

Claude Skills, Fundamental Indicators

BUSI 722: Data-Driven Finance II

Kerry Back

Database Access

The Rice Data Portal

- We will use the **Rice Data Portal** for stock market data.
- The portal provides daily prices, valuation metrics, and fundamental data from SEC filings.
- Four main tables:
 - **SEP** – daily stock prices (open, high, low, close, volume)
 - **DAILY** – daily valuation metrics (marketcap, pb, pe, ps)
 - **SF1** – fundamentals from 10-K/10-Q filings
 - **TICKERS** – company metadata (sector, industry)

Storing Your Access Token

Visit data-portal.rice-business.org to get an access token.

Prompt: “Create a `.env` file with my Rice access token: `abc123...`”

Claude creates a `.env` file:

```
RICE_ACCESS_TOKEN=abc123...
```

- The `.env` file keeps your token out of your code.
- Python’s `dotenv` library loads it automatically.
- Add `.env` to `.gitignore` to avoid sharing your token.

Step 1: Fetch Monthly Returns

Fetching Returns

Prompt: “Get monthly returns for all stocks from 2010 onward and save as `monthly.parquet`.”

Claude uses the `fetch-returns` skill, which:

1. Queries **SEP** for end-of-month prices (year by year to avoid timeouts)
2. Queries **DAILY** for end-of-month market cap
3. Queries **TICKERS** for sector and industry
4. Calculates return, momentum, and lagged return
5. Assigns size categories (Nano- through Mega-Cap)

Includes **delisted stocks** by default to avoid survivorship bias.

Returns Data: What You Get

```
>>> df = pd.read_parquet("monthly.parquet")
>>> df.shape
(544262, 8)
>>> df.columns
['ticker', 'month', 'return', 'momentum', 'lagged_return',
 'close', 'marketcap', 'pb']
```

```
>>> df[df.ticker=="AAPL"].head(5)
```

ticker	month	return	momentum	lagged_return	close	marketcap	pb
AAPL	2010-02	0.0655	NaN	NaN	6.859	174151.7	4.9
AAPL	2010-03	0.1484	NaN	0.0655	7.308	185551.9	5.2
AAPL	2010-04	0.1110	NaN	0.1484	8.393	213100.4	6.0
AAPL	2010-05	-0.0161	NaN	-0.0161	9.325	237584.9	6.0
AAPL	2010-06	-0.0208	NaN	-0.0161	9.174	233737.7	5.9

How Momentum Is Calculated

Momentum is the cumulative return from month $t-13$ to month $t-2$:

$$\text{momentum}_t = \frac{\text{closeadj}_{t-2}}{\text{closeadj}_{t-13}} - 1$$

- Skips the most recent month ($t-1$): short-term reversal contaminates the signal.
- Uses split- and dividend-adjusted prices (`closeadj`).
- Requires 13 months of price history, so the first 12 months are NaN.
- All calculations are **grouped by ticker** — never mixing prices across stocks.

Variables from the Returns Step

- `ticker`, `month` – identifiers
- `return` – monthly return (decimal: $0.05 = 5\%$)
- `momentum` – 12-month return skipping the most recent month
- `lagged_return` – prior month's return
- `close` – split-adjusted closing price (end of month)
- `marketcap` – market cap in thousands (**shifted by 1 month**)
- `size` – Nano/Micro/Small/Mid/Large/Mega-Cap
- `sector`, `industry` – classifications

Step 2: Fetch Fundamentals

Precomputed vs. Calculated Variables

Rule: Before calculating any ratio, check if it already exists in the database.

Precomputed in DAILY: marketcap, pb, pe, ps, ev, evebit, evebitda

Precomputed in SF1: roe, roa, grossmargin, netmargin, de (leverage),
assetturnover, currentratio

Must calculate:

- Growth rates: asset growth, revenue growth
- Custom ratios: gross profit / assets, book-to-market

Fetching Fundamental Data

Prompt: "Get annual fundamentals from SF1: equity, assets, gross profit, and the precomputed ratios roe, grossmargin, assetturnover, and de. Calculate asset growth and gross-profit-to-assets. Save as fundamentals.parquet."

Claude uses the `fetch-fundamentals` skill, which:

1. Checks which variables are precomputed (fetches, doesn't calculate)
2. Queries SF1 for annual data (`dimension='ARY'`)
3. Calculates growth rates **before** merging (critical!)
4. Queries DAILY for valuation ratios (pb, pe) if requested
5. Shifts DAILY variables by 1 month

Why Calculate Growth Rates Before Merging?

After merging, fundamentals are **forward-filled**: each filing's values repeat every month until the next filing.

The Problem

If you calculate `pct_change()` after forward-fill, consecutive months with the same value produce **zero growth** — which is wrong.

Solution: Calculate growth from the raw SF1 data (one row per filing), where consecutive rows are consecutive filings.

```
df_fund['asset_growth'] = df_fund.groupby('ticker')  
                               ['assets'].pct_change()
```

Step 3: Merge Returns & Fundamentals

Merging the Datasets

Prompt: “Merge `monthly.parquet` with `fundamentals.parquet` and save as `merged.parquet`.”

Claude uses the `merge-data` skill, which:

1. Aligns fundamentals to the first month **after** the SEC filing date
2. Merges on (ticker, month)
3. Forward-fills fundamentals within each ticker
4. Shifts all SF1 variables by 1 month (look-ahead bias)
5. Shifts `close` to represent the prior month's price
6. Applies data quality filters

Merged Data: What You Get

```
>>> df = pd.read_parquet("merged.parquet")
>>> df.shape
(886460, 20)
>>> df[df.ticker=="AAPL"].dropna().head(5)
```

ticker	month	return	momentum	close	marketcap	pb	roe	grossmargin
AAPL	2012-02	0.1883	0.1936	16.303	425612.0	4.7	0.396	0.405
AAPL	2012-03	0.1053	0.2924	19.373	505758.5	5.6	0.396	0.405
AAPL	2012-04	-0.0260	0.5564	21.413	559015.5	6.2	0.396	0.405
AAPL	2012-05	-0.0107	0.7123	20.857	546072.5	5.3	0.396	0.405
AAPL	2012-06	0.0109	0.6789	20.633	540207.8	5.3	0.396	0.405

Note: roe and grossmargin are constant across months — they reflect the most recent annual filing, forward-filled and shifted.

Avoiding Look-Ahead Bias

The Core Problem

We want each row to represent: **what we knew at the start of the month** paired with **what happened during the month**.

Look-Ahead Bias

Using information that was **not yet available** at the time of the investment decision. Backtests that suffer from this overstate performance — sometimes dramatically.

How the Skills Handle It

- **Market cap and DAILY ratios** (pb, pe, ps): shifted by 1 month after fetching. The value for January is from end of December.
- **Close price**: shifted by 1 month after merging. Represents prior month's closing price.
- **SF1 fundamentals** (roe, de, etc.): aligned to the first month *after* the SEC filing date, then forward-filled, then shifted by 1 month.
- **Momentum**: uses prices through month $t-2$, so it is known at the start of month t .
- **Lagged return**: prior month's return, known at the start of the current month.

Summary of Shifts

Variable	Source	Shifted By
marketcap, pb, pe, ps	DAILY	1 month after fetching
close	SEP	1 month after merging
roe, de, grossmargin, ...	SF1	filing date + 1 month after merge
momentum	SEP	built-in (uses $t-13$ to $t-2$)
lagged_return	SEP	built-in (uses $t-1$)
return	SEP	not shifted (target variable)

return is the **only variable that is not known at the start of the month**. It is what we are trying to predict.

Filters & Standardization

Data Quality Filters

The merge-data skill applies two filters automatically:

- **Price filter:** drop rows with `close < $5.00`
- **Missing data:** drop rows with any NaN values

Before filters:	886,460 rows	9,375 tickers
After filters:	589,006 rows	6,825 tickers

- Must filter out low-price stocks. Infeasible for equally weighted portfolios and distort portfolio returns.
- Use CQA Competition filter: \$5.60?
- Example: Tell Claude to read the data and drop all rows with $\text{close} \leq \text{price threshold}$. Sort into quintiles each month on momentum and lagret and compute average returns.

Market Cap Filters

- If we were managing a fund, we would specify the universe of stocks in our prospectus.
- We would likely include some market cap filter: large cap = S&P 500, large and midcap = Russell 1000, smallcap = Russell 2000, etc.
- It is hard for large funds to trade microcaps, so they will exclude them.

Standardizing Features

- Many models are highly affected by outliers.
- Standardizing predictive variables is common. For example, subtract mean and divide by standard deviation (z-score).
- Ranking 1 through n is another way to standardize. Can scale to 0 to 1 or compute z-scores of ranks.
- We should do this each month rather than for the full sample.
- For “good” features (like football scores), rank from low to high. For “bad” features (like golf scores), rank from high to low. Then a high rank is always good.

Composite Ranks

The Quality Minus Junk paper computes quality as follows:

1. Compute z-scores of features each month.
2. Group features into three categories: profitability, growth, and safety.
3. Within each category, add z-scores and compute the z-score of the sum each month.
4. Add the three category z-scores and compute the z-score of the sum each month.

The Complete Workflow

End-to-End Prompts

1. "Get monthly returns for all stocks from 2010 onward. Save as `monthly.parquet`."
2. "Get annual fundamentals from SF1: equity, assets, gp, and the precomputed ratios roe, grossmargin, assetturnover, de. Calculate asset growth and gp/assets. Save as `fundamentals.parquet`."
3. "Merge monthly returns with fundamentals. Save as `merged.parquet`."

Three prompts produce a complete, bias-free panel dataset ready for portfolio analysis or machine learning.

The Final Dataset

```
>>> df = pd.read_parquet("merged.parquet")
>>> df.shape
(589006, 17)
>>> df.columns
['ticker', 'month', 'return', 'momentum', 'lagged_return',
 'close', 'marketcap', 'pb', 'asset_growth', 'roe',
 'gp_to_assets', 'grossmargin', 'assetturnover', 'leverage',
 'sector', 'industry', 'size']
>>> df.ticker.nunique()
6825
```

589,006 stock-months \times 17 variables, spanning 2011–2025, covering 6,825 stocks (including delisted).