# DATA ANALYTICS
# Assignment 3

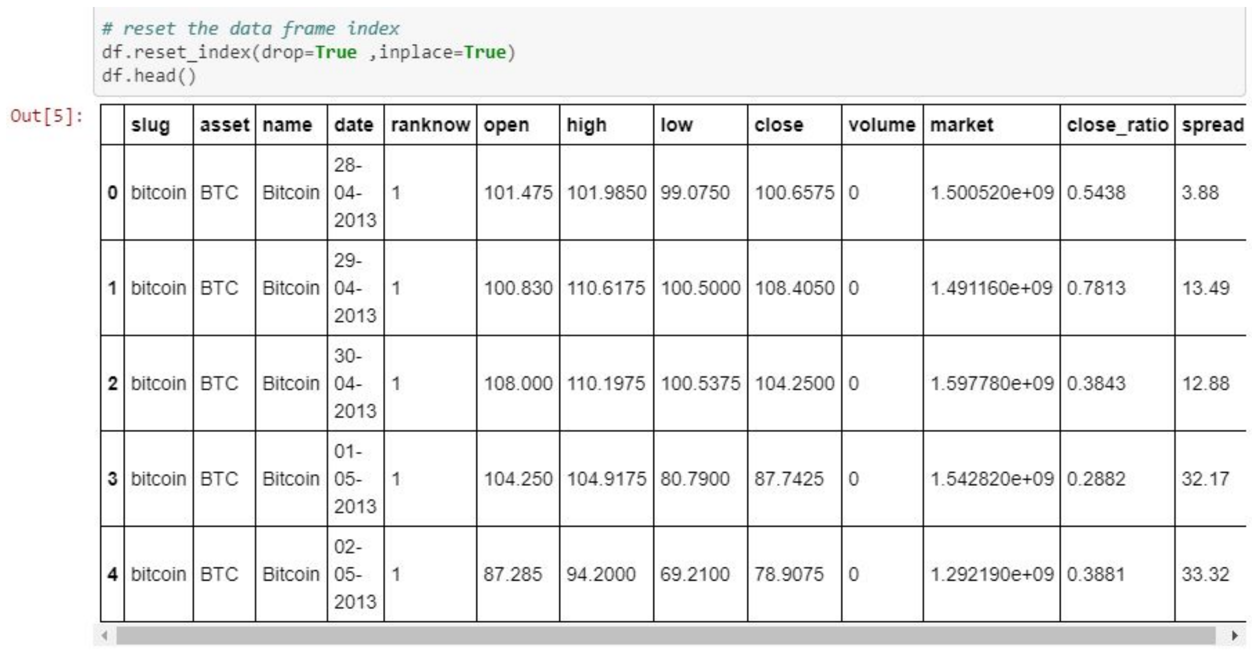
**Amrutha S - USN : PES120700829**

**Yoshitha - USN : PES1201701744**

**Dhruv - USN : PES1201700122**

**Swathi - USN : PES1201701826**

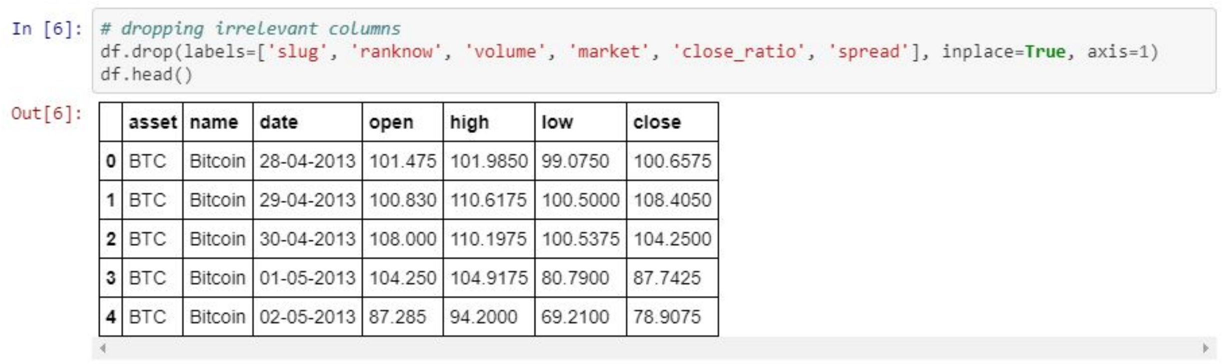
PROBLEM STATEMENT :
**Perform Regression on a Crypt currency dataset.**



```
# reset the data frame index
df.reset_index(drop=True ,inplace=True)
df.head()
```

Out[5]:

| | slug | asset | name | date | ranknow | open | high | low | close | volume | market | close_ratio | spread |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | bitcoin | BTC | Bitcoin | 28-04-2013 | 1 | 101.475 | 101.9850 | 99.0750 | 100.6575 | 0 | 1.500520e+09 | 0.5438 | 3.88 |
| 1 | bitcoin | BTC | Bitcoin | 29-04-2013 | 1 | 100.830 | 110.6175 | 100.5000 | 108.4050 | 0 | 1.491160e+09 | 0.7813 | 13.49 |
| 2 | bitcoin | BTC | Bitcoin | 30-04-2013 | 1 | 108.000 | 110.1975 | 100.5375 | 104.2500 | 0 | 1.597780e+09 | 0.3843 | 12.88 |
| 3 | bitcoin | BTC | Bitcoin | 01-05-2013 | 1 | 104.250 | 104.9175 | 80.7900 | 87.7425 | 0 | 1.542820e+09 | 0.2882 | 32.17 |
| 4 | bitcoin | BTC | Bitcoin | 02-05-2013 | 1 | 87.285 | 94.2000 | 69.2100 | 78.9075 | 0 | 1.292190e+09 | 0.3881 | 33.32 |



In [6]:
```
# dropping irrelevant columns
df.drop(labels=['slug', 'ranknow', 'volume', 'market', 'close_ratio', 'spread'], inplace=True, axis=1)
df.head()
```

Out[6]:

| | asset | name | date | open | high | low | close |
|---|---|---|---|---|---|---|---|
| 0 | BTC | Bitcoin | 28-04-2013 | 101.475 | 101.9850 | 99.0750 | 100.6575 |
| 1 | BTC | Bitcoin | 29-04-2013 | 100.830 | 110.6175 | 100.5000 | 108.4050 |
| 2 | BTC | Bitcoin | 30-04-2013 | 108.000 | 110.1975 | 100.5375 | 104.2500 |
| 3 | BTC | Bitcoin | 01-05-2013 | 104.250 | 104.9175 | 80.7900 | 87.7425 |
| 4 | BTC | Bitcoin | 02-05-2013 | 87.285 | 94.2000 | 69.2100 | 78.9075 |

In [7]:
```
import sqlite3
# import cx_Oracle 'username/password@hostname:port/service_name'
# connect function opens a connection to the SQLite database file,
conn = sqlite3.connect('session.db')
#Similarly we will make connection with other databases like Oracle, DB2 etc.
print(conn)
```

```
<sqlite3.Connection object at 0x000002CA23CB2AB0>
```

In [8]:
```
# Drop a table name Crypto if it exists already
try:
    conn.execute('DROP TABLE IF EXISTS `Crypto` ')
except Exception as e:
    raise(e)
finally:
    print('Table dropped')
```

```
Table dropped
```

In [9]:
```
# Create a new Table named as Crypto
try:
    conn.execute('''
         CREATE TABLE Crypto
         (ID         INTEGER PRIMARY KEY,
         ASSET       TEXT    NOT NULL,
         NAME        TEXT    NOT NULL,
         Date        datetime,
         Open        Float DEFAULT 0,
         High        Float DEFAULT 0,
         Low         Float DEFAULT 0,
         Close       Float DEFAULT 0);''')
    print ("Table created successfully");
except Exception as e:
    print(str(e))
    print('Table Creation Failed!!!!!')
finally:
    conn.close() # this closes the database connection
```

```
Table created successfully
```

In [10]:
```
crypto_list = df.values.tolist()

# lets make new connection to Insert crypto data in SQL DB
conn = sqlite3.connect('session.db')

# make a cursor - it will help with querying SQL DB
cur = conn.cursor()

try:
    cur.executemany("INSERT INTO Crypto(ASSET, NAME, Date, Open, High, Low, Close) VALUES
(?,?,?,?,?,?,?)", crypto_list)
    conn.commit()
    print('Data Inserted Successfully')
except Exception as e:
    print(str(e))
    print('Data Insertion Failed')
finally:
    # finally block will help with always closing the connection to DB even in case of error.
    conn.close()
```

```
Data Inserted Successfully
```

In [11]:
```python
# Let's Read data from DB to verify it

conn = sqlite3.connect('session.db')
rows = conn.cursor().execute('Select * from Crypto')
# print(rows[:2])
for row in rows:
    print(row)
conn.close()
```

```
(1, 'BTC', 'Bitcoin', '28-04-2013', 101.47500000000001, 101.98499999999999, 99.07499999999999, 100.6575)
(2, 'BTC', 'Bitcoin', '29-04-2013', 100.83, 110.6175, 100.5, 108.405)
(3, 'BTC', 'Bitcoin', '30-04-2013', 108.0, 110.1975, 100.53750000000001, 104.25)
(4, 'BTC', 'Bitcoin', '01-05-2013', 104.25, 104.91749999999999, 80.78999999999999, 87.74249999999999)
(5, 'BTC', 'Bitcoin', '02-05-2013', 87.285, 94.19999999999999, 69.21000000000001, 78.9075)
(6, 'BTC', 'Bitcoin', '03-05-2013', 79.6875, 81.0975, 59.324999999999996, 73.3125)
(7, 'BTC', 'Bitcoin', '04-05-2013', 73.57499999999999, 86.25, 69.375, 84.375)
(8, 'BTC', 'Bitcoin', '05-05-2013', 84.67500000000001, 89.1, 80.355, 86.9325)
(9, 'BTC', 'Bitcoin', '06-05-2013', 86.985, 93.495, 79.98, 84.225)
(10, 'BTC', 'Bitcoin', '07-05-2013', 84.1875, 85.08, 73.275, 83.625)
(11, 'BTC', 'Bitcoin', '08-05-2013', 82.19999999999999, 86.83500000000001, 82.19999999999999, 85.1775)
(12, 'BTC', 'Bitcoin', '09-05-2013', 84.9, 85.095, 81.94500000000001, 84.5025)
(13, 'BTC', 'Bitcoin', '10-05-2013', 84.6, 91.5, 83.6625, 87.9)
(14, 'BTC', 'Bitcoin', '11-05-2013', 88.275, 89.01, 84.75750000000001, 86.42999999999999)
(15, 'BTC', 'Bitcoin', '12-05-2013', 86.73, 88.0875, 85.08, 86.25)
(16, 'BTC', 'Bitcoin', '13-05-2013', 86.115, 89.025, 85.875, 88.485)
(17, 'BTC', 'Bitcoin', '14-05-2013', 88.485, 89.85, 82.6875, 83.625)
(18, 'BTC', 'Bitcoin', '15-05-2013', 83.55000000000001, 86.8575, 77.625, 85.66499999999999)
(19, 'BTC', 'Bitcoin', '16-05-2013', 85.66499999999999, 89.07000000000001, 84.15, 89.07000000000001)
(20, 'BTC', 'Bitcoin', '17-05-2013', 88.6575, 93.975, 87.4275, 92.265)
(21, 'BTC', 'Bitcoin', '18-05-2013', 92.625, 93.9375, 91.725, 92.625)
(22, 'BTC', 'Bitcoin', '19-05-2013', 92.4075, 93.375, 89.6775, 91.49249999999999)
(23, 'BTC', 'Bitcoin', '20-05-2013', 91.875, 92.715, 90.09, 91.5)
(24, 'BTC', 'Bitcoin', '21-05-2013', 91.515, 92.25, 90.9075, 92.16)
(25, 'BTC', 'Bitcoin', '22-05-2013', 92.1675, 93.0, 91.5, 92.9175)
(26, 'BTC', 'Bitcoin', '23-05-2013', 92.85, 95.1975, 92.32499999999999, 95.025)
```