Portfolio Application & Study

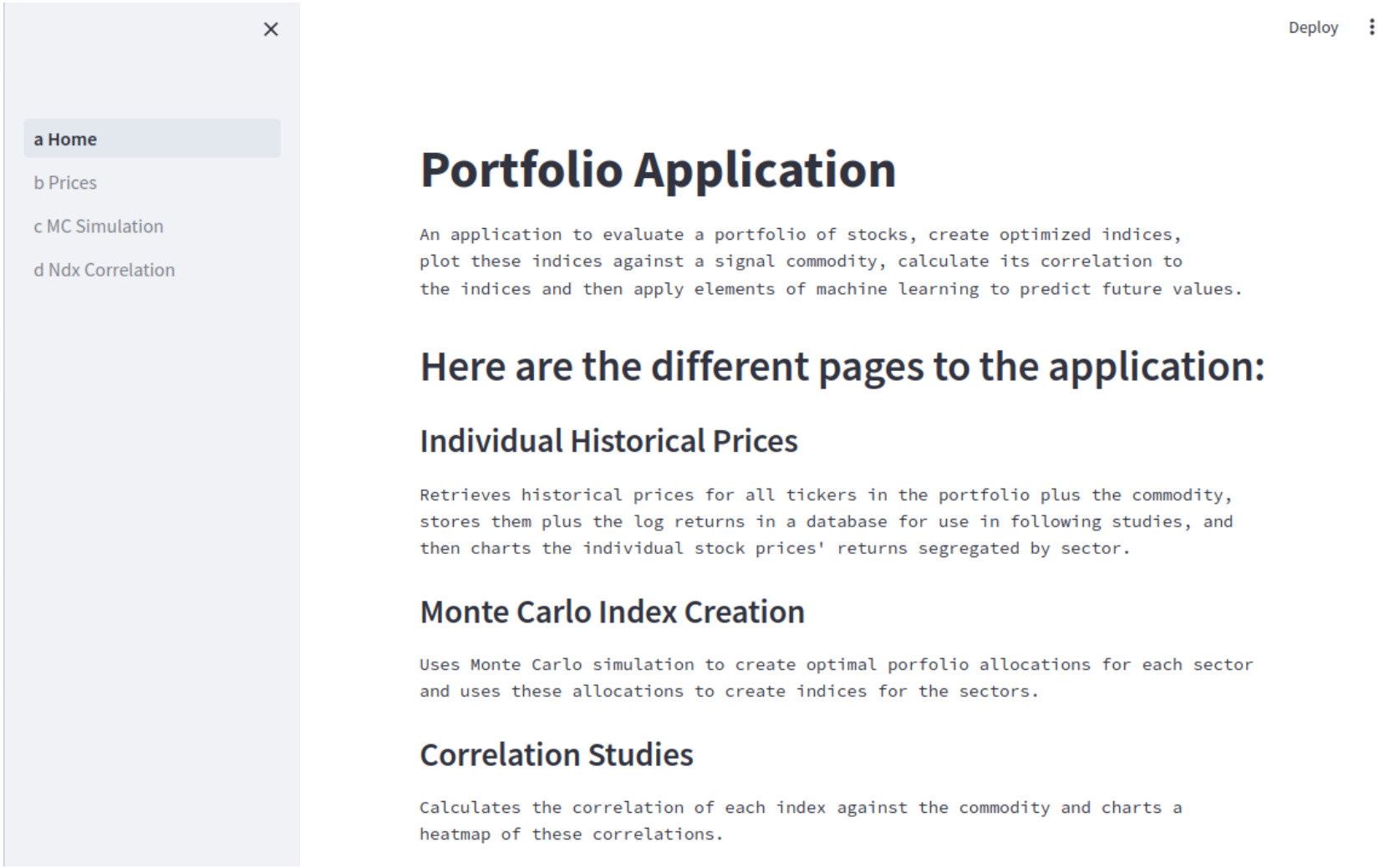
A real-world application in conclusion to an 8-week Python & data analytics bootcamp.

Beginning Business & Tech Specs

- Given an equity portfolio across industry sectors:
  - Bring in 20 years of price data
  - Find optimal weighting for combined & individual sectors
  - Compute indices for each sector and combined and chart against underlying commodity
  - Find correlation of each sector to the underlying commodity
- Technology Currently Used:
  - Python: numpy, pandas, streamlit, yfinance, matplotlib, VS Code
  - PostgreSQL Database server, RazorSQL
- Next Version Technology
  - All currently used tech +:
  - Amazon RDS for Postgres – AWS
  - Financial Modeling Prep & OpenBB for financial fundamental data

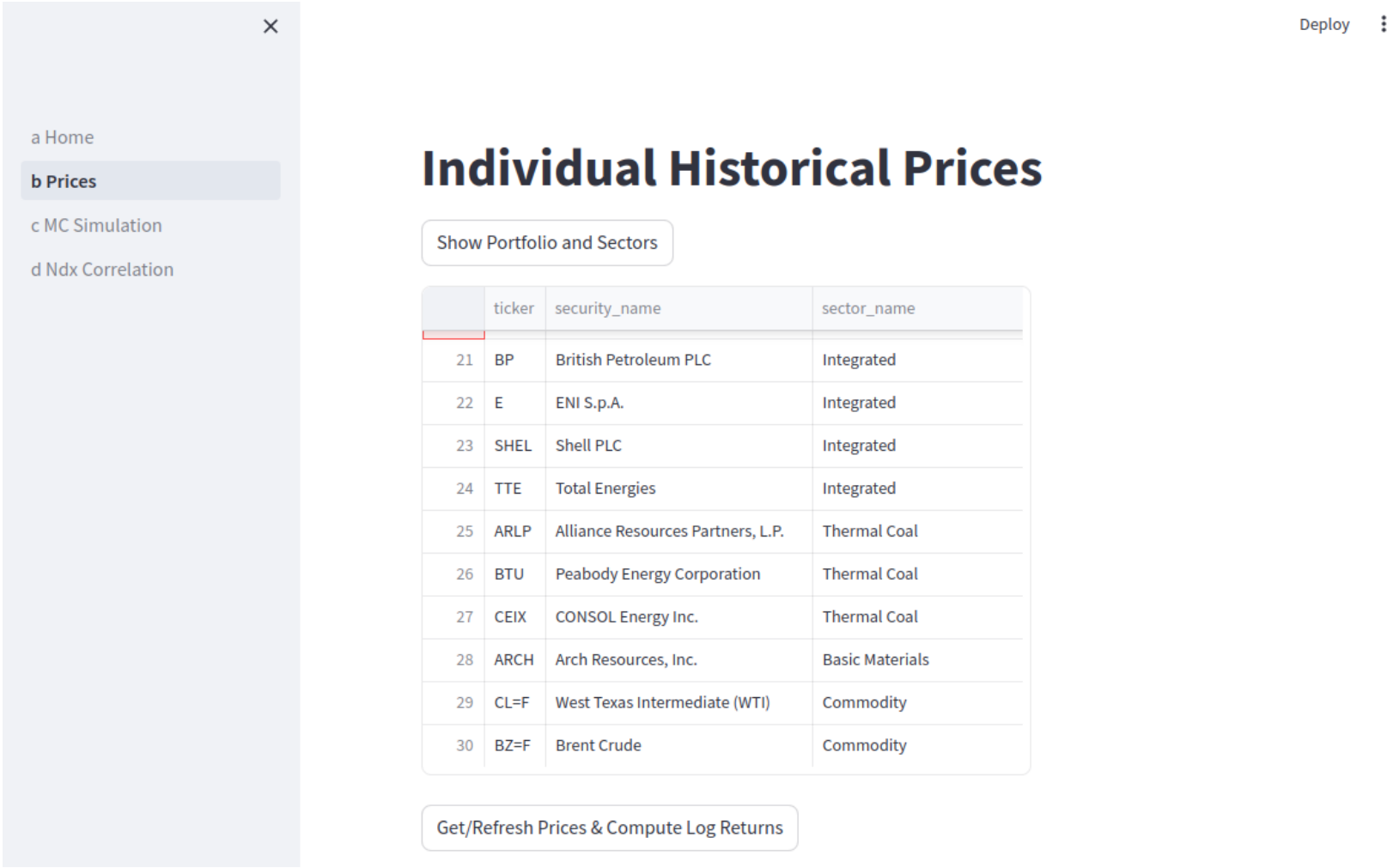
Application Screen: Home

The first, "Home", screen:



Application Screen: Prices 1

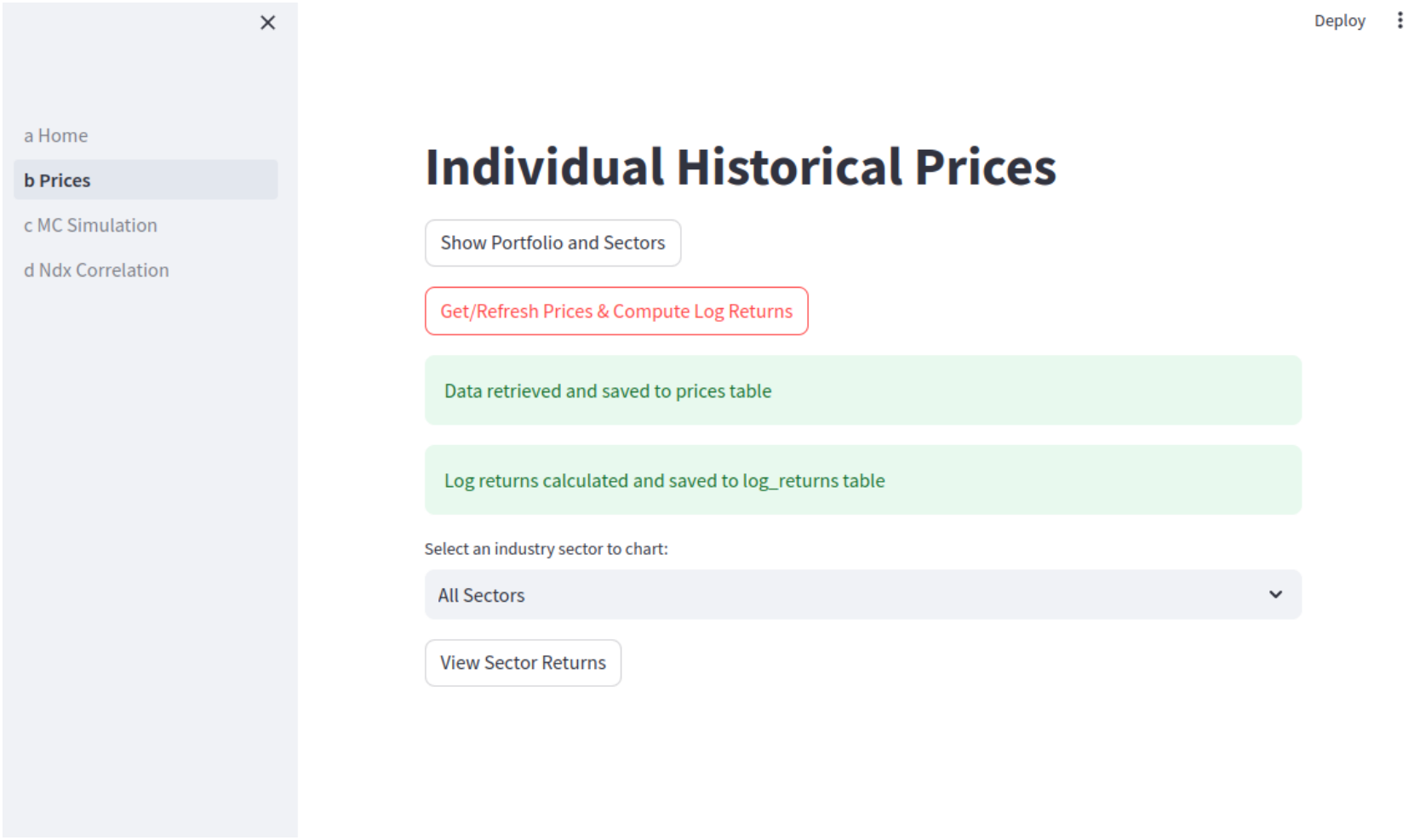
Prices Screen Showing the portfolio across its various industry sectors



	ticker	security_name	sector_name
21	BP	British Petroleum PLC	Integrated
22	E	ENI S.p.A.	Integrated
23	SHEL	Shell PLC	Integrated
24	TTE	Total Energies	Integrated
25	ARLP	Alliance Resources Partners, L.P.	Thermal Coal
26	BTU	Peabody Energy Corporation	Thermal Coal
27	CEIX	CONSOL Energy Inc.	Thermal Coal
28	ARCH	Arch Resources, Inc.	Basic Materials
29	CL=F	West Texas Intermediate (WTI)	Commodity
30	BZ=F	Brent Crude	Commodity

Application Screen: Prices 2

Pull in 20 years of pricing data, compute the log returns for each security or commodity, and save prices and returns to database tables.



Application Screen: Prices 3

Chart prices in each industry sector

Deploy

a Home

b Prices

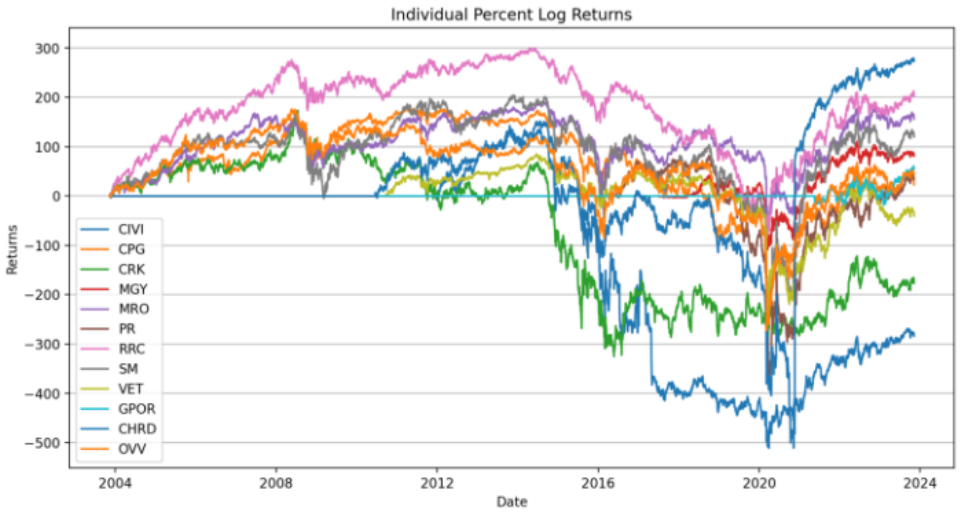
c MC Simulation

d Ndx Correlation

Select an industry sector to chart:

Exploration and Production

View Sector Returns



Security Descriptions

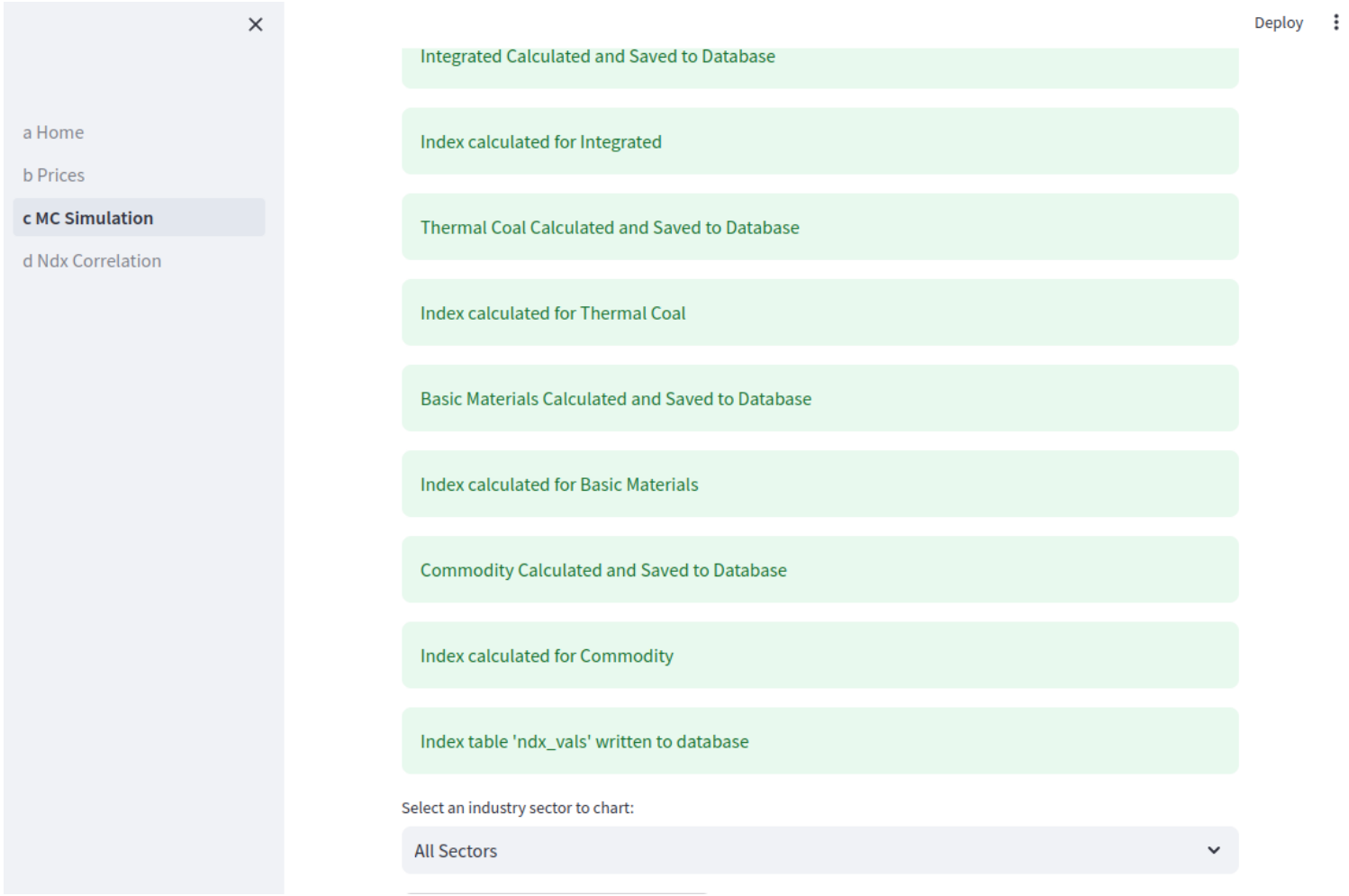
	ticker	security_name
0	CIVI	Civitas Resources, Inc.
1	CPG	Crescent Point Energy Corporation

Application Screen: Monte Carlo Simulation 1

Screen will take log returns of securities and compute optimal weighting of securities within each sector and combined portfolio by Sharpe Ratio, compute indices starting at a value of 100 at start date for all sectors and commodity.

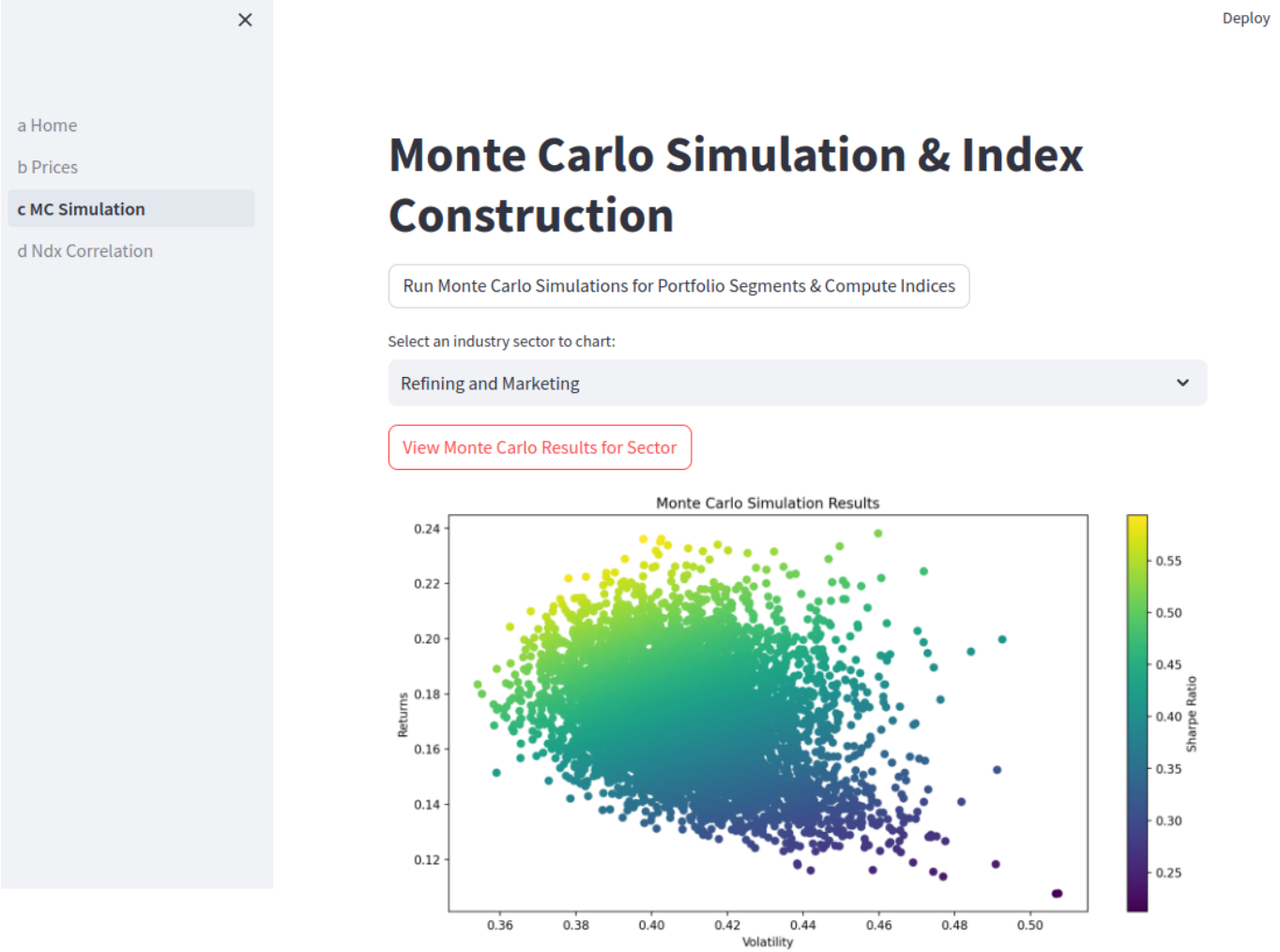
Application Screen: Monte Carlo Simulation 2

Screen after simulations, indices calculations then storing all simulations and indices to their respective database tables



Application Screen: Monte Carlo Simulation 3

Charting of the simulation for an industry across 6000 random simulations of weights.



Example Index Derived From Simulations

price_date	all_sect	ep	ref_mkt	equip	int
2003-11-11 00:00:00	100	100	100	100	100
2003-11-12 00:00:00	100.442057674713354	100.502677079450123	100.114884457620917	100.326610214399409	101.055044996069341
2003-11-13 00:00:00	100.739725184761895	101.107665041182017	100.026641173601504	100.544449826409931	101.808593856599131
2003-11-14 00:00:00	100.953690739740324	101.294096428059504	100.012721282291722	101.772783605821544	102.547288964646981
2003-11-17 00:00:00	100.598785211230336	101.012433800355751	99.5210153730186278	99.9760634882280641	101.164257244991774
2003-11-18 00:00:00	100.497562498755002	100.938285343144472	99.4703845139057279	100.628036726280584	100.952564354104481
2003-11-19 00:00:00	100.560444984645699	100.885593838974629	99.83517617289165	100.300473225983524	101.535953659317926
2003-11-20 00:00:00	100.493157927429635	100.943019882069279	100.192316751997822	98.2815324434258031	100.733763280363448
2003-11-21 00:00:00	100.571865123710495	101.190466839351373	100.423829540675584	98.6803843534722915	101.40712260642961
2003-11-24 00:00:00	100.773977192498066	101.106191034694987	100.85633479003522	97.6577850931120821	102.183712357480644
2003-11-25 00:00:00	100.984049749697306	101.146846290394066	101.100777148105067	97.7304397672216538	102.03903930164428
2003-11-26 00:00:00	101.402850915202819	101.406650446606861	101.19455010648754	99.1371678929434381	103.375764898530321
2003-11-28 00:00:00	101.606555270249473	101.555265813830502	101.103631835682876	99.2460774113616679	103.794265920496485
2003-12-01 00:00:00	102.552209547326996	102.286921913761688	101.582321145478176	101.2597983683211	106.113955672752454

Exploration & Production	
Return	12.2%
Volatility	54.7%
Sharpe Ratio	0.222
CIVI	0.3%
CPG	9.1%
CRK	0.9%
MGY	15.0%
MRO	10.0%
PR	5.3%
RRC	15.9%
SM	1.2%
VET	0.8%
GPOR	18.0%
CHRD	17.0%
OVV	6.4%
Weights Sum	**100.0%**

- Using the simulation producing the best Sharpe Ratio over 20 years, a weighting is derived for each industry sector
- These weightings for each sector are combined with the daily log returns for each security to construct indices – all beginning at the price of 100 at the beginning date back 20 years and work forward to the present.

Application Screen: Index Results I

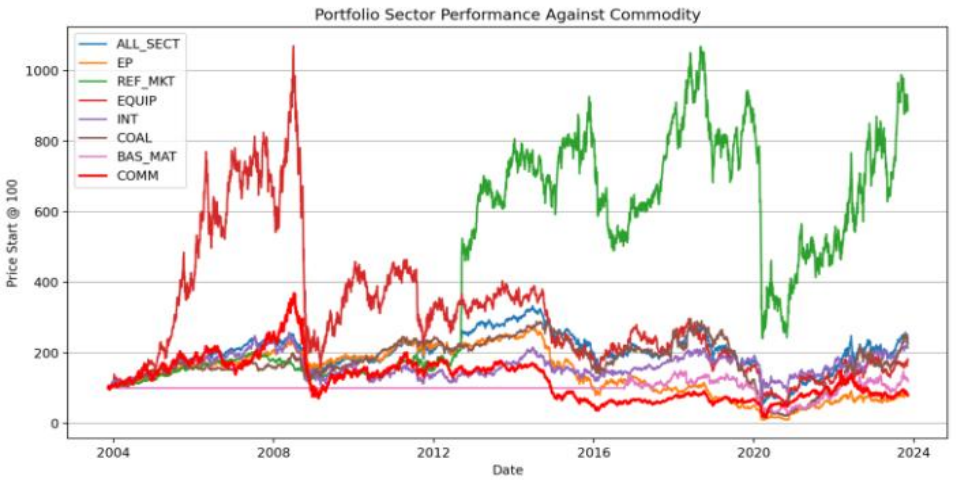
All indices can now be charted against the underlying commodity (COMM, bright red line)

a Home

b Prices

c MC Simulation

d Ndx Correlation

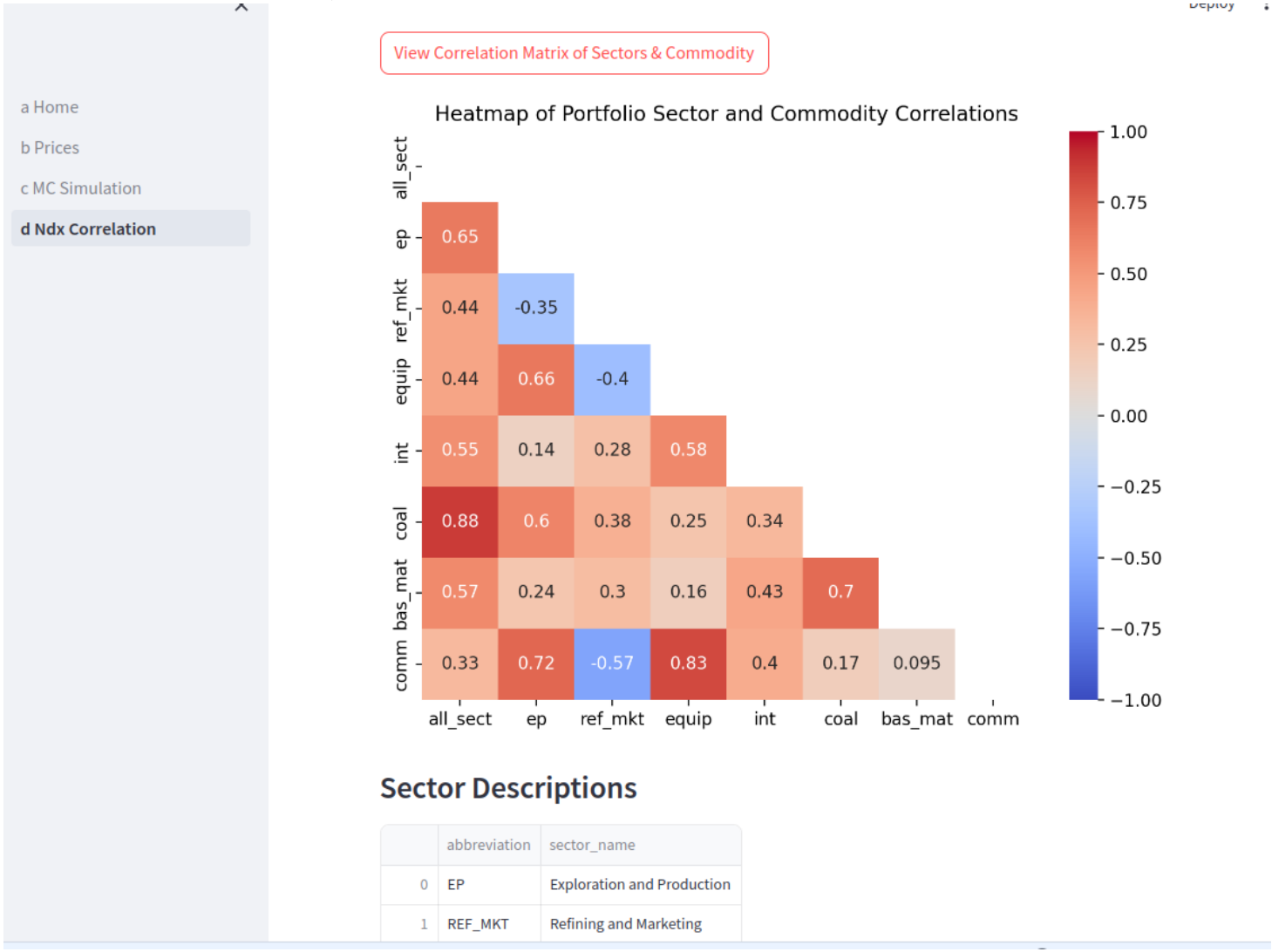


Sector Descriptions

	abbreviation	sector_name
0	EP	Exploration and Production
1	REF_MKT	Refining and Marketing
2	EQUIP	Equipment and Services
3	INT	Integrated
4	COAL	Thermal Coal
5	BAS_MAT	Basic Materials
6	COMM	Commodity
7	ALL_SECT	All Sectors

Application Screen: Index Results 2

With the indices and underlying commodity normalized, their correlations can be revealed

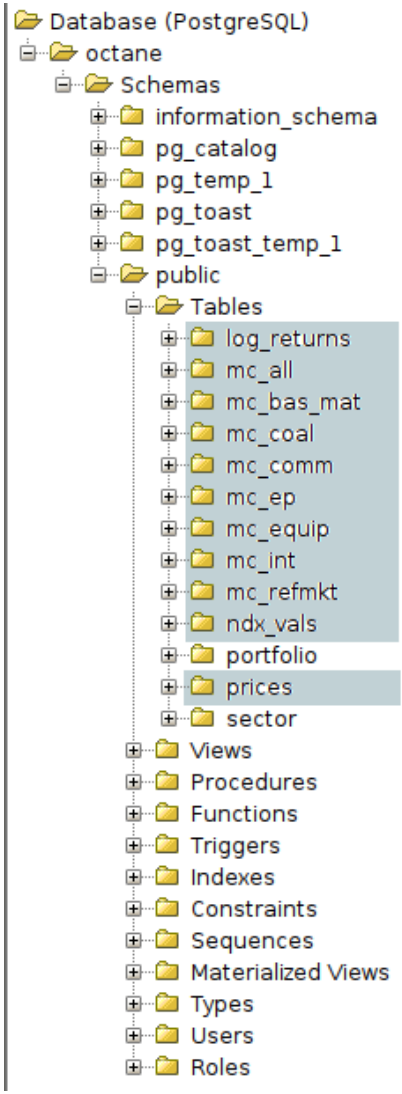


Future Fundamental Metrics

- With indices and correlations determined, next phase will be to retrieve:
  - Free Cash Flow industry rankings current and historical
  - Balance Sheet strength via debt/equity
  - Return on Equity
- Construct a screening tool for top tercile companies.
- Construct a backtest for the various screening methodologies considered.

Database Population via Application

Starting with two tables for the individual securities and the industry sectors, all other tables are calculated and stored during the process.



Python to SQL Interaction

An example of using field names in database tables to write SQL and Python to generate SQL statements:

```
# get sector table and fields that drive indices name and monte carlo table name
sql = "select sector_id, sector_name, mc_table, ndx_name from sector order by sector_id"
sector_df = db.alc_query(sql)

first = True
for _, row in sector_df.iterrows():
    sector_id = row["sector_id"]
    sector_name = row["sector_name"]
    mc_table = row["mc_table"]
    ndx_name = row["ndx_name"]

    if sector_id == 0:
      sql = "select db_name from portfolio where sector_id <> 7"
    else:
      sql = f"select db_name from portfolio where sector_id = {sector_id}"
    # get tickers for portfolio
    df = db.alc_query(sql)
    # turn into a list
    port_list = df["db_name"].tolist()
    #### use this list to run through the external Monte Carlo function
    mc_df = mc_core.mc_hammer(port_list)
    # save results to a table
    tablename = mc_table
    db.alc_df_2_db(mc_df, tablename)
    st.success(f"{sector_name} Calculated and Saved to Database")
    #### calculate the index value on optimal weighting found
    if first:
      # put first index into ndx_vals dataframe
      ndx_vals = mc_core.ndx_calc(mc_df)
      ndx_vals.rename(columns={"portfolio_value": ndx_name}, inplace=True)
      first = False
    else:
      # calc successive indices in their own dataframes
      new_ndx = mc_core.ndx_calc(mc_df)
      new_ndx.rename(columns={"portfolio_value": ndx_name}, inplace=True)
      # ... and then join them into the ndx_vals dataframe
      ndx_vals = pd.merge(
        ndx_vals,
        new_ndx[["price_date", ndx_name]],
        on="price_date",
        how="left",
        )
    st.success(f"Index calculated for {sector_name}")
```