Introduction:

The purpose of this project is to analyze the stock price data of ***Bharat Petroleum Corporation Limited (BPCL)*** using Python. The data will be retrieved from data.nasdaq.com API and plotted using a plotting library such as Matplotlib or Seaborn.

Technology Requirements:

The Jupyter Notebook should be written in Python programming language. The stock price data should be retrieved using **`data.nasdaq.com API.`** The line graph should be generated using a plotting library such as Matplotlib or Seaborn.

Deliverables:

A completed Jupyter Notebook that meets the technology requirements outlined above. A brief summary of the findings and observations from the analysis of BPCL stock price data. All source code and supporting documentation necessary to run the Jupyter Notebook on a computer with Python and the required libraries installed. Instructions for Running the Jupyter Notebook:

Install the latest version of Python on your computer. Install the required libraries for the project, including Jupyter Notebook, Matplotlib or Seaborn, and any other libraries required for API data retrieval. Download the project files from the source provided. Open the Jupyter Notebook file and run the cells one by one. The line graph of BPCL stock price data will be generated and displayed in the Notebook. Expected Outcome: After following the instructions above and running the Jupyter Notebook, a line graph of BPCL stock price data will be generated and displayed. The graph will show the trend of BPCL stock prices over time and provide insights into the stock performance.

Conclusion:

The project is a comprehensive analysis of BPCL stock price data using Python programming language, data.nasdaq.com API, and a plotting library such as Matplotlib or Seaborn. The deliverables of the project include a completed Jupyter Notebook, a brief summary of the findings, and all the source code and supporting documentation necessary to run the Notebook. By following the instructions and running the Notebook, you can generate a line graph of BPCL stock price data and gain valuable insights into the stock performance.

In [ ]:

In [1]:
```
import quandl
import pandas
from matplotlib import pyplot as plt , dates as mdates
from datetime import datetime
import seaborn as sns
from dateutil.relativedelta import relativedelta
from scipy.stats import zscore
import config
import numpy as np
# pandas.set_option('display.max_rows' , None)
```

Please paste you api_key in config.py file

Just for the sake of simplicity environment variable or dotenv api encryption is not done

In [2]:
```
#This is the value of our api key
API_KEY= config.api_key
```

In [3]:
```
# every single stock has its own unique code and while we fetch our data this is the
BPCL_code = 'BSE/BOM500547'
```

In [4]:
```
#Here we are getting the values for the start date and the end date for which we wan
#As mentioned in the requirements we need the data for the past 5 years so our start
CURRENT_DATE = datetime.now()
START_DATE = CURRENT_DATE - relativedelta(years = 25)
START_DATE = datetime.strftime(START_DATE , '%Y-%m-%d')
CURRENT_DATE = datetime.strftime(CURRENT_DATE, '%Y-%m-%d')
START_DATE , CURRENT_DATE
```

Out[4]:
```
('1998-02-03', '2023-02-03')
```

In [5]:
```
quandl.ApiConfig.api_key = API_KEY
bpcl_df = quandl.get(BPCL_code, start_date= START_DATE, end_date=CURRENT_DATE, colla
```

In [6]:
```
bpcl_df.tail()
```

Out[6]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliverable Quantity	% Deli. Qty to Traded Qty	Spr
2023-01-27	346.00	349.35	331.75	336.70	338.32	73625.0	2589.0	24908761.0	24956.0	33.90	1
2023-01-30	336.10	339.30	332.35	335.05	335.26	169164.0	3968.0	56713442.0	44601.0	26.37	
2023-01-31	341.90	351.50	341.65	343.25	345.63	174047.0	4448.0	60156085.0	58483.0	33.60	
2023-02-01	340.05	344.65	330.10	334.20	337.86	184604.0	6247.0	62370305.0	50522.0	27.37	1
2023-02-02	331.75	336.05	329.60	331.90	332.76	63298.0	2197.0	21063014.0	22438.0	35.45	

In [7]:
```
bpcl_df.head()
```

Out[7]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliverable Quantity	% Deli. Qty to Traded Qty
1998-02-03	348.0	349.75	340.00	340.5	340.972460	93500.0	77.0	31880925.0	NaN	NaN
1998-02-04	345.0	363.00	342.00	354.5	346.311831	186800.0	185.0	64691050.0	NaN	NaN
1998-02-05	356.0	364.00	350.00	360.0	359.638643	33900.0	86.0	12191750.0	NaN	NaN
1998-02-10	351.5	352.00	346.00	350.0	349.182384	56200.0	29.0	19624050.0	NaN	NaN
1998-02-11	350.0	354.00	347.75	350.0	351.112952	16600.0	17.0	5828475.0	NaN	NaN

In [8]:

```
bpcl_df.isnull().sum()
```

Out[8]:

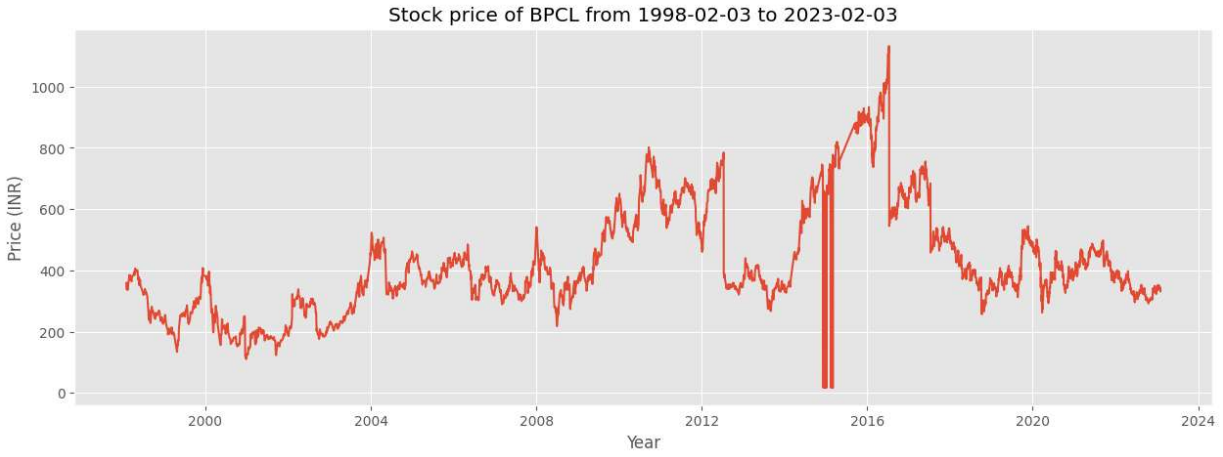
```
Open                         0
High                         0
Low                          0
Close                        0
WAP                          0
No. of Shares                0
No. of Trades                0
Total Turnover               0
Deliverable Quantity       461
% Deli. Qty to Traded Qty  461
Spread H-L                   0
Spread C-O                   0
dtype: int64
```

Task 1

1. Plot the raw stock prices over time.

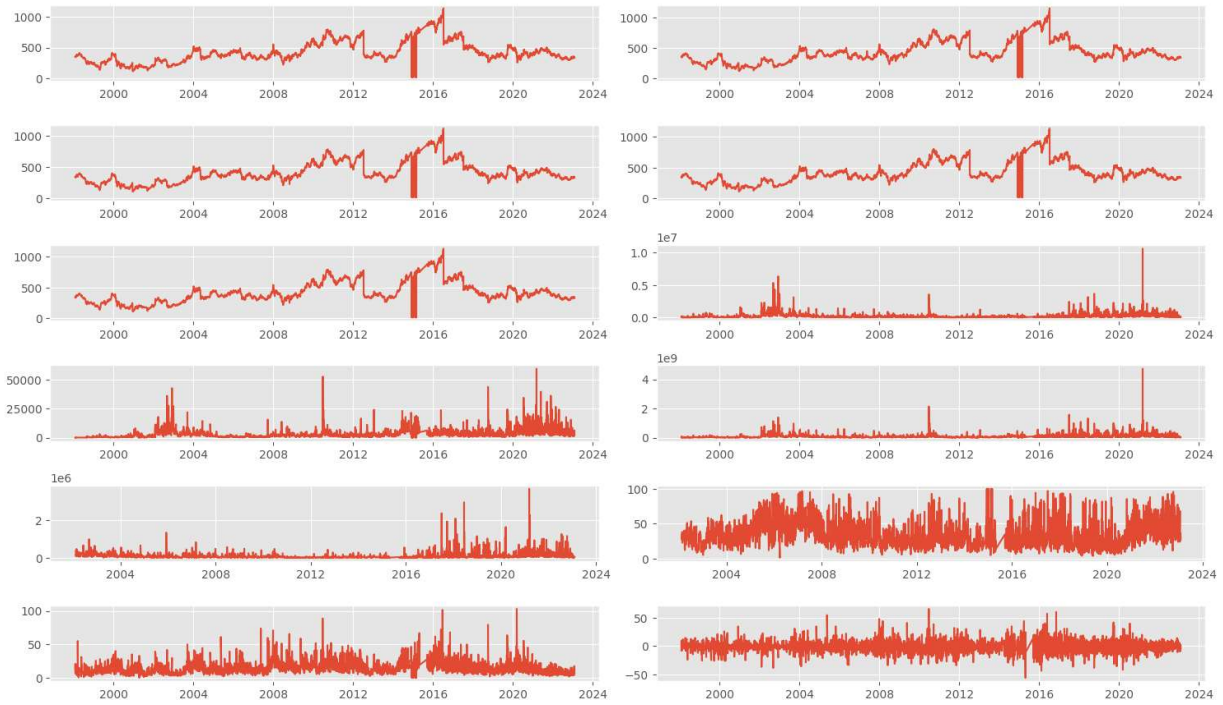
In [9]:

```
plt.style.use('ggplot')
plt.figure(figsize =[15,5] )
plt.plot(bpcl_df.index , bpcl_df['Close']  , linewidth = 1.5)
plt.xlabel('Year')
plt.ylabel('Price (INR)')
# plt.grid(visible = True)
plt.title(f'Stock price of BPCL from {START_DATE} to {CURRENT_DATE}')
plt.show()
```



In [10]:

```
plt.figure(figsize = [15,10])
position = 1
for column in bpcl_df.columns:
    plt.subplot(7,2, position)
    plt.plot(bpcl_df[column])
    position+=1
    plt.tight_layout()
plt.show()
```



Nothing conclusive could be said so let's try fetching transformed data

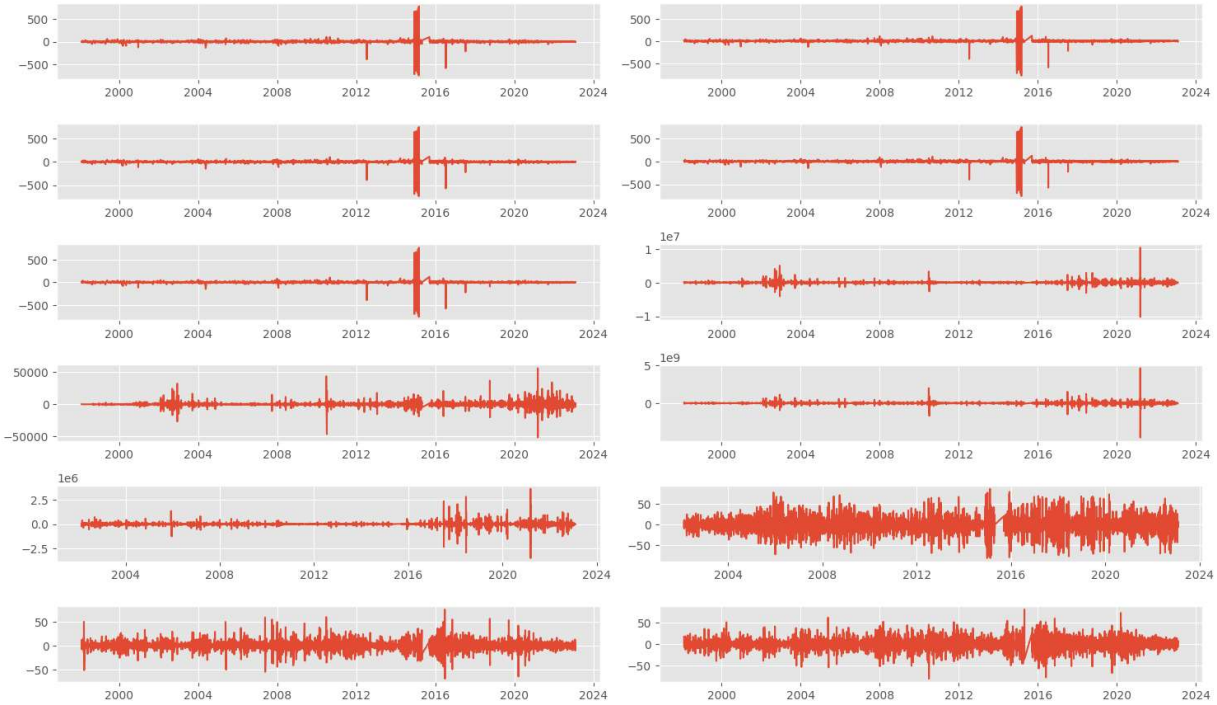
In [11]:

```
quandl.ApiConfig.api_key = API_KEY
df = quandl.get(BPCL_code, start_date= START_DATE, end_date=CURRENT_DATE, collapse =
```

In [12]:

```
plt.figure(figsize = [15,10])
position = 1
for column in df.columns:
    plt.subplot(7,2, position)
    plt.plot(df[column])
    position+=1
```

```
    plt.tight_layout()
plt.show()
```



Apart from the recurrent spikes from 2014 to 2015 if we observe data carefully then we get 5 different spikes

One in 2000, 2004 , 2012, 2016 , 2017

Task 2

2. Programmitically detect any splits and consolidations that have occurred.

In [13]:
```
newdf = bpcl_df.pct_change()
```

In [14]:
```
errors = newdf[abs(newdf['Close']) > 0.30]
clean_errors = errors
errors
```

Out[14]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	To Turno
2000-12-18	-0.514403	-0.502049	-0.498922	-0.492399	-0.499331	0.272748	0.047649	-3.62774
2004-05-18	-0.287257	-0.275424	-0.328462	-0.314064	-0.322514	0.441289	0.666586	-2.35470
2012-07-13	-0.498037	-0.501584	-0.504672	-0.507262	-0.503471	0.276081	-0.071390	-3.66388
2014-12-10	-0.976655	-0.976794	-0.976044	-0.976125	-0.976372	-0.999314	-0.999398	-9.99983
2014-12-11	38.616519	39.117994	37.849558	38.023599	38.293388	5249.820000	4278.000000	2.063210e·

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	To Turno
2014-12-17	-0.973050	-0.973958	-0.973001	-0.973747	-0.973690	-0.999510	-0.999816	-9.99987
2014-12-22	36.514451	36.997110	36.289017	36.884393	37.286471	80315.000000	1517.500000	3.075027e·
2014-12-23	-0.973498	-0.973834	-0.973337	-0.973756	-0.973712	-0.999888	-0.999341	-9.99997
2014-12-30	36.750000	37.229651	36.514535	36.779070	37.079486	7343.444444	1853.000000	2.796538e·
2015-01-05	-0.974436	-0.974529	-0.973988	-0.974090	-0.974221	-0.997195	-0.999392	-9.99927
2015-01-09	39.598802	40.215569	39.323353	39.559880	39.790419	652.984456	2550.000000	2.667628e·
2015-02-12	-0.973972	-0.974334	-0.973387	-0.973705	-0.974338	-0.999986	-0.999756	-9.99999
2015-02-16	38.678474	40.256131	38.302452	38.574932	40.153889	104634.000000	7207.000000	4.306157e·
2015-02-23	-0.975443	-0.975860	-0.975410	-0.975707	-0.975696	-0.996301	-0.999052	-9.99910
2015-02-26	45.218462	45.670769	44.661538	44.836923	44.982154	22588.000000	2722.500000	1.038685e·
2015-03-03	-0.978066	-0.978917	-0.978066	-0.978723	-0.978725	-0.999604	-0.999737	-9.99991
2015-03-04	46.341390	46.676737	45.163142	45.519637	46.109970	594.446154	1582.600000	2.805275e·
2016-07-13	-0.506630	-0.509827	-0.512780	-0.513347	-0.512129	0.613536	-0.336958	-2.12797
2017-07-13	-0.311730	-0.316739	-0.330412	-0.330191	-0.328410	1.023680	0.325251	3.590786e

In [15]:
```
# the most common stock split is 1/2 ,1/3 or 2/3 and it wouldn't be feasible if a co
# SO any thing beyond that should be excluded
errors[abs(errors['Close']) < 0.67]
```

Out[15]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliveral Quant
2000-12-18	-0.514403	-0.502049	-0.498922	-0.492399	-0.499331	0.272748	0.047649	-0.362774	N
2004-05-18	-0.287257	-0.275424	-0.328462	-0.314064	-0.322514	0.441289	0.666586	-0.023547	-0.203

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliveral Quant
2012-07-13	-0.498037	-0.501584	-0.504672	-0.507262	-0.503471	0.276081	-0.071390	-0.366388	0.273
2016-07-13	-0.506630	-0.509827	-0.512780	-0.513347	-0.512129	0.613536	-0.336958	-0.212797	1.074
2017-07-13	-0.311730	-0.316739	-0.330412	-0.330191	-0.328410	1.023680	0.325251	0.359079	0.527

Looking at the above data we can see that when the stock split occured the closing prices changed dramatically

But in the year 2004 there weren't any splits that occured so what could be the reason for this spike ??

here could be several reasons why the shift in the closing price of two different days may indicate a stock split, but no stock split was announced. Some possible reasons are:

Market volatility: Market conditions can cause fluctuations in the stock price, even if no major events such as a stock split have occurred.

Corporate news or announcements: Company-specific news, such as earnings reports or management changes, can also impact the stock price and cause large shifts.

Technical factors: Technical factors, such as trading volumes, can also impact the stock price and cause fluctuations.

Data error: There is always the possibility of data errors, such as typos or incorrect data inputs, that can result in incorrect stock price information.

It's important to consider multiple sources of information and not rely solely on a single dataset when analyzing stocks. It's also recommended to use multiple analytical methods to confirm the results and cross-reference with other sources to ensure accuracy.

Technical factors, such as trading volumes, can also impact the stock price and cause fluctuations.

Here we are having No of trades feature which shows that the shift in prices are definetely due to intra-day trading So we can re-filter the data that are no affected due to intraday trades

It can also occur due to incorrect entry but we can't verify it because of source limitations

In [16]:
```
errors  = errors[(abs(errors['Close']) < 0.67) & (abs(errors['No. of Trades'])<0.40)
errors
# ie less than 40% of trades have taken place assuming that the majority shares are
```

Out[16]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliveral Quant
2000-12-18	-0.514403	-0.502049	-0.498922	-0.492399	-0.499331	0.272748	0.047649	-0.362774	N
2012-07-13	-0.498037	-0.501584	-0.504672	-0.507262	-0.503471	0.276081	-0.071390	-0.366388	0.273
2016-07-13	-0.506630	-0.509827	-0.512780	-0.513347	-0.512129	0.613536	-0.336958	-0.212797	1.074
2017-07-13	-0.311730	-0.316739	-0.330412	-0.330191	-0.328410	1.023680	0.325251	0.359079	0.527

In [17]:
```
# hence in order to get all entries where the split occured we can index them and cr
```

Task 3

3. Correct the stock prices for any `splits` and `consolidations` that have occurred.

In [18]:
```
bpcl_df['Splits/Consolidation'] = np.nan
```

In [19]:
```
cumulative_multiple = 1
for date in errors.index[::-1] :
#     bpcl_df[date , 'Splits/Consolidation'] = errors.loc[date,'WAP']
    cumulative_multiple *= (1-abs(errors.loc[str(date) , 'WAP']))
    bpcl_df.loc[str(date), 'Splits/Consolidation'] = cumulative_multiple
    print(cumulative_multiple)
```

```
0.6715902775746586
0.3276492609502214
0.16268746771908787
0.08145262358583408
```

In [20]:
```
bpcl_df['Splits/Consolidation']= bpcl_df['Splits/Consolidation'].bfill()
```

In [21]:
```
bpcl_df['Splits/Consolidation'].replace(np.nan , 1, inplace = True)
```

In [22]:
```
bpcl_df['AdjClose'] = bpcl_df['Close']*bpcl_df['Splits/Consolidation']
```

In [23]:
```
cumulative_multiple = 1
for date in errors.index[::-1] :

    bpcl_df.loc[str(date) , 'AdjClose']  = cumulative_multiple*bpcl_df.loc[str(date)
    cumulative_multiple *= (1-abs(errors.loc[str(date) , 'WAP']))
```

In [24]:
```
bpcl_df['AdjClose'].plot(figsize = [15,5] , color = 'Green')
# plt.axvline('2000-12-18')
plt.show()
```

In [ ]:
```
```

In [25]:
```
incorrect_inputs = bpcl_df.loc[clean_errors[clean_errors.index.isin(errors.index) ==
```

In [26]:
```
incorrect_inputs = bpcl_df.loc[incorrect_inputs[incorrect_inputs['Close']== True].in
incorrect_inputs
```

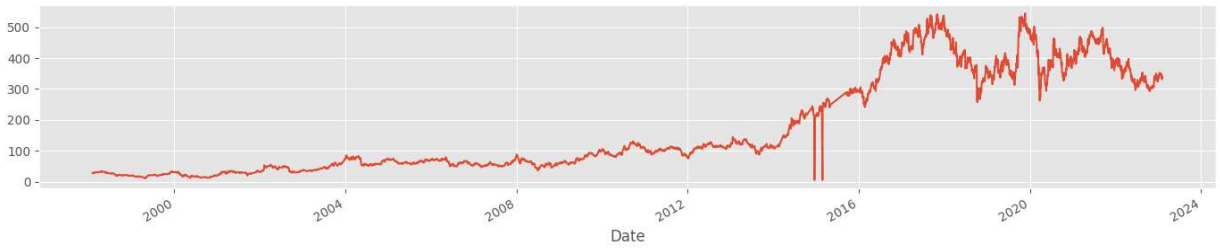
Out[26]:

Date	Open	High	Low	Close	WAP	No. of Shares	No. of Trades	Total Turnover	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread H-L	Spr
2014-12-10	16.95	16.95	16.95	16.95	16.94	50.0	2.0	847.0	50.0	100.0	0.0	
2014-12-17	17.00	17.00	17.00	17.00	17.00	100.0	2.0	1700.0	100.0	100.0	0.0	
2014-12-23	17.20	17.20	17.20	17.20	17.11	9.0	2.0	154.0	9.0	100.0	0.0	
2015-01-05	16.70	16.70	16.70	16.70	16.70	193.0	3.0	3223.0	193.0	100.0	0.0	
2015-02-12	18.35	18.35	18.35	18.35	18.00	2.0	2.0	36.0	2.0	100.0	0.0	
2015-02-23	18.00	18.00	18.00	18.00	18.00	1105.0	6.0	19890.0	1105.0	100.0	0.0	
2015-03-03	16.55	16.55	16.55	16.55	16.55	195.0	5.0	3227.0	195.0	100.0	0.0	

In [27]:
```
bpcl_df.drop(index = incorrect_inputs.index , inplace = True)
```

In [28]:
```
bpcl_df['AdjClose'].plot(kind='line' , figsize = [17,3])
```

Out[28]:
```
<AxesSubplot: xlabel='Date'>
```



In [29]:
```
while len(incorrect_inputs)>0:
    newdf = bpcl_df.pct_change()
    errors = newdf[abs(newdf['Close']) > 0.30]
    clean_errors = errors
    errors  = errors[(abs(errors['Close']) < 0.67) & (abs(errors['No. of Trades'])<0
    incorrect_inputs = bpcl_df.loc[clean_errors[clean_errors.index.isin(errors.index
    incorrect_inputs = bpcl_df.loc[incorrect_inputs[incorrect_inputs['Close']== True
    bpcl_df.drop(index = incorrect_inputs.index , inplace = True)
```

In [30]:
```
def annotation(index):
    plt.annotate('Split', (mdates.date2num(errors.index[index]), bpcl_df.loc[errors.
            textcoords='offset points', arrowprops=dict(arrowstyle='-|>'))
```

Task 4

4. Display the final adjusted stock prices.

In [31]:
```
bpcl_df['AdjClose'].plot(kind='line' , figsize = [15,5] , color = 'Green' , label= '
# for date in errors.index:
#     plt.axvline(date , color = 'Green')
plt.scatter(x=errors.index , y= bpcl_df.loc[errors.index,'AdjClose'] , s = 150 , c =


plt.title('BPCL Stock Price 1998-2023')
plt.xlabel('Date (1998 - 2023)')
plt.ylabel('Price (INR)')
plt.axvspan(errors.index[0], errors.index[1], alpha=0.2, color="green")
plt.axvspan(errors.index[1], errors.index[2], alpha=0.2, color="red")
plt.axvspan(errors.index[2], errors.index[3], alpha=0.2, color="indigo")
plt.axvspan(errors.index[3], datetime.now(), alpha=0.2, color="yellow")
for index in range(len(errors)):
    annotation(index)
plt.legend()
plt.tight_layout()
plt.show()
```

Task 5

1. Show the dates for any splits and consolidations that have occurred.

Dates for which splits / consolidation occured

['2000-12-18', '2012-07-13', '2016-07-13', '2017-07-13']

In [32]:
```
errors.index
```

Out[32]:
```
DatetimeIndex(['2000-12-18', '2012-07-13', '2016-07-13', '2017-07-13'], dtype='datet
ime64[ns]', name='Date', freq=None)
```

Project Summary

The code imports the necessary libraries such as Quandl, Pandas, Matplotlib, and Seaborn. It also sets the maximum number of rows to display in the output to None. The API key for Quandl is stored in the variable "API_KEY".

The stock code for Bharat Petroleum Corporation Limited (BPCL) is stored in the variable "BPCL_code". The current date and the start date (25 years prior to the current date) are calculated using the datetime library and stored as "CURRENT_DATE" and "START_DATE", respectively. The stock data for BPCL is then fetched from Quandl using the start and end dates and stored in the data frame "bpcl_df".

The code then plots a line graph to show the stock price of BPCL over time and also plots subplots for each column in the data frame to show the trends in each feature.

In the next section, the code fetches the difference in the stock price data and plots subplots to show the trends in each feature. The code also calculates the percentage change in the "Close" feature and identifies errors where the absolute value of the change exceeds `0.30`. The code then removes any errors where the absolute change exceeds `0.67`, as it could'nt indicate a stock split, which is not feasible.

The code finally concludes that the spike in the stock price in 2004 could be due to several factors such as market volatility, corporate news or announcements, technical factors, or data errors. It is important to consider multiple sources of information and not rely solely on a single dataset when analyzing stocks.

Note :

In the above notebook there were multiple noisy features from 2014 to 2015 and given the quick succession of changes that occured in the short of time it is assumed that these were errors and not real market changes If these fluctuations are meant to a part of consolidations and splits then `revert back` *so that the changes could be made*