**China Index Data Platform:**
**End-to-End Pipeline on Snowflake, dbt, Airflow, and Streamlit**

Author: Zhe Shen

## 1. Introduction

This document describes the design and implementation of the China Index Data Platform, a small end-to-end analytics warehouse for Chinese equity index daily prices.

The goal is to build a realistic but compact data pipeline that:

- Ingests daily OHLCV data for China A-share indices from AWS S3 bucket,
- Stores the data in Snowflake,
- Uses dbt to create clean staging, dimension, and fact tables,
- Uses Airflow to orchestrate a daily workflow, and
- Exposes the results in a simple Streamlit dashboard for analysis.

The main business questions are:

- How has a key index (or a small set of indices) performed over time in terms of cumulative returns and major drawdowns?
- How does volatility change over time, and when do we see calm vs. turbulent periods?
- What does the distribution of daily returns look like, and when do large positive or negative moves occur?
- Are there simple seasonality patterns by month or by weekday?

## 2. Data and Business Questions

2.1 Dataset

The original dataset contains daily OHLCV data (open, high, low, close, volume, turnover) for many China A-share indices over the period 2018-2023.
Each record corresponds to one index on one trading day and includes:

- trade_date - trading date,
- index_code - index identifier (e.g., SSE Composite, CSI 300),
- open, high, low, close - daily OHLC prices,
- volume - trading volume,
- turnover - trading value.

To keep the project scope and Snowflake storage manageable, the data is pre-filtered locally:

- Keep only a small set of major indices (e.g., SSE Composite Index and CSI 300),
- Restrict to a recent time range (e.g., 2019-2023),
- Keep only the columns needed for modeling.

The filtered result is saved as a single CSV file: index_daily_2019_2023_filtered.csv

This file is then uploaded to S3 and becomes the source for the end-to-end pipeline.

2.2 Business Questions
The platform is designed to support the following questions:

1. Performance and Drawdowns:
- How has each chosen index performed over time?
- What are the major drawdown periods and how deep were the losses?

2. Volatility and Extreme Days:
- How does rolling volatility evolve over time?
- When do we see large daily moves (big up days and big down days)?

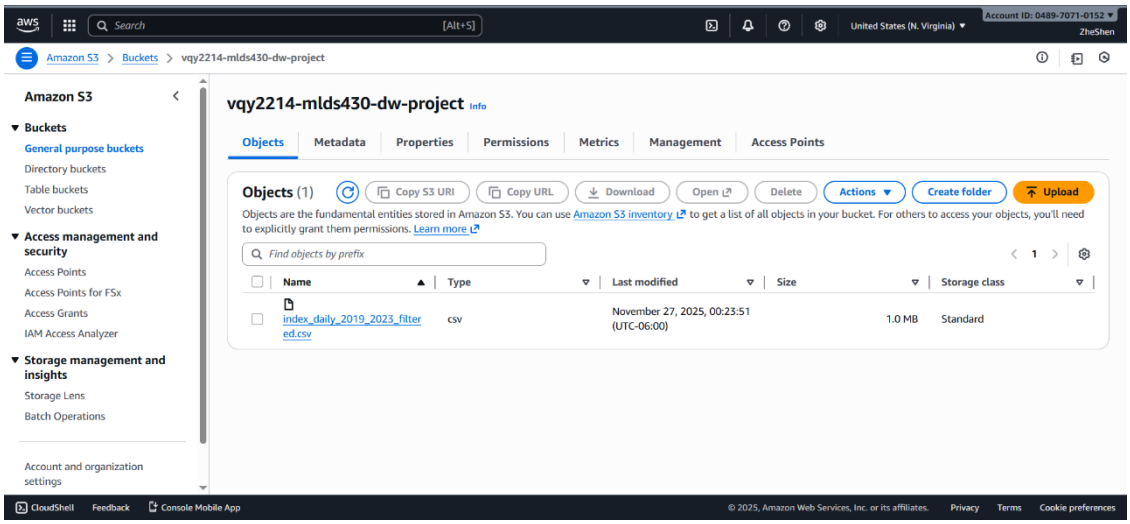
3. Distribution of Returns:
- What is the shape of the daily return distribution?
- Are tails symmetric, and how common are large absolute moves?

4. Simple Seasonality:
- Are there clear differences in average returns by month?
- Are certain weekdays consistently stronger or weaker?

These questions motivate the design of the fact table and the metrics computed in dbt, as well as the visualizations in the Streamlit app.

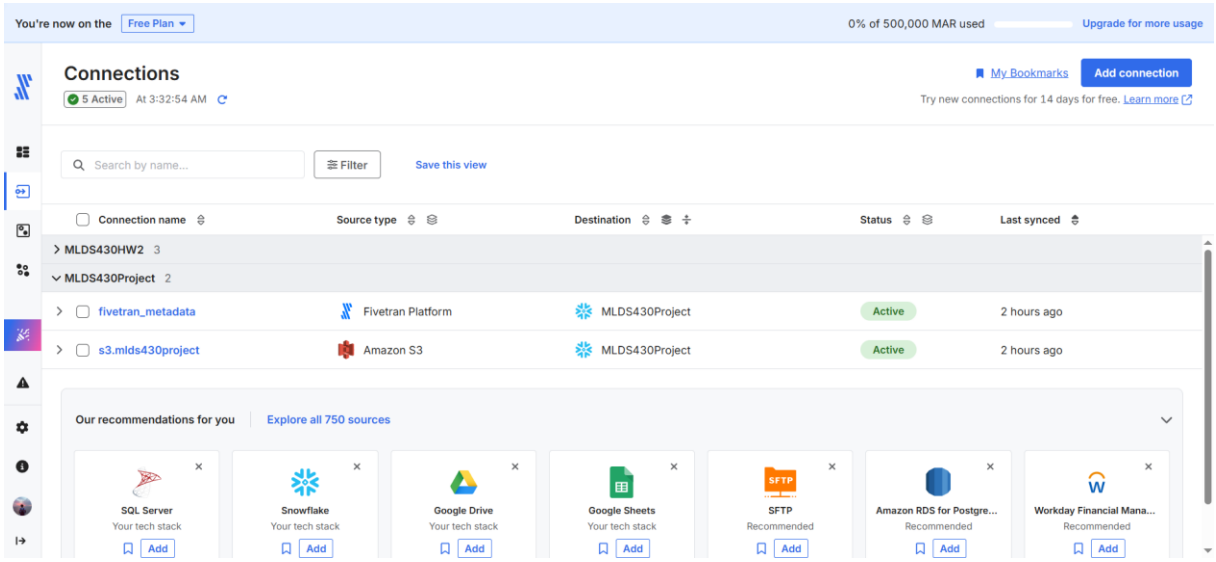
**3. Architecture Overview**
The project follows a modern data stack pattern:
S3 → Fivetran → Snowflake → dbt → Airflow → Streamlit

3.1 High-Level Flow
1. S3 (Source Layer)
A pre-filtered CSV file containing index daily OHLCV data is stored in S3. Each row contains trade_date, index_code, open, high, low, close, volume, and turnover.



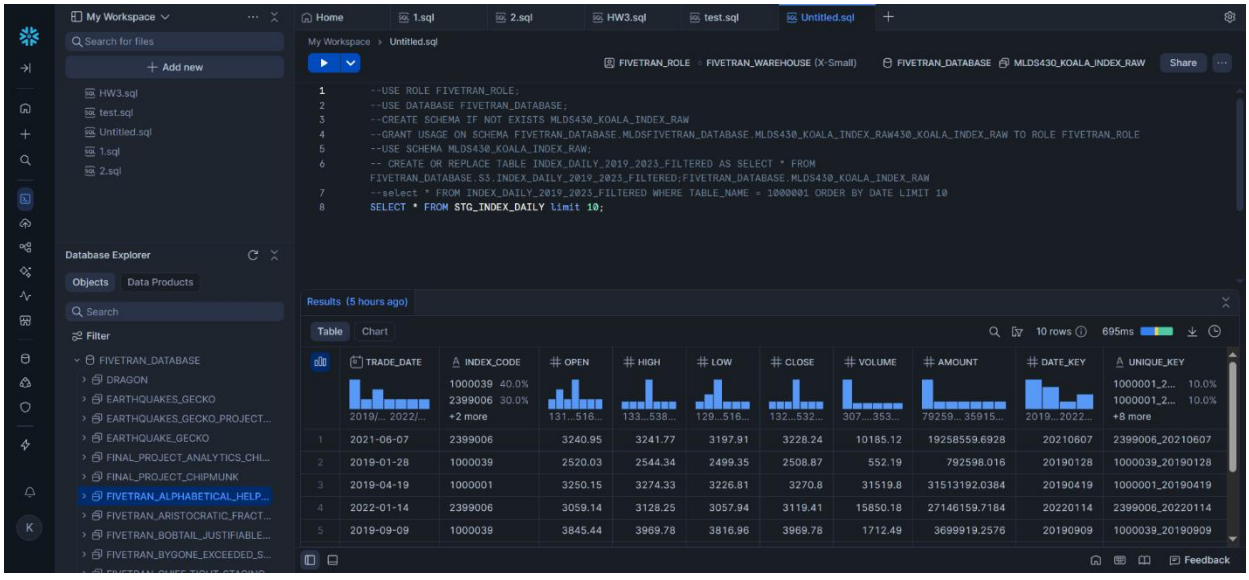
2. Fivetran S3 → Snowflake (Raw Layer)

A Fivetran S3 connector ingests the CSV and loads it into Snowflake as a raw table in the schema MLDS430_KOALA_INDEX_RAW. Fivetran handles parsing, schema inference, and incremental loads.




3. Snowflake (Warehouse Layer)

The raw table in MLDS430_KOALA_INDEX_RAW is the single source of truth for dbt. No modeling is done here; Snowflake acts as the data warehouse and compute engine.




4. dbt Transformations (Model Layer)

A dbt project (index_dbt) defines:

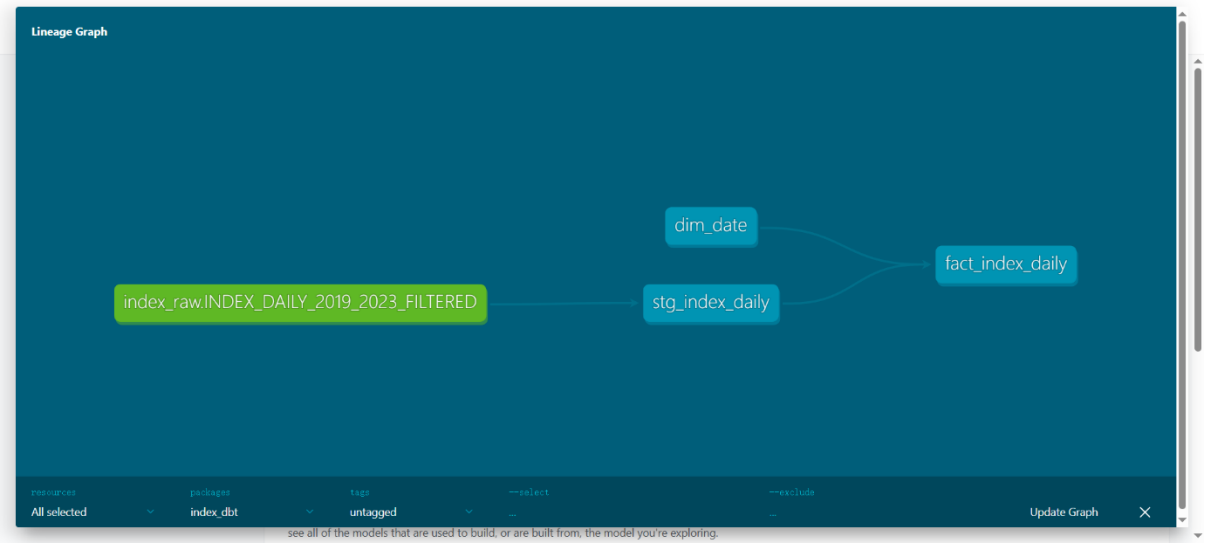
- stg_index_daily - a clean staging model,
- dim_date - a date dimension,
- fact_index_daily - a fact/mart table with returns and rolling metrics.

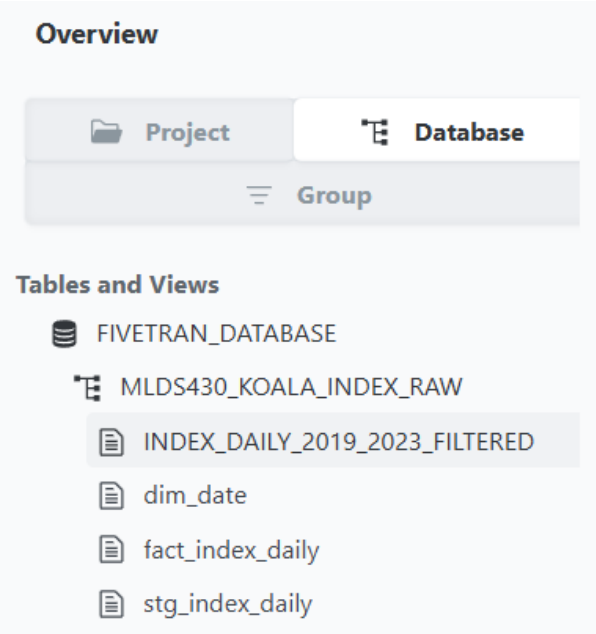
dbt also manages tests and basic documentation.

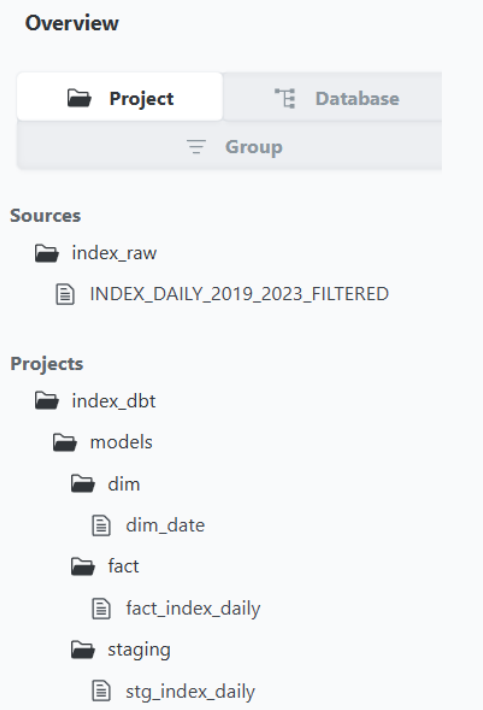
```
D:\MLDS430-Project-China-Index\index_dbt>dbt run
09:36:57  Running with dbt=1.10.15
09:36:58  Registered adapter: snowflake=1.10.3
09:37:00  Found 3 models, 8 data tests, 1 source, 494 macros
09:37:00
09:37:00  Concurrency: 4 threads (target='dev')
09:37:00
09:37:09  1 of 3 START sql table model MLDS430_KOALA_INDEX_RAW.dim_date ..... [RUN]
09:37:09  2 of 3 START sql table model MLDS430_KOALA_INDEX_RAW.stg_index_daily ..... [RUN]
09:37:13  1 of 3 OK created sql table model MLDS430_KOALA_INDEX_RAW.dim_date ..... [SUCCESS 1 in 3.41s]
09:37:13  2 of 3 OK created sql table model MLDS430_KOALA_INDEX_RAW.stg_index_daily ..... [SUCCESS 1 in 3.61s]
09:37:13  3 of 3 START sql table model MLDS430_KOALA_INDEX_RAW.fact_index_daily ..... [RUN]
09:37:16  3 of 3 OK created sql table model MLDS430_KOALA_INDEX_RAW.fact_index_daily ..... [SUCCESS 1 in 3.00s]
09:37:17
09:37:17  Finished running 3 table models in 0 hours 0 minutes and 17.42 seconds (17.42s).
09:37:17
09:37:17  Completed successfully
09:37:17
09:37:17  Done. PASS=3 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=3

D:\MLDS430-Project-China-Index\index_dbt>dbt test
09:37:42  Running with dbt=1.10.15
09:37:43  Registered adapter: snowflake=1.10.3
09:37:45  Found 3 models, 8 data tests, 1 source, 494 macros
09:37:45
09:37:45  Concurrency: 4 threads (target='dev')
09:37:45
09:37:47  1 of 8 START test not_null_dim_date_date_key ..... [RUN]
09:37:47  2 of 8 START test not_null_fact_index_daily_date_key ..... [RUN]
09:37:47  3 of 8 START test not_null_fact_index_daily_index_code ..... [RUN]
09:37:47  4 of 8 START test not_null_stg_index_daily_index_code ..... [RUN]
09:37:50  4 of 8 PASS not_null_stg_index_daily_index_code ..... [PASS in 2.61s]
09:37:50  1 of 8 PASS not_null_dim_date_date_key ..... [PASS in 2.62s]
09:37:50  5 of 8 START test not_null_stg_index_daily_trade_date ..... [RUN]
09:37:50  6 of 8 START test not_null_stg_index_daily_unique_key ..... [RUN]
09:37:50  2 of 8 PASS not_null_fact_index_daily_date_key ..... [PASS in 2.78s]
09:37:50  7 of 8 START test unique_dim_date_date_key ..... [RUN]
09:37:50  6 of 8 PASS not_null_stg_index_daily_unique_key ..... [PASS in 0.18s]
09:37:50  8 of 8 START test unique_stg_index_daily_unique_key ..... [RUN]
09:37:50  3 of 8 PASS not_null_fact_index_daily_index_code ..... [PASS in 2.84s]
09:37:50  5 of 8 PASS not_null_stg_index_daily_trade_date ..... [PASS in 0.29s]
09:37:50  8 of 8 PASS unique_stg_index_daily_unique_key ..... [PASS in 0.18s]
09:37:50  7 of 8 PASS unique_dim_date_date_key ..... [PASS in 0.29s]
09:37:51
09:37:51  Finished running 8 data tests in 0 hours 0 minutes and 6.39 seconds (6.39s).
09:37:51
09:37:51  Completed successfully
09:37:51
09:37:51  Done. PASS=8 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=8

D:\MLDS430-Project-China-Index\index_dbt>
```

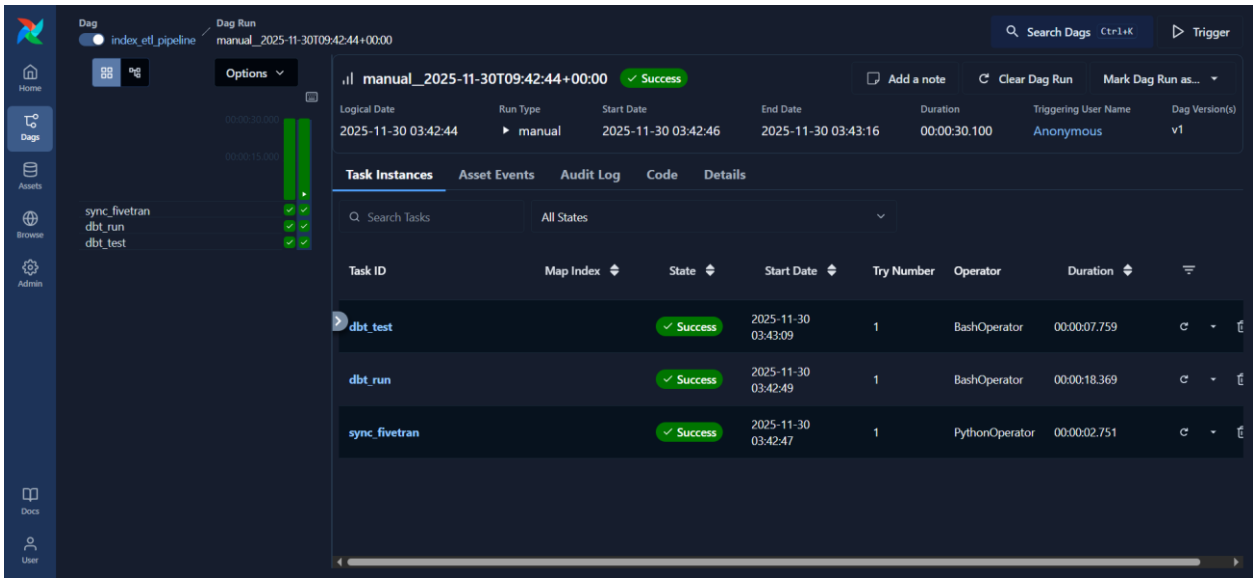





5. Airflow Orchestration

An Astro-based Airflow project defines a DAG index_etl_pipeline with three tasks: sync_fivetran, dbt_run, and dbt_test. The DAG can be scheduled daily to keep the models up to date.




6. Streamlit Analytics

A Streamlit app connects to Snowflake, queries fact_index_daily, and provides interactive charts and tables for performance, drawdowns, volatility, extreme days, and seasonality.

# China Index Analytics Dashboard

Multi-index analytics for major Chinese equity indices: levels, cumulative return, drawdown, volatility and basic seasonality.

Use the filters on the left to change indices and date range. All charts will update automatically.

## 📊 Dashboard Filters

### Index Selection

Select index

1000001 - SSE C… ×

1000032 - SSE En… ×

### Date Range

Start date

2019/01/01

End date

2023/12/31

**Note:** Data sourced from `fact_index_daily` table in Snowflake.

**Selected indices**

SSE Composite Index | SSE Energy

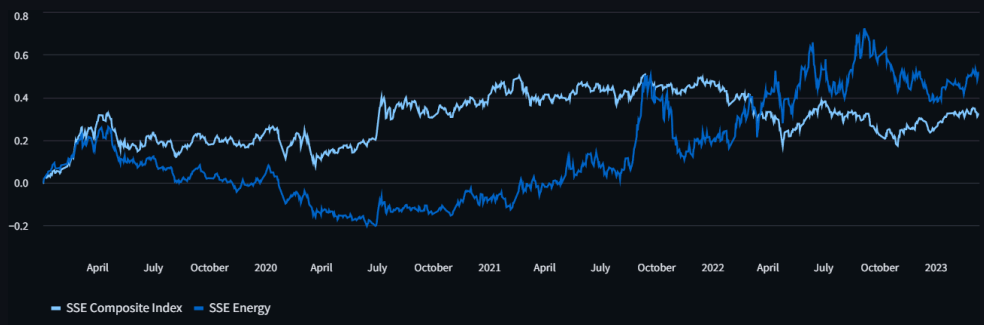
Performance & Drawdown | Volatility & Seasonality

## 📈 Overall Index Performance

Index closing levels over time for the selected indices.


## 🚀 Cumulative Return

Cumulative return since the first date in the selected range.




## 📉 Drawdown from Peak


Drawdown = Current cumulative return vs. historical peak (per index).

# China Index Analytics Dashboard

Multi-index analytics for major Chinese equity indices: levels, cumulative return, drawdown, volatility and basic seasonality.

Use the filters on the left to change indices and date range. All charts will update automatically.

### 📊 Dashboard Filters

**Index Selection**

Select index

1000001 - SSE C… ×

1000032 - SSE En… ×

**Date Range**

Start date

2019/01/01

End date

2023/12/31

**Note:** Data sourced from `fact_index_daily` table in Snowflake.

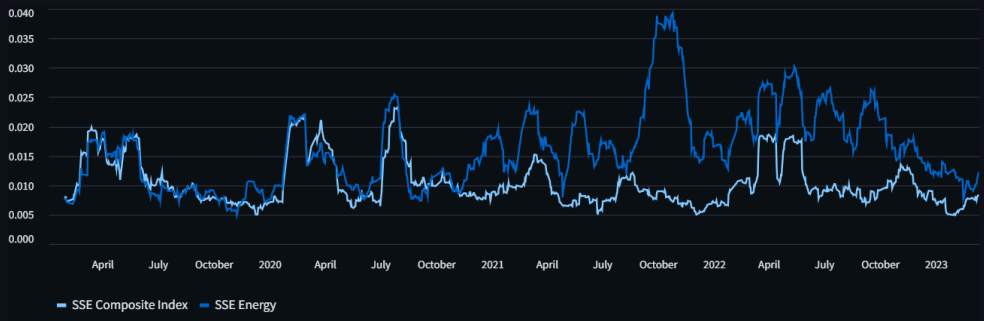
**Selected indices**

SSE Composite Index | SSE Energy

Performance & Drawdown | Volatility & Seasonality

## 🌪️ 20-day Rolling Volatility

Standard deviation of daily returns over a 20-day rolling window.




## 🔝 Top 10 Up & Down Days

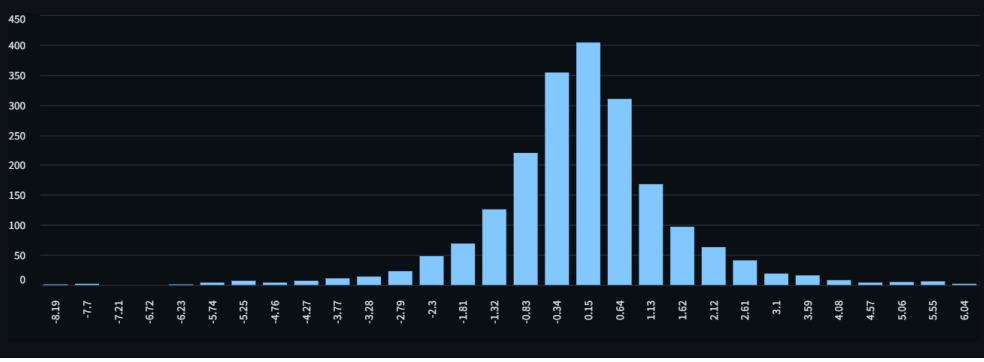
**Top 10 Up Days**

| | trade_date | year | month | index_code | index_name | daily_return_% | close |
|---|---|---|---|---|---|---|---|
| 1381 | 2020-07-06 00:00:00 | 2020 | 7 | 1000032 | SSE Energy | 6.29 | 1112.26 |
| 1532 | 2021-02-18 00:00:00 | 2021 | 2 | 1000032 | SSE Energy | 6.05 | 1163.79 |
| 1672 | 2021-09-09 00:00:00 | 2021 | 9 | 1000032 | SSE Energy | 5.74 | 1781.67 |
| 364 | 2020-07-06 00:00:00 | 2020 | 7 | 1000001 | SSE Composite Index | 5.71 | 3332.88 |
| 1674 | 2021-09-13 00:00:00 | 2021 | 9 | 1000032 | SSE Energy | 5.69 | 1822.08 |
| 33 | 2019-02-25 00:00:00 | 2019 | 2 | 1000001 | SSE Composite Index | 5.6 | 2961.28 |
| 1679 | 2021-09-22 00:00:00 | 2021 | 9 | 1000032 | SSE Energy | 5.39 | 1819.33 |
| 1683 | 2021-09-28 00:00:00 | 2021 | 9 | 1000032 | SSE Energy | 5.31 | 1761.85 |
| 1910 | 2022-09-05 00:00:00 | 2022 | 9 | 1000032 | SSE Energy | 5.29 | 2048.79 |
| 1692 | 2021-10-18 00:00:00 | 2021 | 10 | 1000032 | SSE Energy | 5.25 | 1710.44 |

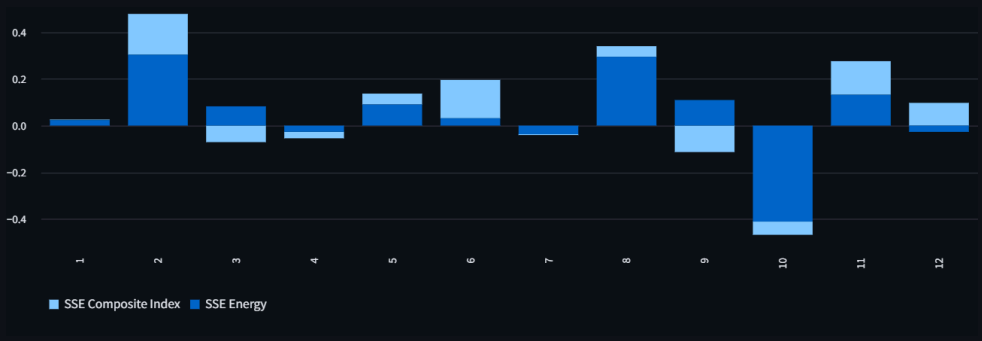
**Top 10 Down Days**

| | trade_date | year | month | index_code | index_name | daily_return_% | close |
|---|---|---|---|---|---|---|---|
| 1277 | 2020-02-03 00:00:00 | 2020 | 2 | 1000032 | SSE Energy | -8.44 | 1094.39 |
| 260 | 2020-02-03 00:00:00 | 2020 | 2 | 1000001 | SSE Composite Index | -7.72 | 2746.61 |
| 1792 | 2022-03-15 00:00:00 | 2022 | 3 | 1000032 | SSE Energy | -7.6 | 1469.8 |
| 1700 | 2021-10-28 00:00:00 | 2021 | 10 | 1000032 | SSE Energy | -6.21 | 1466.82 |
| 1855 | 2022-06-20 00:00:00 | 2022 | 6 | 1000032 | SSE Energy | -5.75 | 1767.27 |
| 79 | 2019-05-06 00:00:00 | 2019 | 5 | 1000001 | SSE Composite Index | -5.58 | 2906.46 |
| 1684 | 2021-09-29 00:00:00 | 2021 | 9 | 1000032 | SSE Energy | -5.57 | 1663.67 |
| 1694 | 2021-10-20 00:00:00 | 2021 | 10 | 1000032 | SSE Energy | -5.55 | 1626.56 |
| 1689 | 2021-10-13 00:00:00 | 2021 | 10 | 1000032 | SSE Energy | -5.38 | 1577.48 |
| 1867 | 2022-07-06 00:00:00 | 2022 | 7 | 1000032 | SSE Energy | -5.38 | 1809.88 |

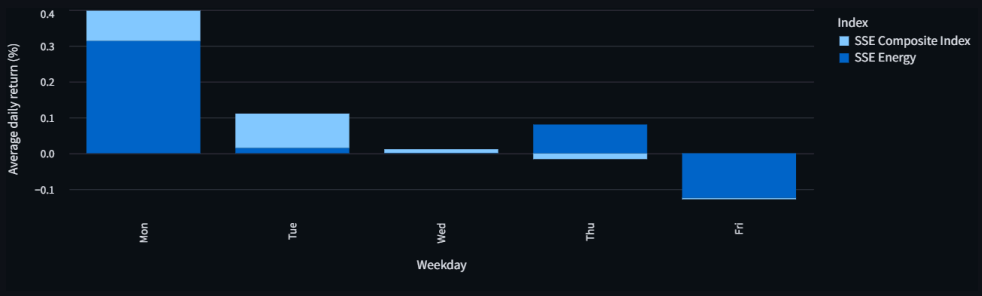
## 📊 Distribution of Daily Returns



## Seasonality by Month (Average Daily Return)




## Seasonality by Weekday (Average Daily Return)



**4. Ingestion Layer: S3 + Fivetran + Snowflake**

4.1 S3 Source

The pre-filtered CSV file (e.g., index_daily_2019_2023_filtered.csv) is uploaded to an S3 bucket.

This file contains only the indices and date range needed for the project, which keeps the pipeline simple and efficient.

4.2 Fivetran S3 Connector

A Fivetran S3 connector is configured to:

- Read from the S3 bucket and path,
- Parse CSV files with a header row,
- Load them into Snowflake as a raw table.

The target configuration is:

- Database: the course Snowflake database,
- Schema: MLDS430_KOALA_INDEX_RAW,
- Table: a raw table such as INDEX_DAILY_2019_2023_FILTERED.

Fivetran can also be triggered via API in Airflow to force a manual sync, which is useful during development or demos.

4.3 Raw Table in Snowflake

After a Fivetran sync, Snowflake contains a raw table with one row per index_code and trade_date. The columns remain close to the original dataset. This table is referenced by dbt as a source and is not modified directly; all modeling and logic live in dbt models.

5. Transformation Layer: dbt Models

The dbt project index_dbt defines sources, models, tests, and configuration for Snowflake.

5.1 Source Definition

A dbt source is declared to reference the Fivetran raw table in MLDS430_KOALA_INDEX_RAW. This makes it easy to manage the connection and keeps SQL models focused on transformation rather than connection details.

5.2 stg_index_daily - Staging Model

Purpose: Clean and standardize the raw index data before further modeling.

Key tasks:

- Standardize column names to a consistent set: trade_date, index_code, open, high, low, close, volume, turnover.
- Cast trade_date to a proper date type.
- Cast all numeric columns to numeric types.

- Optionally filter to the subset of target indices (for example, SSE Composite and CSI 300).
- Create a date_key (YYYYMMDD as integer).
- Create a unique_key per index and date (e.g., concatenation of index_code and date_key).

Tests:
- not_null on trade_date, index_code, close, and unique_key.
- unique on unique_key.

This model is the clean base for all downstream metrics.

5.3 dim_date - Date Dimension
Purpose: Provide a calendar dimension for time-based analysis and seasonality.

Key tasks:
- Generate a continuous sequence of calendar dates covering the trading period (e.g., 2019–2023).
- Derive attributes such as:
  - year,
  - month,
  - day,
  - quarter,
  - weekday (e.g., Monday-Friday codes).
- Create a date_key (YYYYMMDD) that matches the one in stg_index_daily.

This table is used to filter by time period and to group returns by month or weekday for seasonality charts.

5.4 fact_index_daily - Fact / Mart Model
Purpose: Provide the main analytic table at the index-day grain.

Grain:
- One row per trading day per index.

Keys:
- date_key,
- index_code (and/or unique_key).

Measures and derived metrics:
- Base columns: open, high, low, close, volume, turnover.
- Daily return: Computed as close / lag(close) - 1 within each index_code, ordered by trade_date.

- Cumulative return: Constructed from cumulative log returns or a running product of (1 + daily_return) over time.
- Rolling 20-day average daily return: 20-day moving average of daily_return for each index.
- Rolling 20-day volatility: 20-day rolling standard deviation of daily_return.

All of these are implemented using SQL window functions (lag, avg, stddev_samp over partitions by index_code and ordered by trade_date). The model joins to dim_date on date_key to attach calendar attributes, enabling time-based grouping and filtering.

**6. Orchestration Layer: Airflow DAG**
The orchestration is implemented in an Astro-based Airflow project. The main DAG is named index_etl_pipeline.

6.1 DAG Tasks and Dependencies
The DAG contains three core tasks:

1. sync_fivetran
- A PythonOperator that reads environment variables(FIVETRAN_API_KEY, FIVETRAN_API_SECRET, FIVETRAN_CONNECTOR_ID) and calls the Fivetran REST API to force an S3 → Snowflake sync.

2. dbt_run
- A BashOperator that runs dbt run inside the Airflow container, targeting the index_dbt project.

3. dbt_test
- A BashOperator that runs dbt test after dbt_run completes.

The dependency chain is:
sync_fivetran >> dbt_run >> dbt_test

The DAG is scheduled to run daily, with catchup=False so that Airflow does not try to backfill historical dates. This reflects a realistic daily refresh in production while keeping the setup simple for the course project.

6.2 DAG Testing
A small test module (e.g., tests/test_index_pipeline_dag.py) checks:
- The DAG id and that all expected tasks exist,
- The dependency order between sync_fivetran, dbt_run, and dbt_test,
- The behavior of the Fivetran trigger function using mocks and environment variables.

These tests help ensure the DAG is correctly wired and make future changes safer.

7. Analytics Layer: Streamlit Dashboard
Instead of Tableau, the project uses Streamlit to build a lightweight, interactive analytics front end.

7.1 Connection and Filters
The app:
- Connects to Snowflake using the same private key configuration as dbt (SNOWFLAKE_PRIVATE_KEY_PATH, SNOWFLAKE_PRIVATE_KEY_PASSPHRASE and related settings).
- Queries the fact_index_daily table based on user selections.
- Provides a sidebar where the user can:
  - Select one or more index codes,
  - Choose a date range for analysis.

When the selections change, the app reloads the data from Snowflake into pandas and recomputes or re-displays the charts.

7.2 Visualizations

The Streamlit dashboard implements the visualization plan that was originally written for Tableau, adapted into interactive tabs:
1. Performance & Drawdown:
- Line chart of index closing level over time,
- Line chart of cumulative return over time,
- Simple drawdown curve showing the percent drop from the running peak,
- Optional summary metrics (e.g., cumulative return and max drawdown for the selected period).
2. Volatility & Extreme Days
- Line chart of 20-day rolling volatility of daily returns,
- Daily return chart, with visual emphasis on days where |daily_return| exceeds a threshold (e.g., 3%),
- Tables listing the top N up days and top N down days with dates and return values.

3. Seasonality
- Charts showing average daily returns grouped by month and by weekday, using dim_date attributes,
- Used to check for simple seasonal patterns in returns or volatility.

Together, these views allow the user to explore:
- Long-term performance and drawdowns,
- Changes in volatility over time,
- Extreme events (shock days),
- Basic seasonality patterns.

## 8. Implementation Notes and Environment

Key implementation details include:

- Authentication:
  Snowflake authentication uses private keys, with paths and passphrases stored in environment variables for both dbt and Streamlit.
- Local development:
  - dbt is run locally via dbt run / dbt test,
  - Airflow is started via astro dev start in the Airflow/ directory,
  - Streamlit is started via streamlit run app.py in the streamlit/ directory.
- Performance:
  The data volume (a few indices over a few years) is moderate, so Snowflake can easily handle transformations and queries. Window functions for returns and rolling volatility are executed in Snowflake rather than in Python, which keeps the pipeline efficient.

## 9. Conclusion

The China Index Data Platform demonstrates an end-to-end analytics pipeline for equity index data:

- Ingestion from S3 into Snowflake via Fivetran,
- Modeling with dbt staging, dimension, and fact tables,
- Orchestration with a daily Airflow DAG,
- Interactive analytics using a Streamlit dashboard.

The project shows how the tools from MLDS-430 course can be combined in a practical workflow to answer real business questions about performance, risk, and seasonality for Chinese stock indices. The design is simple but extensible, and it can serve as a template for larger, more complex data platforms in the future.