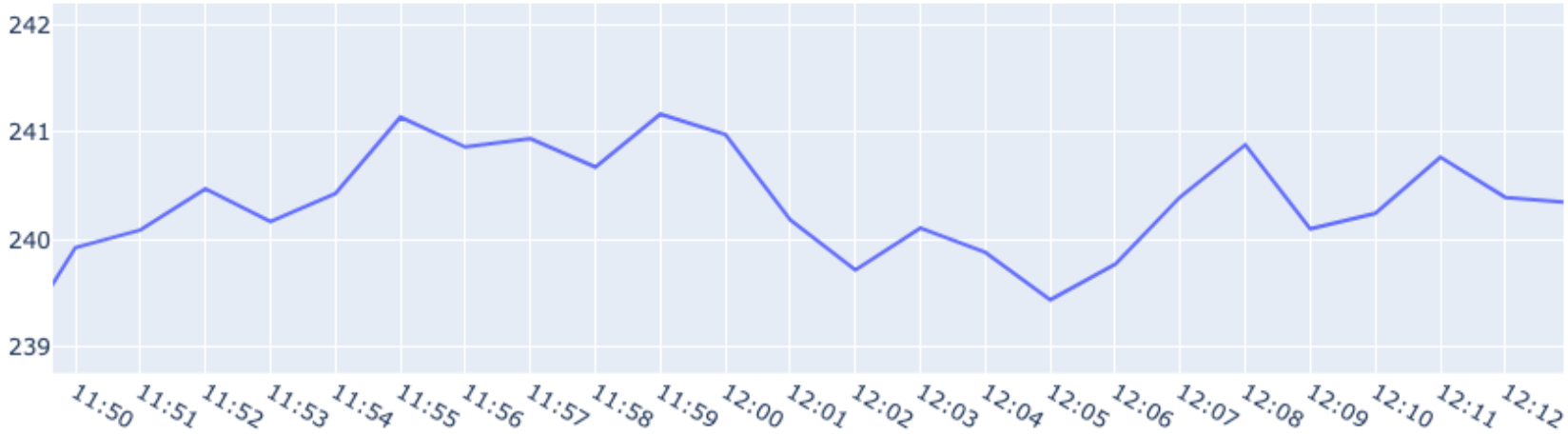

Average Price Resampled at Different Frequencies
245
240
235
230
9:30
9:37
9:44
9:51
9:58
10:05
10:12
10:19
10:26
10:33
10:40
10:47
10:54
11:01
11:08
11:15
11:22
11:29
11:36
11:43
11:50
11:57
12:04
12:11
12:18
12:25
12:32
12:39
12:46
12:53
13:00
13:07
13:14
13:21
13:28
13:35
13:42
13:49
13:56
14:03
14:10
14:17
14:24
14:31
14:38
14:45
14:52
14:59
15:06
15:13
15:20
15:27
15:34
15:41
15:48
15:55
245
240
235
230
9:30
9:40
9:50
10:00
10:10
10:20
10:30
10:40
10:50
11:00
11:10
11:20
11:30
11:40
11:50
12:00
12:10
12:20
12:30
12:40
12:50
13:00
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
14:30
14:40
14:50
15:00
15:10
15:20
15:30
15:40
15:50
245
240



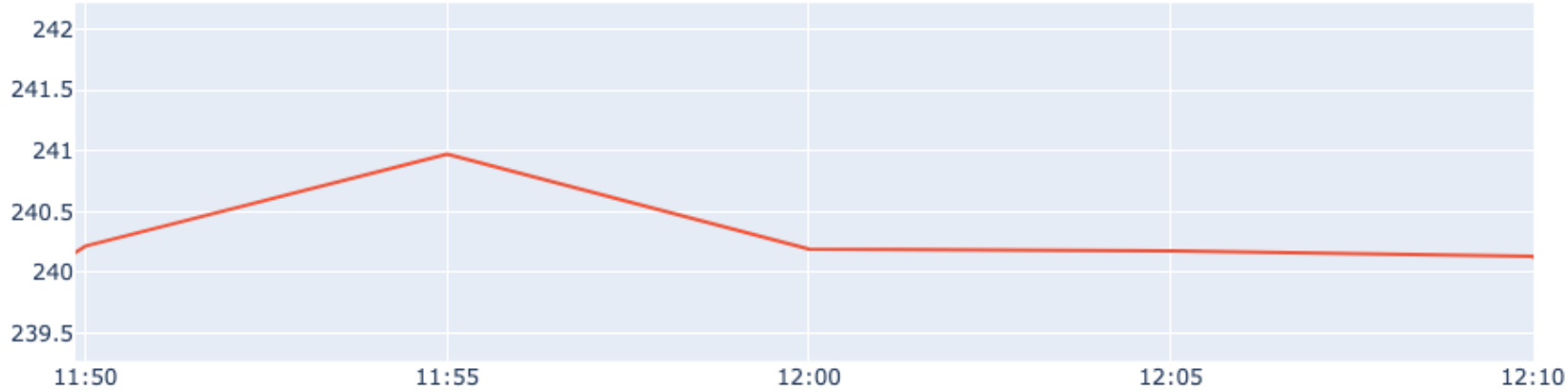
ANSWER 1c:

In the above charts we could see that the average price kept relatively steady for a short period of time, approximately from 11:50AM through 12:10PM, see zoomed in versions of chart at 1min and 5 min frequencies:

see the chart at 1 min frequency:



approximately the same period resampled at 5 min frequency:



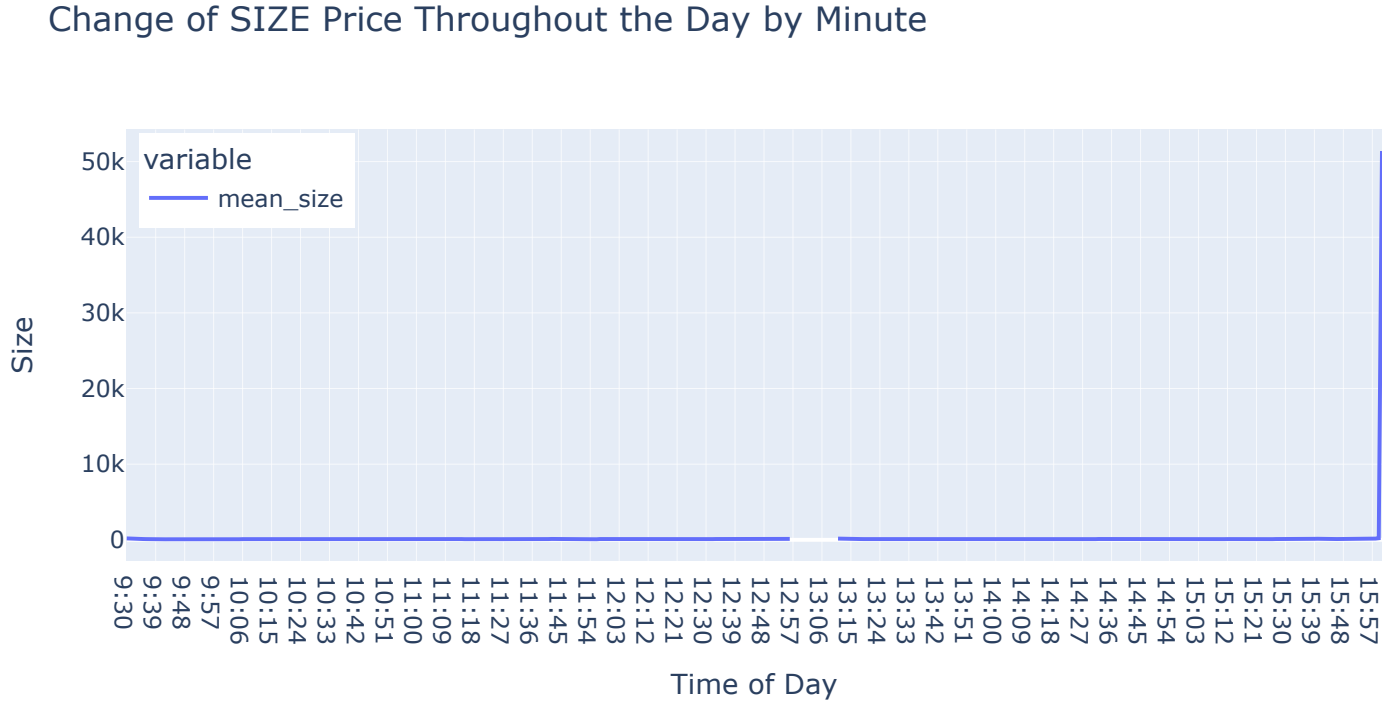
Please note that smoothed-out graphs above remove noise (frequent changes in avg price) somewhat artificially and in reality during that time we could see fluctuations in price (especially if we were to look at extreme metrics (min/max) as opposed to using the average (displayed above). So it is fair to say that if the price did keep relatively steady at all, it only happened for several minutes at a time between cyclical fluctuations we mentioned above.

QUESTION 2:

What happened to sizes of individual trades over the course of the day? Does trade size seem correlated with time of day at all in your opinion?

In [19]:
```
# resampling the data at 1 min frequency and calculating several metrics
size = df4['SIZE'].resample('1min').agg(min_size = 'min',
                                        max_size = 'max',
                                        mean_size = 'mean',
                                        sum_shares = 'sum',
                                        count_trades = 'size')

# adding a column for hours
size['hour'] = size.index.hour.astype(int)
#adding a column for minutes
size['minute'] = size.index.minute.astype(int)
#a dding a column for seconds
size['second'] = size.index.second.astype(int)

# adding count of minutes since midnight
size['minute_num'] = size.hour * 60 + size.minute
# adding count of minutes since midnight
size['second_num'] = size.hour * 60 + size.minute_num * 60 * 60 + size.second

#adding a label as a string for time series charts in 'HH:MM' format
size['label'] = size.hour.astype(str) + ':' + size.index.strftime('%M')

#resetting the index
size.reset_index(inplace = True)

# checking the result
# size.head()
```

In [20]:
```
fig = px.line(size,
              x='label',
              y=[size["mean_size"]],
             title = "Change of SIZE Price Throughout the Day by Minute",
             width=800, height=400)
# name axes
fig.update_xaxes(title_text = "Time of Day", title_font = {"size": 14})
fig.update_yaxes(title_text = "Size", title_font = {"size": 14})

# position the legend inside of the chart
fig.update_layout(
    legend=dict(
    yanchor="top",
    y=0.99,
    xanchor="left",
    x=0.01)
)
fig.show()
```



On the above chart we could see that the average size per minute drastically went up in the last minute of the trading day.

In [21]:
```
# checking the numbers in the dataframe with tail()
size.tail()
# in the very last minute of the trading day only 41 trades occured (compared to thousands of trades per minute
# however, the maximum size among those trades shot up to over 2M in once case, which dragged the avg all the wa
```

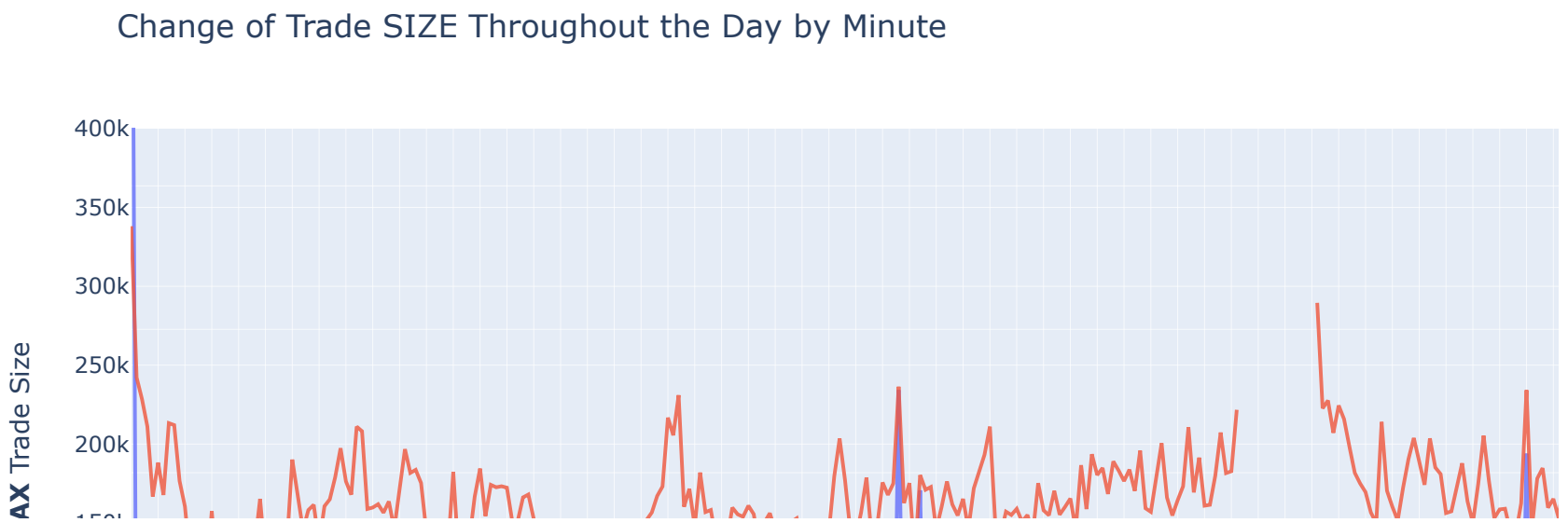
Out[21]:

	Time	min_size	max_size	mean_size	sum_shares	count_trades	hour	minute	second	minute_num	second_num	label
386	2022-12-01 15:56:00	1.0	15000.0	124.556771	3291039	26422	15	56	0	956	3442500	15:56
387	2022-12-01 15:57:00	1.0	22316.0	154.575883	3435449	22225	15	57	0	957	3446100	15:57
388	2022-12-01 15:58:00	1.0	14000.0	160.102215	3165541	19772	15	58	0	958	3449700	15:58
389	2022-12-01 15:59:00	1.0	19555.0	208.634076	4353150	20865	15	59	0	959	3453300	15:59
390	2022-12-01 16:00:00	10.0	2094942.0	51394.146341	2107160	41	16	0	0	960	3456960	16:00

To get a better picture of changes in sizes of individual trades throughout the day let's plot average sizes together with max sizes on one chart and limit the axes for better visualizations:

In [22]:

```
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter(x=size.label, y=size.max_size, name="max_size", opacity = 0.8),
              secondary_y=False)

fig.add_trace(go.Scatter(x=size.label, y=size.mean_size, name="mean_size", opacity = 0.8),
              secondary_y=True)

# Add title
fig.update_layout(title_text="Change of Trade SIZE Throughout the Day by Minute")
fig.update_yaxes(range=[0, 400000], secondary_y=False)
fig.update_yaxes(range=[0, 220], secondary_y=True)

# Set y-axes titles
fig.update_yaxes(title_text="<b>MAX</b> Trade Size", secondary_y=False)
fig.update_yaxes(title_text="<b>AVG</b> Trade Size", secondary_y=True)
```



ANSWER 2:

We could see that, on top of having a larger sizes of trades at the beginning of the trading day (as expected) and a huge spike in the last minutes of the trading day, there were significant fluctuations in trade sizes throughought the day. It will be interesting to look for potential correlations betwen sizes of individual trades and prices, so we will add it to a list of charts answering question 3.

In [23]:
```
# # getting top 10 time labels by number of trades
# top10_count_trades = size[['label', 'count_trades']]
# top10_count_trades = top10_count_trades.sort_values(by = 'count_trades', ascending = False)
```

```
# top10_count_trades = top10_count_trades.head(10)
# top10_count_trades.reset_index(inplace = True, drop = True)
# top10_count_trades
```

QUESTION 3:What happened to the rate of trading throughout the day?

What was the greatest number of trades per minute?

Per second?

How did the density of trading activity vary throughout the day?

What was the greatest number of trades per minute?

Identifying time of day to the minute with the highest number of trades

In [24]:
```
# getting the label for the max count of trades in HH:MM format
size[size['count_trades'] == size['count_trades'].max()][['label', 'count_trades','sum_shares']]
```

Out[24]:

	label	count_trades	sum_shares
384	15:54	26518	3349613

At 15:54 a minute with the highest number of trades was recorded: 26,518 individual transactions trading 3,349,613 shares

In [25]:
```
# zooming into a short period of time around the spike for ease of comutation
s = df4.between_time('15:55', '15:58')
s = s['SIZE'].resample('1s').agg(max_size = 'max', mean_size = 'mean', sum_shares = 'sum',count_trades = 'size')

#adding a label as a string for time series charts in 'HH:MM' format
s['label'] = s.index.hour.astype(str) + ':' + s.index.strftime('%M')+ ':' + s.index.strftime('%S')

#resetting the index
s.reset_index(inplace = True)

# getting the label for the max count of trades in HH:MM:SS format
s[s['count_trades'] == s['count_trades'].max()][['label', 'count_trades', 'sum_shares']]
```

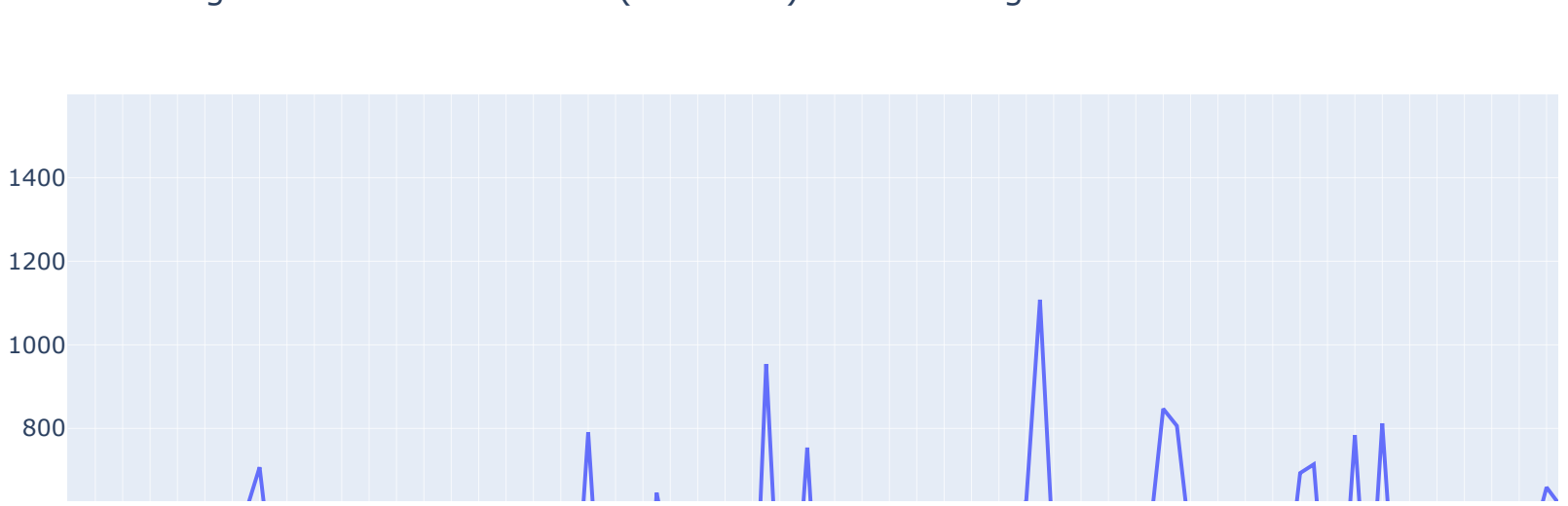
Out[25]:

	label	count_trades	sum_shares
110	15:56:50	1521	155218

At 15:56:50 a second with the highest number of trades was recorded: 1,521 individual transactions trading 155,218 shares

In [26]:

```
# show a chart to visualize the highest number of individual trades (1,521) which happened at 15:56:50
fig = go.Figure()
fig.add_trace(go.Scatter(x=s.label, y=s.count_trades, name="count_trades"))
fig.update_layout(title_text="Visualizing the time to the second (15:56:50) when the highest number of trades oc
fig.show()
```



QUESTION: How did the density of trading activity vary throughout the day?

We will gather several metrics we looked at above into one dataframe to plot them simultaneously

In [27]:
```
# make a copy of relevant price and size columns from earlier dataframes resampled to 1min
price_ = df[['label','min_price','max_price', 'mean_price']].copy()
size_ = size[['max_size', 'mean_size', 'sum_shares', 'count_trades']].copy()

# concatenate two dataframes
metrics = pd.concat([price_, size_], axis=1)
```

```
# checking the result
metrics.head()
```

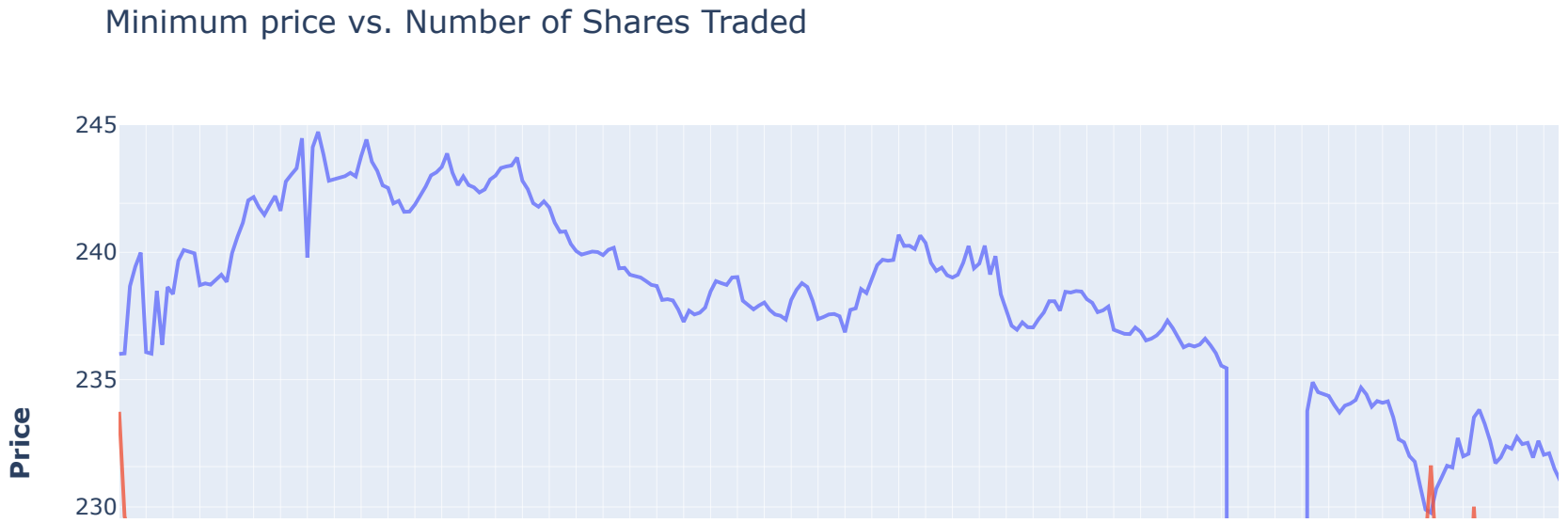
Out[27]:

	label	min_price	max_price	mean_price	max_size	mean_size	sum_shares	count_trades
0	9:30	236.01	239.23	237.652030	680087.0	185.944124	2415972	12993
1	9:31	236.03	240.68	239.453419	5000.0	133.186868	1620751	12169
2	9:32	238.67	240.52	239.495180	10100.0	125.615221	1477235	11760
3	9:33	239.44	243.54	241.525011	10300.0	116.106007	1387699	11952
4	9:34	239.99	242.36	241.074947	24403.0	91.716405	1051070	11460

Below we will explore "density of trading activity" using the number of shares traded throughout the day `sum_shares` or number of trades executed throughout the day `count_trades`

In [28]:

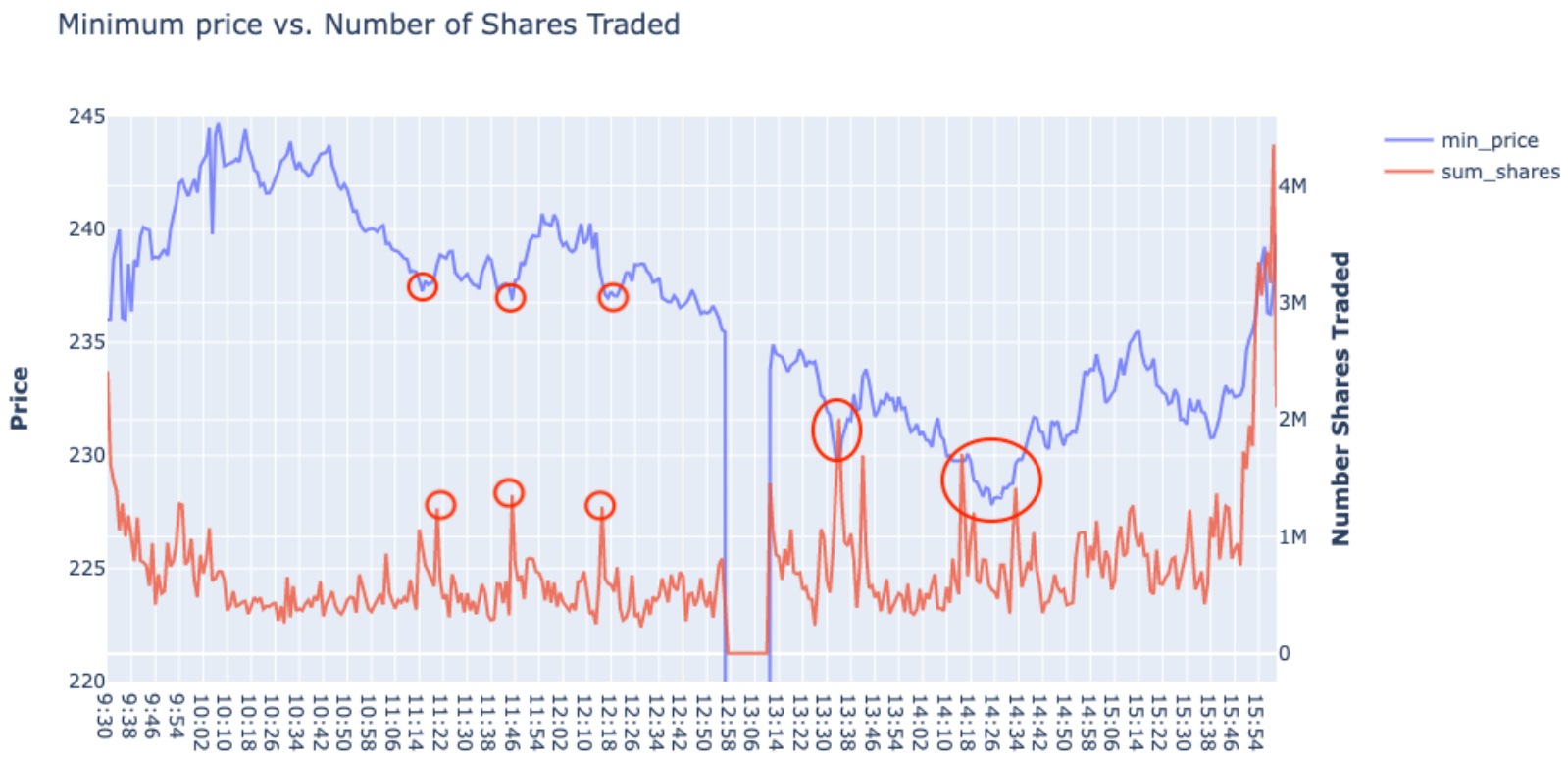
```
# plot more than one metric in one chart with two y axes

# Create a figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Minimum Price
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.min_price, name="min_price", opacity = 0.8),
              secondary_y=False)
# Number Shares Traded
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.sum_shares, name="sum_shares", opacity = 0.8),
              secondary_y=True)

# Add title
fig.update_layout(title_text="Minimum price vs. Number of Shares Traded")
fig.update_yaxes(range=[220, 245], secondary_y=False)
# fig.update_yaxes(range=[0, 40000], secondary_y=True)
# Set y-axes titles
fig.update_yaxes(title_text="<b>Price</b>", secondary_y=False)
fig.update_yaxes(title_text="<b>Number Shares Traded</b>", secondary_y=True)

fig.show()
```



OBSERVATIONS:

In the above chart we could see that on several occasions there werecases when at or around the time when the price was at a local minimum (see downward spikes at 11:15, 11:45, 12:15, 13:30, 14:15, 14:30 etc), simultaneosly, the number of shares traded at the same time skyrocketed (see local max on the red graph). It is also interesting that the above mentioned fluctuations were happening in a cyclical manner, occuring at approximately 15 min intervals all the way until the trading was stopped at 12:57 but reappering again after the trading was resumed (even if it was less pronounced).

Also, immediately after the trading was resumed at 13:11, while the price was still at a low point, the number of shares traded had also spiked.



In [29]:
```
# plot more than one metric in one chart with tho y axes

# Create a figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Minimum Price
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.min_price, name="min_price", opacity=0.8),
              secondary_y=False)
# Count of Trades
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.count_trades, name="count_trades", opacity=0.6),
              secondary_y=True)

# Add title
fig.update_layout(title_text="Minimum price vs. Count of Trades")
# fig.update_layout(yaxis_range=[200, 249], secondary_y=False)
fig.update_yaxes(range=[220, 248], secondary_y=False)
fig.update_yaxes(range=[0, 40000], secondary_y=True)

# Set y-axes titles
```

```
fig.update_yaxes(title_text="<b>Price</b>", secondary_y=False)
fig.update_yaxes(title_text="<b>Count of Trades</b>", secondary_y=True)

fig.show()
```

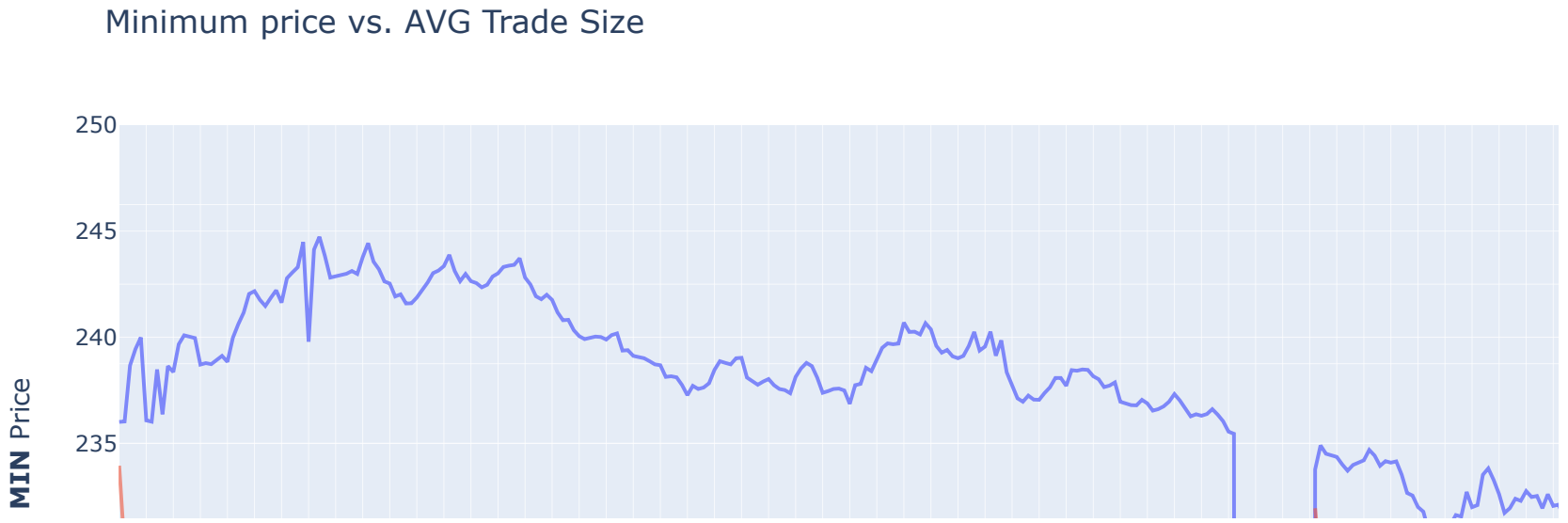
OBSERVATIONS:

In the above chart we could see a similar picture when spikes in the number of trades executed were happening at or around the time of price dropping to alocal minimum.

Look at the relationship between price and size of trades to answer a part of the question 2:

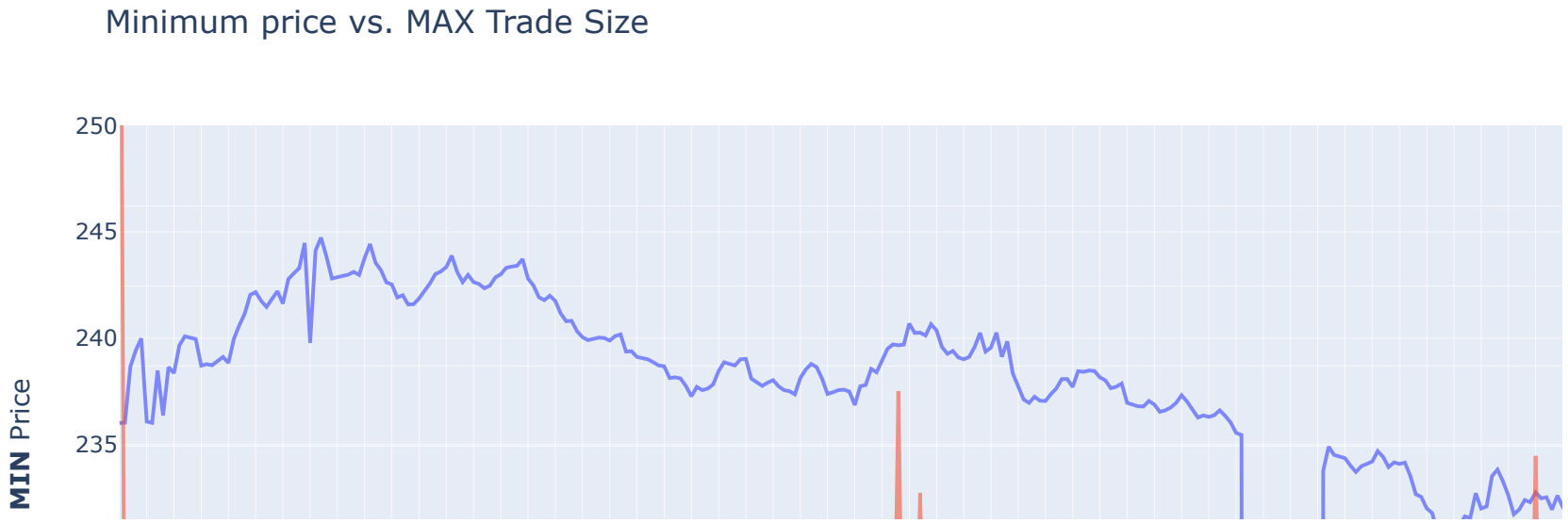
Does trade size seem correlated with time of day at all in your opinion?

In [30]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter(x=metrics.label, y=metrics.min_price, name="min_price", opacity=0.8),
              secondary_y=False)
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.mean_size, name="mean_size", opacity=0.6),
              secondary_y=True)

fig.update_layout(title_text="Minimum price vs. AVG Trade Size")
fig.update_yaxes(range=[220, 250], secondary_y=False)
# limit the secondary y-axis to be able to see the data outside of the big spike at the end of the day
fig.update_yaxes(range=[0, 400], secondary_y=True)

# Set y-axes titles
fig.update_yaxes(title_text="<b>MIN</b> Price", secondary_y=False)
fig.update_yaxes(title_text="<b>AVG</b> Size", secondary_y=True)

fig.show()
```



In [31]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter(x=metrics.label, y=metrics.min_price, name="min_price", opacity=0.8),
              secondary_y=False)
fig.add_trace(go.Scatter(x=metrics.label, y=metrics.max_size, name="max_size", opacity=0.6),
              secondary_y=True)

fig.update_layout(title_text="Minimum price vs. MAX Trade Size")
fig.update_yaxes(range=[220, 250], secondary_y=False)
# limit the secondary y-axis to be able to see the data outside of the big spike at the end of the day
fig.update_yaxes(range=[2000, 400000], secondary_y=True)
```

```
# Set y-axes titles
fig.update_yaxes(title_text="<b>MIN</b> Price", secondary_y=False)
fig.update_yaxes(title_text="<b>AVG</b> Size", secondary_y=True)

fig.show()
```

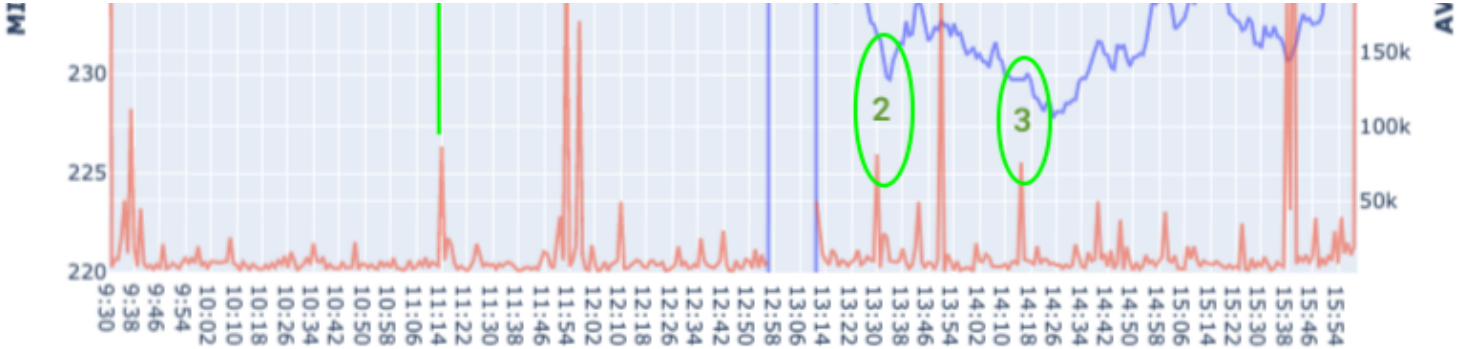


OBSERVATIONS:

The relationship between minimum price and the size of individual trades was not as pronounced as it was for the total number of shares traded or the number of transactions. We observe a correlation between moments when during the time of a significant loss

in price some of the largest increases in trade size had happened at 11:14, 13:30, 14:15, 15:40 (as you could see they also followed the 15-minute minimum price fluctuation to some extent).

Also, similar to the number of shares traded, immediately after the trading was resumed at 13:11, while the price was still at a low point, the size of individual trades had also spiked.


230
225
220
150k
100k
50k
2
3
9:30
9:38
9:46
9:54
10:02
10:10
10:18
10:26
10:34
10:42
10:50
10:58
11:06
11:14
11:22
11:30
11:38
11:46
11:54
12:02
12:10
12:18
12:26
12:34
12:42
12:50
12:58
13:06
13:14
13:22
13:30
13:38
13:46
13:54
14:02
14:10
14:18
14:26
14:34
14:42
14:50
14:58
15:06
15:14
15:22
15:30
15:38
15:46
15:54

In [ ]: