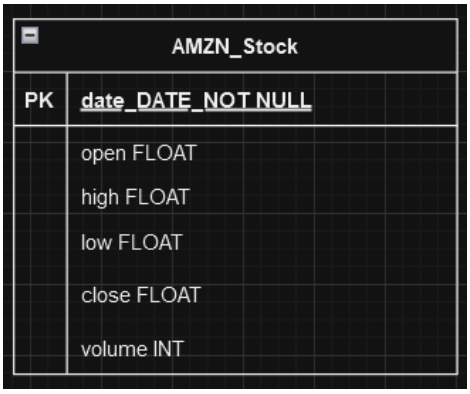
PEP Capstone Checkpoint 3

Data Loading



Given the nature of the retrieved data, a simple schema was judged to be sufficient.

Most of this checkpoint's complexity lay in the mechanics of inserting non-duplicate data into the table every time the code was run, and only inserting the most recent data.

Depending on the state of the table ie. empty/not empty, the parameters of the API call would change.

Function for retrieving data from API, output size either 'full' (entire dataset) or 'compact' (last 100 days' worth):

```
def api_call(output_size, api_key):
    return rq.get(f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=AMZN&outputsize={output_size}&apikey={api_key}').json()
```

SQLite statement executed to count existing rows in table. If table is empty, the full dataset is retrieved. If table already has data, only the last 100 days is retrieved. Dataframe is created in both cases.

```
is_empty = True
if conn.execute(f"SELECT COUNT(*) FROM amzn;").fetchone()[0] == 0:
    data = api_call('full', 'KVUJT3KI90X21VGB')
else:
    data = api_call('compact', 'KVUJT3KI90X21VGB')
    is_empty = False

time_series_data = data["Time Series (Daily)"]
```

If table is empty, every row in the dataframe is inserted into the table. If table is not empty, only the entries with a more recent date than the highest date in the table will be inserted.

```
if is_empty:
    df.to_sql('amzn', conn, index=False, if_exists='append')
    print("All data inserted.")
else:
    max_date = conn.execute("SELECT MAX(date) FROM amzn;").fetchone()[0]
    df_filtered = df[pd.to_datetime(df['date']) > pd.to_datetime(max_date)]
    if df_filtered.empty:
        print("No new data to insert.")
    else:
        df_filtered.to_sql('amzn', conn, index=False, if_exists='append')
        conn.commit()
        print("New data inserted.")
```

SQLite table filled with data (6113 entries):

SQL ▾ < 1 / 123 > 1 - 50 of 6113

date	open	high	low	close	volume
2024-02-16	168.74	170.42	167.17	169.51	48107744
2024-02-15	170.58	171.17	167.59	169.8	49855196
2024-02-14	169.21	171.21	168.28	170.98	42815544
2024-02-13	167.73	170.95	165.75	168.64	56345122
2024-02-12	174.8	175.39	171.54	172.34	51050440
2024-02-09	170.9	175.0	170.5803	174.45	56985986

Successful data insert message:

```
if is_empty:
    df.to_sql('amzn', conn, index=False, if_exists='append')
    print("All data inserted.")
else:
    max_date = conn.execute("SELECT MAX(date) FROM amzn;").fetchone()[0]
    df_filtered = df[pd.to_datetime(df['date']) > pd.to_datetime(max_date)]
    if df_filtered.empty:
        print("No new data to insert.")
    else:
        df_filtered.to_sql('amzn', conn, index=False, if_exists='append')
        conn.commit()
        print("New data inserted.")
```

✓ 0.0s

```
All data inserted.
```

Successful data update message:

```
if is_empty:
    df.to_sql('amzn', conn, index=False, if_exists='append')
    print("All data inserted.")
else:
    max_date = conn.execute("SELECT MAX(date) FROM amzn;").fetchone()[0]
    df_filtered = df[pd.to_datetime(df['date']) > pd.to_datetime(max_date)]
    if df_filtered.empty:
        print("No new data to insert.")
    else:
        df_filtered.to_sql('amzn', conn, index=False, if_exists='append')
        conn.commit()
        print("New data inserted.")
```

✓ 0.0s

```
New data inserted.
```