In [2]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

In [18]:

```python
df = pd.read_csv("result.csv")
```

In [19]:

```python
df
```

Out[19]:

	high	name	time
0	439.519897	SHOP	14
1	37.804901	BYND	9
2	118.993202	DDOG	11
3	118.526497	DDOG	13
4	198.600006	NFLX	9
...	...	...	...
65	29.440001	SNAP	10
66	104.219902	SQ	12
67	120.650002	DDOG	15
68	454.140015	SHOP	15
69	102.370003	SQ	13

70 rows × 3 columns

1) Highest Stock Price at the First Trading Hour (or Any Hour) (A Bar Chart: Each bar refers to a company)

2) Highest Hourly Stock Price Trend (A Line Chart: Each line refers to a company)

3) Comparison of Opening and Closing Price (A Grouped Bar Chart: Each group refers to a company and the bars refer to the opening and closing prices)

4) Average Highest Hourly Stock Price (A Bar Chart: Each bar refers to a company)

Highest Stock Price at the First Trading Hour (or Any Hour) (A Bar Chart: Each bar refers to a company)

In [10]:

```python
#filter time value by 9, since we filtered max high value for each trading hour that would be the answer for this
```

In [5]:

```python
result1_df = df[df['time'] == 9]
```

In [6]:

```python
result1_df
```

Out[6]:

	high	name	time
1	37.804901	BYND	9
4	198.600006	NFLX	9
7	210.729996	FB	9
18	61.224998	TTD	9
26	103.069901	SQ	9
34	123.339996	DDOG	9
53	29.540001	SNAP	9
60	123.349998	OKTA	9
62	21.240000	PINS	9
64	451.000000	SHOP	9

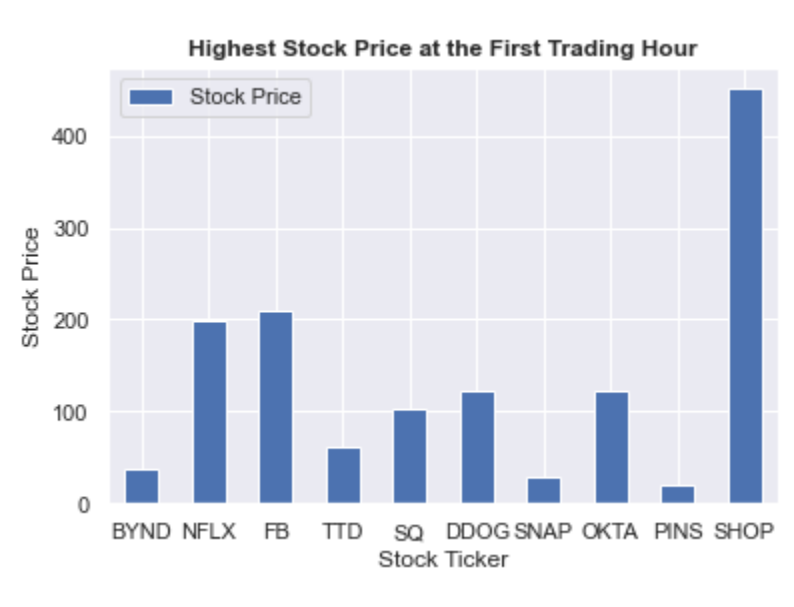
In [ ]:

In [7]:

```python
ax = result1_df.plot.bar(x='name', y='high', rot=0,legend=True)
plt.legend(["Stock Price"])
plt.xlabel("Stock Ticker")
plt.ylabel("Stock Price")
plt.title("Highest Stock Price at the First Trading Hour",fontweight="bold")
```

Out[7]:

```
Text(0.5, 1.0, 'Highest Stock Price at the First Trading Hour')
```



Highest Hourly Stock Price Trend (A Line Chart: Each line refers to a company)

In [20]:

```python
df
```

Out[20]:

	high	name	time
0	439.519897	SHOP	14
1	37.804901	BYND	9
2	118.993202	DDOG	11
3	118.526497	DDOG	13
4	198.600006	NFLX	9
...	...	...	...
65	29.440001	SNAP	10
66	104.219902	SQ	12
67	120.650002	DDOG	15
68	454.140015	SHOP	15
69	102.370003	SQ	13

70 rows × 3 columns

In [137…

```python
df1 = df[df['name'] == "SHOP"]
df1 = df1.set_index('time')
df2 = df[df['name'] == "NFLX"]
df2 = df2.set_index('time')
df3 = df[df['name'] == "FB"]
df3 = df3.set_index('time')
df4 = df[df['name'] == "TTD"]
df4 = df4.set_index('time')
df5 = df[df['name'] == "SQ"]
df5 = df5.set_index('time')
df6 = df[df['name'] == "DDOG"]
df6 = df6.set_index('time')
df7 = df[df['name'] == "SNAP"]
df7 = df7.set_index('time')
df8 = df[df['name'] == "OKTA"]
df8 = df8.set_index('time')
df9 = df[df['name'] == "PINS"]
df9 = df9.set_index('time')
```

In [138…

```python
df1 = df1.drop('name', 1)
df1 = df1.rename(columns={"high": "SHOP"})
```

```
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1956832412.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df1 = df1.drop('name', 1)
```

In [40]:

```python
df1
```

Out[40]:

time	SHOP
14	439.519897
12	445.989990
11	446.100006
13	440.204987
10	448.420013
9	451.000000
15	454.140015

In [139…

```python
df2 = df2.drop('name', 1)
df2 = df2.rename(columns={"high": "NFLX"})
```

```
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1187464798.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df2 = df2.drop('name', 1)
```

In [42]:

```python
df2
```

Out[42]:

time	NFLX
9	198.600006
13	196.089996
14	196.599899
11	199.449799
12	199.439896
15	199.660004
10	200.214096

In [140…

```python
df3 = df3.drop('name', 1)
df3 = df3.rename(columns={"high": "FB"})
df4 = df4.drop('name', 1)
df4 = df4.rename(columns={"high": "TTD"})
df5 = df5.drop('name', 1)
df5 = df5.rename(columns={"high": "SQ"})
df6 = df6.drop('name', 1)
df6 = df6.rename(columns={"high": "DDOG"})
df7 = df7.drop('name', 1)
df7 = df7.rename(columns={"high": "SNAP"})
df8 = df8.drop('name', 1)
df8 = df8.rename(columns={"high": "OKTA"})
df9 = df9.drop('name', 1)
df9 = df9.rename(columns={"high": "PINS"})
```

```
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df3 = df3.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df4 = df4.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:5: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df5 = df5.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:7: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df6 = df6.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:9: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df7 = df7.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:11: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df8 = df8.drop('name', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1969411820.py:13: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df9 = df9.drop('name', 1)
```

In [ ]:

In [142…

```python
df0 = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9], axis=1)
```

In [84]:

```python
df0
```

Out[84]:

time	SHOP	NFLX	FB	TTD	SQ	DDOG	SNAP	OKTA	PINS
9	451.000000	198.600006	210.729996	61.224998	103.069901	123.339996	29.540001	123.349998	21.240000
10	448.420013	200.214096	210.860001	61.349998	103.278999	122.578903	29.440001	123.320000	21.365000
11	446.100006	199.449799	209.429993	60.639999	102.439903	118.993202	29.235001	121.760002	21.410000
12	445.989990	199.439896	207.570007	60.979900	104.219902	120.709999	29.440001	123.845001	21.400000
13	440.204987	196.089996	204.809998	59.939999	102.370003	118.526497	28.969999	121.695099	21.219999
14	439.519897	196.599899	205.389999	60.279999	102.974998	118.309998	29.014999	121.879997	21.490000
15	454.140015	199.660004	211.880005	62.119999	105.970001	120.650002	29.540001	124.004997	22.275000

In [ ]:

In [ ]:

In [ ]:

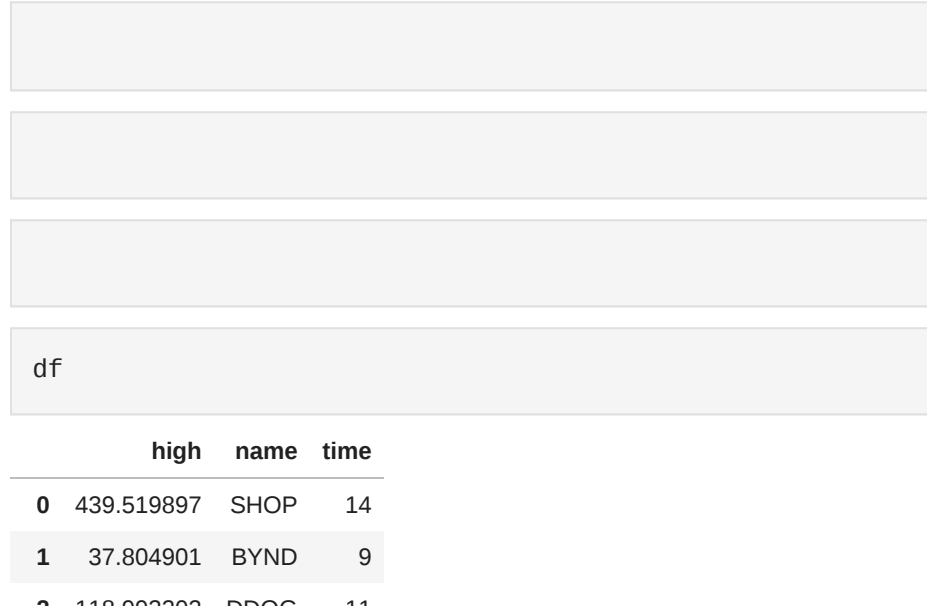
In [ ]:

In [72]:

```python
plt.style.use('seaborn-darkgrid')
palette = plt.get_cmap('Set1')
num=0
for column in df0:
    num+=1
    plt.plot(df0[column], marker='', color=palette(num), linewidth=1, alpha=0.9, label=column)
#add legends

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))

# Add titles
plt.title("Highest Hourly Stock Price Trend", loc='center', fontsize=15, fontweight=0, color='black')
plt.xlabel("Stock Price")
plt.ylabel("Market Hour")

#https://stackoverflow.com/questions/4700614/how-to-put-the-legend-outside-the-plot-in-matplotlib
#for legends location future purpose
```

Out[72]:

```
Text(0, 0.5, 'Market Hour')
```



Comparison of Opening and Closing Price (A Grouped Bar Chart: Each group refers to a company and the bars refer to the opening and closing prices)

In [ ]:

In [85]:

```python
#df0.drop(df0.index[1:6], inplace=True)
#Dropping Method
```

In [ ]:

In [ ]:

In [ ]:

In [91]:

```python
df
```

Out[91]:

	high	name	time
0	439.519897	SHOP	14
1	37.804901	BYND	9
2	118.993202	DDOG	11
3	118.526497	DDOG	13
4	198.600006	NFLX	9
...	...	...	...
65	29.440001	SNAP	10
66	104.219902	SQ	12
67	120.650002	DDOG	15
68	454.140015	SHOP	15
69	102.370003	SQ	13

70 rows × 3 columns

In [130…

```python
#time = column, index = company name
df11 = df[df['time'] == 9]
df11 = df11.set_index('name')
df22 = df[df['time'] == 10]
df22 = df22.set_index('name')
df33 = df[df['time'] == 11]
df33 = df33.set_index('name')
df44 = df[df['time'] == 12]
df44 = df44.set_index('name')
df55 = df[df['time'] == 13]
df55 = df55.set_index('name')
df66 = df[df['time'] == 14]
df66 = df66.set_index('name')
df77 = df[df['time'] == 15]
df77 = df77.set_index('name')
#df88 = df[df['time'] == 16]
#df88 = df88.set_index('time')
```

In [131…

```python
df11 = df11.drop('time', 1)
df11 = df11.rename(columns={"high": 9})
df22 = df22.drop('time', 1)
df22 = df22.rename(columns={"high": 10})
df33 = df33.drop('time', 1)
df33 = df33.rename(columns={"high": 11})
df44 = df44.drop('time', 1)
df44 = df44.rename(columns={"high": 12})
df55 = df55.drop('time', 1)
df55 = df55.rename(columns={"high": 13})
df66 = df66.drop('time', 1)
df66 = df66.rename(columns={"high": 14})
df77 = df77.drop('time', 1)
df77 = df77.rename(columns={"high": 15})
#df88 = df88.drop('time', 1)
#df88 = df88.rename(columns={"high": "16"})
```

```
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df11 = df11.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df22 = df22.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:5: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df33 = df33.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:7: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df44 = df44.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:9: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df55 = df55.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:11: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df66 = df66.drop('time', 1)
C:\Users\hlee0\AppData\Local\Temp/ipykernel_3392/1061918911.py:13: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  df77 = df77.drop('time', 1)
```

In [132…

```python
df00 = pd.concat([df11, df77], axis=1)
```

In [133…

```python
df00
```

Out[133…

name	9	15
BYND	37.804901	38.330002
NFLX	198.600006	199.660004
FB	210.729996	211.880005
TTD	61.224998	62.119999
SQ	103.069901	105.970001
DDOG	123.339996	120.650002
SNAP	29.540001	29.540001
OKTA	123.349998	124.004997
PINS	21.240000	22.275000
SHOP	451.000000	454.140015

In [120…

```python
type(df00)
```

Out[120…

```
pandas.core.frame.DataFrame
```

In [ ]:

In [134…

```python
df00.plot(kind="bar")
plt.title("Comparison of Opening and Closing Price")
plt.xlabel("Stock Price")
plt.ylabel("Stock Tickers")
```

Out[134…

```
Text(0, 0.5, 'Stock Tickers')
```

Average Highest Hourly Stock Price (A Bar Chart: Each bar refers to a company)

In [143…

```python
df0
```

Out[143…

time	SHOP	NFLX	FB	TTD	SQ	DDOG	SNAP	OKTA	PINS
9	451.000000	198.600006	210.729996	61.224998	103.069901	123.339996	29.540001	123.349998	21.240000
10	448.420013	200.214096	210.860001	61.349998	103.278999	122.578903	29.440001	123.320000	21.365000
11	446.100006	199.449799	209.429993	60.639999	102.439903	118.993202	29.235001	121.760002	21.410000
12	445.989990	199.439896	207.570007	60.979900	104.219902	120.709999	29.440001	123.845001	21.400000
13	440.204987	196.089996	204.809998	59.939999	102.370003	118.526497	28.969999	121.695099	21.219999
14	439.519897	196.599899	205.389999	60.279999	102.974998	118.309998	29.014999	121.879997	21.490000
15	454.140015	199.660004	211.880005	62.119999	105.970001	120.650002	29.540001	124.004997	22.275000

In [144…

```python
#add row as mean
df0.loc['mean'] = df0.mean()
```

In [145…

```python
df0
```

Out[145…

time	SHOP	NFLX	FB	TTD	SQ	DDOG	SNAP	OKTA	PINS
9	451.000000	198.600006	210.729996	61.224998	103.069901	123.339996	29.540001	123.349998	21.240000
10	448.420013	200.214096	210.860001	61.349998	103.278999	122.578903	29.440001	123.320000	21.365000
11	446.100006	199.449799	209.429993	60.639999	102.439903	118.993202	29.235001	121.760002	21.410000
12	445.989990	199.439896	207.570007	60.979900	104.219902	120.709999	29.440001	123.845001	21.400000
13	440.204987	196.089996	204.809998	59.939999	102.370003	118.526497	28.969999	121.695099	21.219999
14	439.519897	196.599899	205.389999	60.279999	102.974998	118.309998	29.014999	121.879997	21.490000
15	454.140015	199.660004	211.880005	62.119999	105.970001	120.650002	29.540001	124.004997	22.275000
mean	446.482130	198.579099	208.667143	60.933556	103.474815	120.444085	29.311429	122.836442	21.485714

In [146…

```python
df_try = df0['mean'] = df0.mean()
```

In [148…

```python
#returns mean
df_try
```

Out[148…

```
SHOP    446.482130
NFLX    198.579099
FB      208.667143
TTD      60.933556
SQ      103.474815
DDOG    120.444085
SNAP     29.311429
OKTA    122.836442
PINS     21.485714
dtype: float64
```

In [149…

```python
type(df_try)
```

Out[149…

```
pandas.core.series.Series
```

In [151…

```python
ax = df_try.plot.bar(x='Stock Tickers', y='Average Stock Price',title = "Average Highest Hourly Stock Price", rot=0)
```


In [ ]: