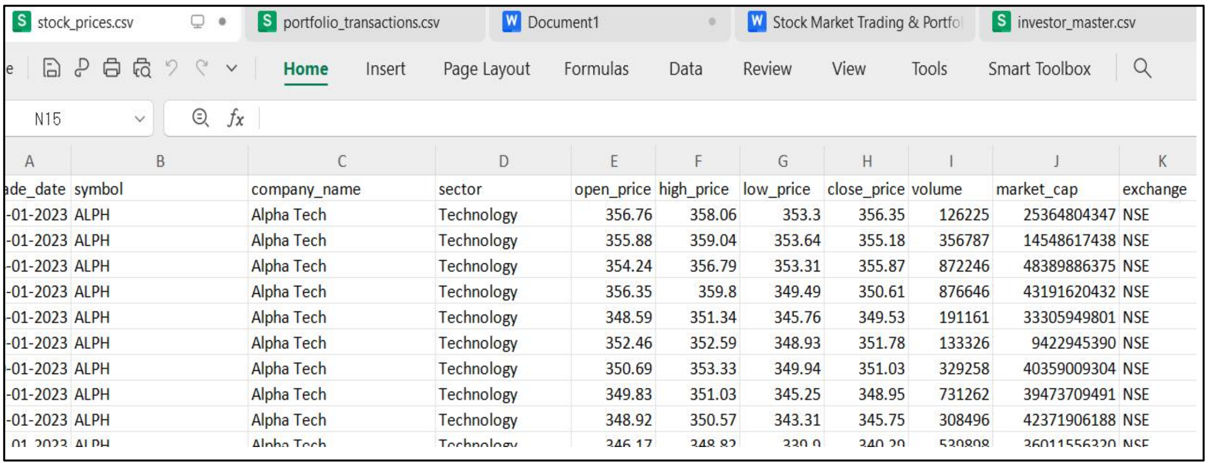
Stock Market Trading & Portfolio Performance Analytics Platform

1.Dataset Creation:

a) stock_prices.csv

This dataset contains daily stock market price information for multiple companies across different sectors. This data is useful for all market analytics, such as returns, volatility, trends, and sector performance.It has around 5201 records and each record represents one stock traded on one specific day.

Column	Explanation
trade_date	Trading date (used for time-series & incremental processing)
symbol	Unique stock identifier
company_name	Name of the company
sector	Business sector (Technology, Finance, Energy, etc.)
open_price	Price at market open
high_price	Highest price during the day
low_price	Lowest price during the day
close_price	Price at market close
volume	Number of shares traded
market_cap	Total company valuation
exchange	Stock exchange (simulated: NSE)



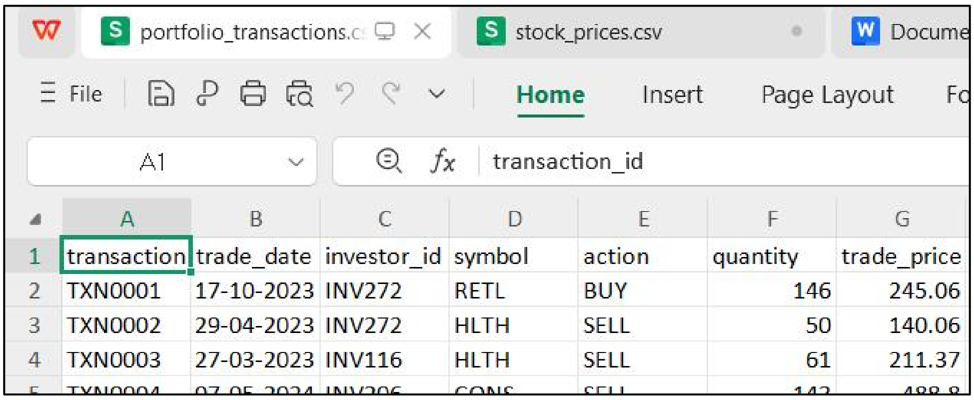
ade_date	symbol	company_name	sector	open_price	high_price	low_price	close_price	volume	market_cap	exchange
-01-2023	ALPH	Alpha Tech	Technology	356.76	358.06	353.3	356.35	126225	25364804347	NSE
-01-2023	ALPH	Alpha Tech	Technology	355.88	359.04	353.64	355.18	356787	14548617438	NSE
-01-2023	ALPH	Alpha Tech	Technology	354.24	356.79	353.31	355.87	872246	48389886375	NSE
-01-2023	ALPH	Alpha Tech	Technology	356.35	359.8	349.49	350.61	876646	43191620432	NSE
-01-2023	ALPH	Alpha Tech	Technology	348.59	351.34	345.76	349.53	191161	33305949801	NSE
-01-2023	ALPH	Alpha Tech	Technology	352.46	352.59	348.93	351.78	133326	9422945390	NSE
-01-2023	ALPH	Alpha Tech	Technology	350.69	353.33	349.94	351.03	329258	40359009304	NSE
-01-2023	ALPH	Alpha Tech	Technology	349.83	351.03	345.25	348.95	731262	39473709491	NSE
-01-2023	ALPH	Alpha Tech	Technology	348.92	350.57	343.31	345.75	308496	42371906188	NSE

b) portfolio_transactions.csv

This dataset simulates real investor trading activity. In real markets, portfolios are not stored directly. Instead, systems store BUY and SELL transactions, and the portfolio is derived dynamically from these transactions. Each row represents a trade by an investor.

Column	Explanation
transaction_id	Unique trade identifier

Column	Explanation
trade_date	Date of transaction
investor_id	Who executed the trade
symbol	Stock traded
action	BUY or SELL
quantity	Number of shares
trade_price	Price at which trade was executed

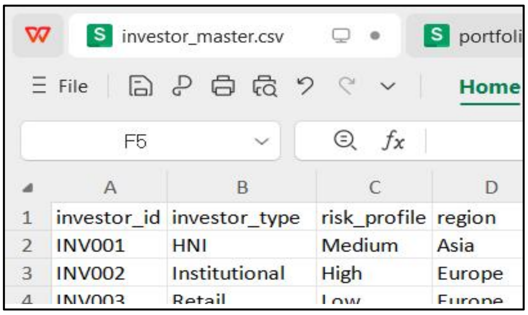


	A	B	C	D	E	F	G
1	transaction	trade_date	investor_id	symbol	action	quantity	trade_price
2	TXN0001	17-10-2023	INV272	RETL	BUY	146	245.06
3	TXN0002	29-04-2023	INV272	HLTH	SELL	50	140.06
4	TXN0003	27-03-2023	INV116	HLTH	SELL	61	211.37

c) investor_master.csv

This dataset contains investor metadata. Usually, investor attributes are stored separately and joined with transactional data to perform risk profiling and segmentation analytics.

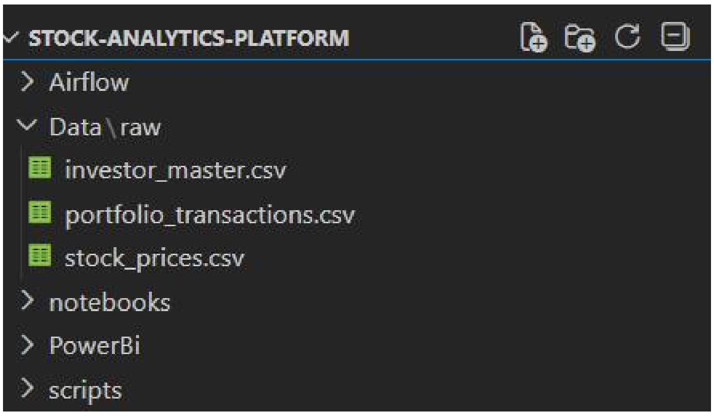
Column	Explanation
investor_id	Unique investor identifier
investor_type	Retail / HNI / Institutional
risk_profile	Risk appetite (Low / Medium / High)
region	Geographic region



	A	B	C	D
1	investor_id	investor_type	risk_profile	region
2	INV001	HNI	Medium	Asia
3	INV002	Institutional	High	Europe

Step 1.1: Data ingestion

The data files are saved in the folder : STOCK-ANALYSIS-PLATFORM/Data/ raw



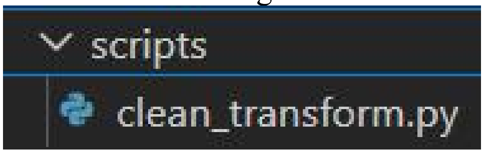
Step 1.2: Data Cleaning & Feature Engineering

The data cleaning and feature engineering phase was performed to transform raw stock market datasets into a form that is suitable for large-scale processing and visualization. The raw datasets contained missing and invalid values to highlight real-world stock market data quality issues.

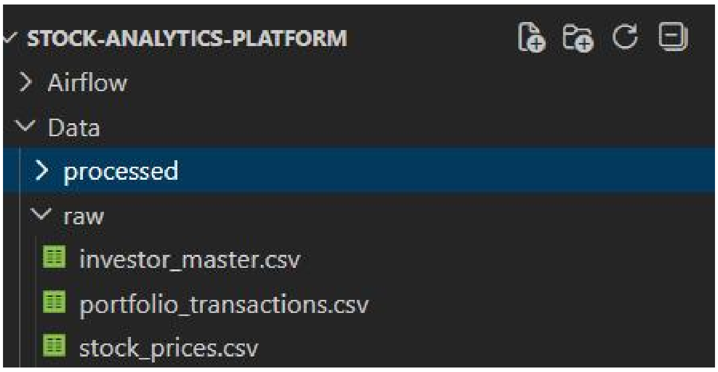
1. Input Datasets
The following raw datasets were used as inputs:

- Stock Prices Dataset : Daily stock price data with missing and invalid values
- Investor Master Dataset : Investor metadata with minor missing and invalid categorical values
- Portfolio Transactions Dataset : Trading transactions with missing fields and invalid quantities

Now we create a file in the scripts folder : clean_transform.py . In this file, we are going to write the code for cleaning the data and handling the missing values .



A sub-folder is created in the Data folder : the sub-folder **processed** stores the cleaned data files which is free from any kind of missing or invalid values.All raw datasets were preserved in an immutable raw data layer.



2. Data Cleaning Steps

2.1 Stock Prices Dataset Cleaning

The following cleaning rules were applied:

- Converted trade_date to a proper datetime format
- Sorted records by symbol and trade_date to enforce time-series consistency
- Removed records with missing close_price, as it is critical for return calculations
- Filled missing volume values with zero to represent no trading activity
- Removed records with negative or zero price values
- Enforced logical constraints where high_price ≥ low_price

2.2 Investor Master Dataset Cleaning

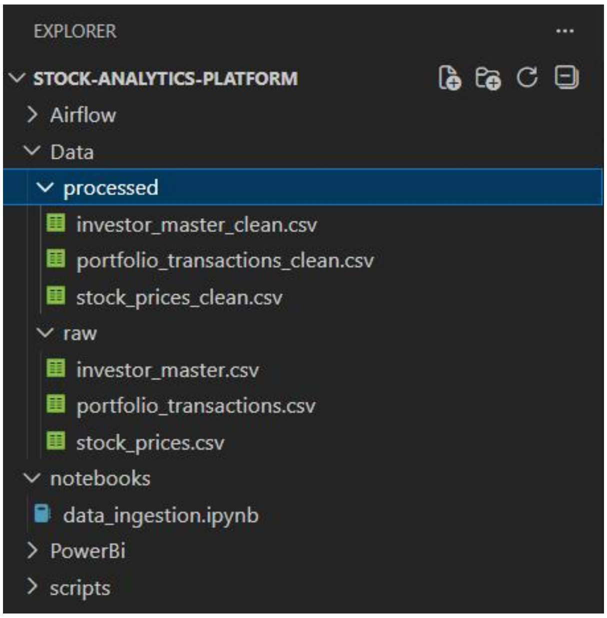
To clean and standardize investor metadata:

- Missing investor_type values were filled with a default category
- Missing region values were imputed with a common region
- Invalid risk_profile values were corrected to valid predefined categories

2.3 Portfolio Transactions Dataset Cleaning

For transaction-level data, the following validations were applied:

- Converted trade_date to datetime format
- Removed transactions with missing action or trade_price
- Removed transactions with negative or zero quantity values

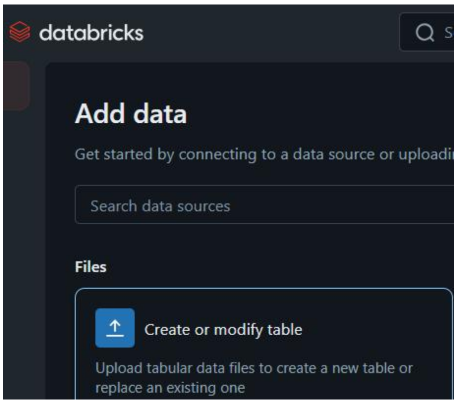


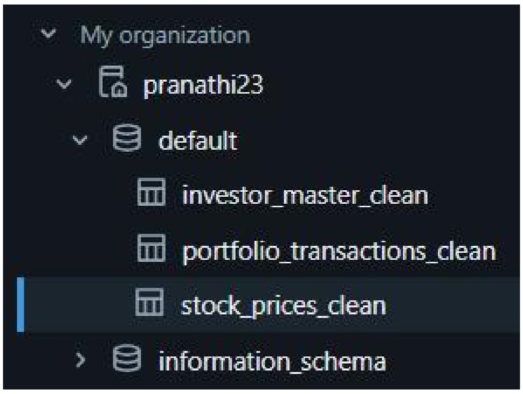
Gold Layer – Detailed Implementation & Calculations

In this project, the Gold Layer is used to transform cleaned transactional and market data into aggregated analytical tables that answer specific business questions related to:

1) Portfolio performance
2) Risk and volatility
3) Sector dominance
4) Investor trading behavior

All Gold tables are derived outputs and are designed to be directly consumed by Power BI for visualization. Initially, the data that is cleaned in the silver layer is added to databricks catalog.





Now, a notebook is created in databricks in order to perform some analysis - **gold_layer_analytics**
Step 1: The three tables are retrived and stored as spark dataframes

```
# retrieving the silver tables and storing them in the form of dataframes
stock_df=spark.table("stock_prices_clean")
investor_df=spark.table("investor_master_clean")
txn_df=spark.table("portfolio_transactions_clean")
```

Step 2: performing the analytics and storing them as tables. Gold Tables Implemented:

1) gold_portfolio_value:
This table tracks the total portfolio value over time, aggregated by trade date.
• Portfolio value is calculated by aggregating transaction-level values per date
• Each row represents portfolio value on a given trade date
• Formula used:

$$\boxed{\text{Portfolio Value (Date)} = \sum(\text{Quantity} \times \text{Trade Price})}$$

• Columns: trade_date, portfolio_value
• Enables portfolio trend analysis
• Used to derive Total Portfolio Value, Average Portfolio Value, Momentum Index in Power BI

4 days ago (1s)

```
%sql
SELECT * FROM gold_portfolio_value LIMIT 10;
```

See performance (1)

Table

	trade_date	1.2 portfolio_value
1	2022-01-03	57686.399999999994
2	2022-01-04	8623.86
3	2022-01-06	31358.7
4	2022-01-10	52921.95999999999
5	2022-01-11	134541.01
6	2022-01-13	7300.599999999999
7	2022-01-17	60824.4
8	2022-01-18	null
9	2022-01-20	60152.4
10	2022-01-21	53858.88

10 rows | 1.37s runtime

4 days ago (<1s)

```
portfolio_base_df = txn_df.join(
    stock_df.select("symbol", "trade_date", "close_price"),
    ["symbol", "trade_date"],
    "inner"
)
```

4 days ago (<1s)

```
portfolio_positions_df = portfolio_base_df.withColumn(
    "position_value",
    F.col("quantity") * F.col("close_price")
)
```

2)gold_stock_volatility:

• This table measures price risk for individual stocks (symbols).
• Volatility calculated per stock using standard deviation of returns
• Stocks classified into volatility ranges
Formula used:

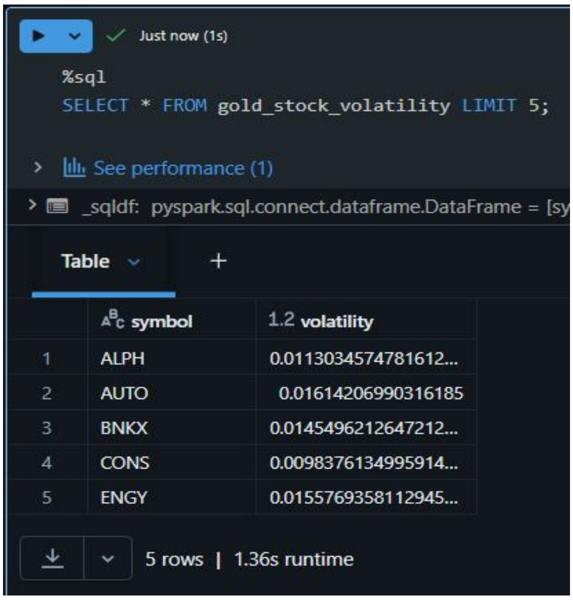
$$\text{Daily Return} = \frac{Close_{today} - Close_{previous}}{Close_{previous}}$$

$$\text{Volatility} = \text{STDDEV}(\text{Daily Returns})$$

Volatility Range Classification:

IF volatility < 0.012 → Low Volatility
IF volatility BETWEEN 0.012 AND 0.02 → Medium Volatility
IF volatility > 0.02 → High Volatility

• Columns: symbol, volatility, Volatility Range
• Quantifies stock-level risk
• Supports risk dashboards and heatmaps
• Enables quick identification of high-risk stocks


Just now (1s)

```
%sql
SELECT * FROM gold_stock_volatility LIMIT 5;
```

See performance (1)
_sqldf: pyspark.sql.connect.dataframe.DataFrame = [sy

Table

	symbol	volatility
1	ALPH	0.0113034574781612...
2	AUTO	0.016142069903161 85
3	BNKX	0.0145496212647212...
4	CONS	0.0098376134995914...
5	ENGY	0.0155769358112945...

5 rows | 1.36s runtime

3) gold_risk_adjusted_returns

• This table evaluates how efficiently a stock generates returns relative to its risk.
• Average daily return calculated per stock
• Combined with volatility to calculate risk-adjusted return
• Formula used:

$$\text{Risk Adjusted Return} = \frac{\text{Average Daily Return}}{\text{Volatility}}$$

$$\text{avg\_daily\_return} = \text{MEAN}(\text{Daily Returns})$$

• Columns: symbol, avg_daily_return, volatility,risk_adjusted_return
• dentifies high-return, low-risk stocks
• Supports risk vs return scatter analysis
• Used for performance ranking

Just now (1s) 25

```
%sql
select * from gold_risk_adjusted_returns LIMIT 5;
```

See performance (1)

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [symbol: string, avg_daily_return: double ... 2 more fiel

Table

	symbol	avg_daily_return	volatility	risk_adjusted_return
1	ALPH	0.0009011752172119902	0.0113034574781612...	0.07972562545160142
2	AUTO	-0.0001411095631312413	0.01614206990316185	-0.008741726679278057
3	BNKX	0.00011591826569761949	0.0145496212647212...	0.007967098496143571
4	CONS	0.00004505973587685943	0.0098376134995914...	0.004580352326174502
5	ENGY	-0.0005303693509360594	0.0155769358112945...	-0.034048374941077875

4) gold_sector_kpis:

- This table aggregates market activity at the sector level.
- Grouped stock data by sector, Aggregated trading volume and average prices
- Formulas used:

$$\boxed{\text{total\_volume} = \sum(\text{volume})}$$

$$\boxed{\text{avg\_close\_price} = \text{AVG}(\text{close\_price})}$$

- Columns:sector, total_volume, avg_close_price
- Helps compare sector strength, Identifies dominant sectors
- Enables treemaps, pie charts, and bar comparisons

3 minutes ago (1s)

```
%sql
select * from gold_sector_kpis LIMIT 5;
```

See performance (1)

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [sector: string, total_volun

Table

	sector	total_volume	avg_close_price
1	Telecom	258381647	297.6717113402059
2	Retail	248908944	162.84385714285705
3	Energy	250345650	309.9992900608519
4	Manufacturing	242555106	417.95985743380834
5	Technology	254453136	403.14660569105695

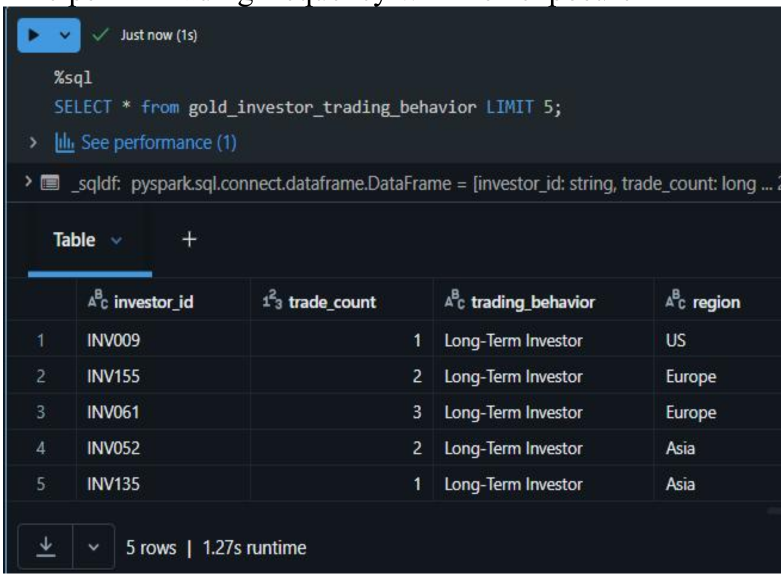
5) gold_investor_trading_behavior

• This table classifies investors based on how frequently they trade.
• Count total number of trades per investor
• Categorize investor behavior using thresholds
Formula used:

$$\boxed{\text{trade\_count} = \text{COUNT}(\text{transactions per investor})}$$

Trading Behavior Classification
IF trade_count <= 3 → Long-Term Investor
IF trade_count BETWEEN 4 AND 6 → Swing Trader
IF trade_count > 6 → Frequent Trader

- Columns: investor_id,trade_count,trading_behavior, region
- Identifies investor risk appetite
- Supports behavioral segmentation dashboards
- Helps link trading frequency with risk exposure



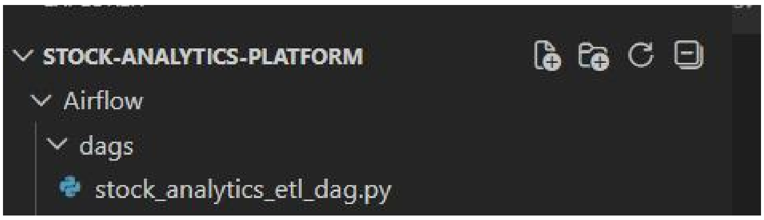
Workflow Automation:

This project implements a scheduled, incremental Bronze–Silver–Gold ETL pipeline using Apache Airflow for orchestration and Databricks for analytics processing.

The pipeline is designed to:
Automatically detect new data
Process data in layered architecture
Trigger analytical workloads in Databricks
Maintain observability through logs, retries, and monitoring

Workflow Orchestration-Apache Airflow:

Initially , we create a folder named Airflow in our project folder where we write our Dag



The pipeline is defined as an Airflow DAG:

```
dag_id="stock_analytics_incremental_etl_pipeline_v2"
schedule_interval="@daily"
```

catchup=False

• The DAG runs once every day
• No backfilling of old dates
• Can also be triggered manually

DAG Task Flow:

```
# ETL FLOW (UNCHANGED)
check_data >> bronze_ingestion >> silver_cleaning >> gold_analytics >> update_run_time
```

Incremental Processing Logic
• Task: check_incremental_data
• Reads raw CSV files andCompares their latest modification time with the last successful run
• Stores the timestamp using Airflow Variables

```
Variable.set("last_successful_run", str(datetime.now().timestamp()))
```

• Prevents unnecessary reprocessing
• Enables incremental ETL
• Makes the pipeline efficient and scalable

Bronze Layer – Raw Data Ingestion
• Executed using BashOperator
• Runs a Python ingestion script: python /opt/airflow/project/Data/Bronze/bronze_layer.py
• Reads raw CSV files and Copies them into the Bronze directory - Maintains raw data integrity
Logging:
Application-level logs written to: stock-analytics-platform/logs/bronze.log

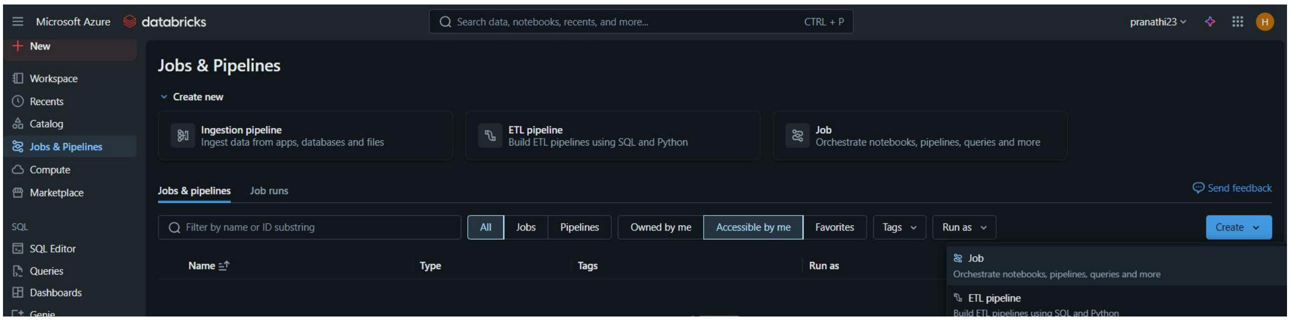
```
logs > bronze.log
2026-01-03 09:14:47,708 | INFO | Bronze layer ingestion started
2026-01-03 09:14:47,734 | INFO | Copied file: investor_master.csv
2026-01-03 09:14:47,754 | INFO | Copied file: portfolio_transactions.csv
2026-01-03 09:14:47,794 | INFO | Copied file: stock_prices.csv
2026-01-03 09:14:47,794 | INFO | Bronze layer completed successfully. Files copied: 3
```

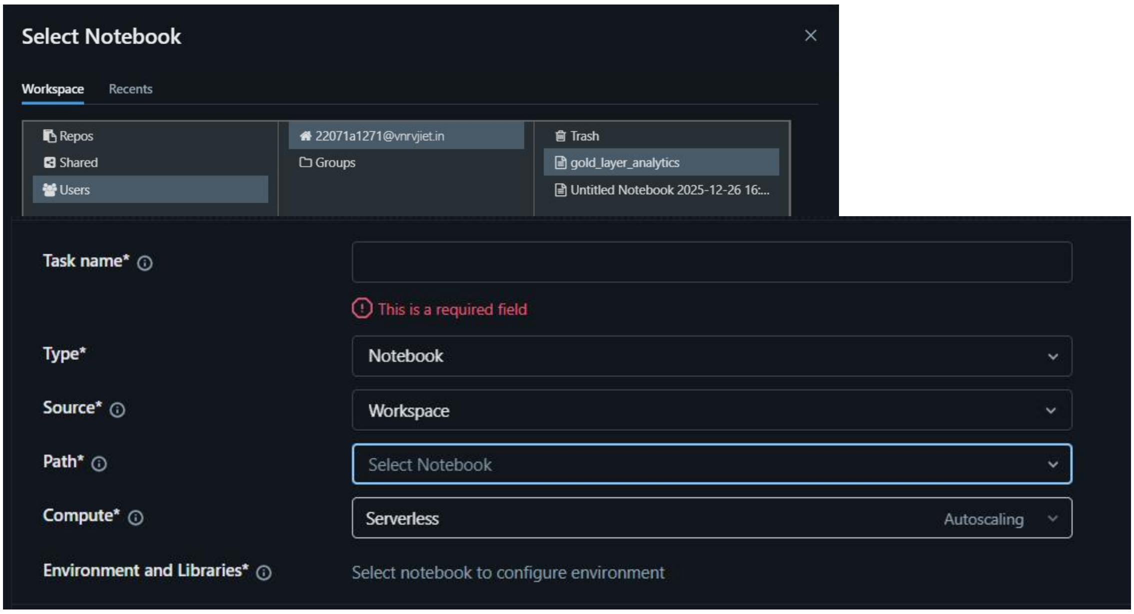
Silver Layer – Cleaning & Feature Engineering
• Runs clean_transform.py and Includes Null handling, Duplicate removal
• Feature engineering:Risk category, Daily stock returns
Logging:
Logs written to: stock-analytics-platform/logs/bronze.log

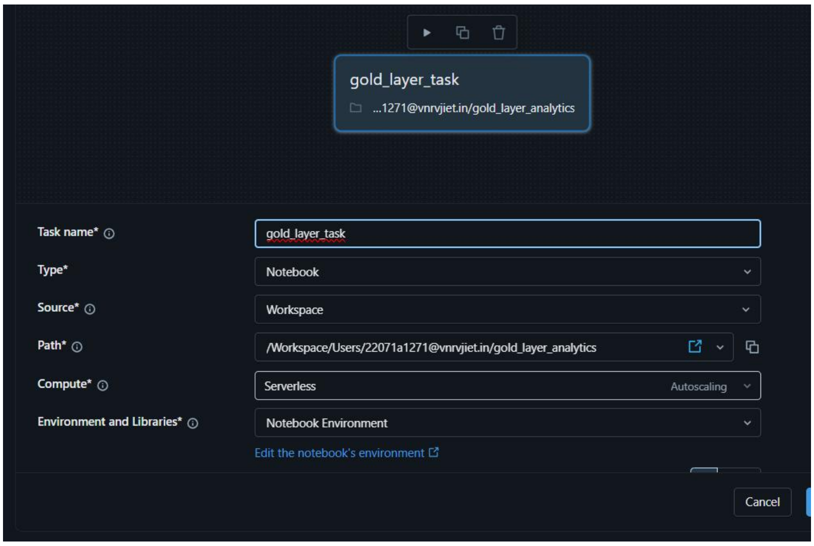
Error Handling
• Uses Python try/except
• Errors are logged with stack traces
• Task fails cleanly, triggering retries

```
2026-01-03 06:12:38,652 - INFO - Silver layer transformation started
2026-01-03 06:12:38,752 - INFO - Investor master cleaned successfully
2026-01-03 06:12:38,964 - INFO - Portfolio transactions cleaned successfully
2026-01-03 06:12:39,119 - INFO - Stock prices cleaned successfully
2026-01-03 06:12:39,119 - INFO - Silver layer completed successfully
```

Gold Layer – Databricks Analytics:
Initially a job is created in databricks where we select an existing notebook and provide the necessary details like task_name and the path

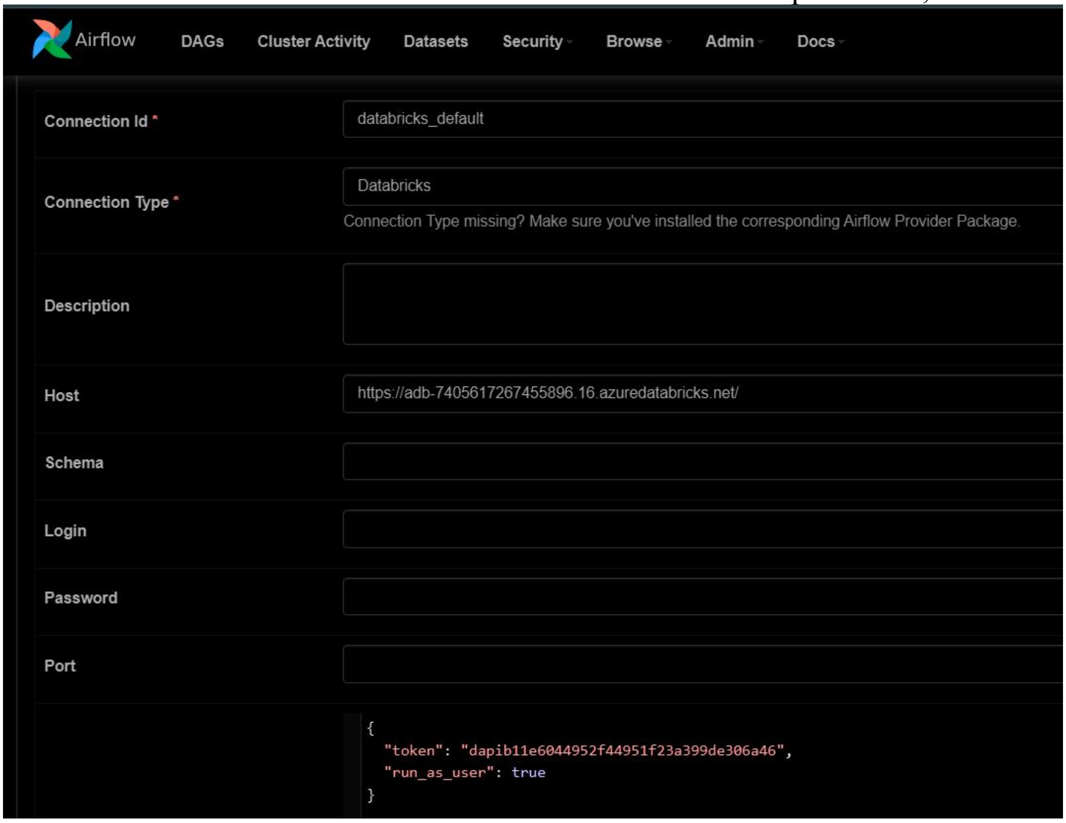




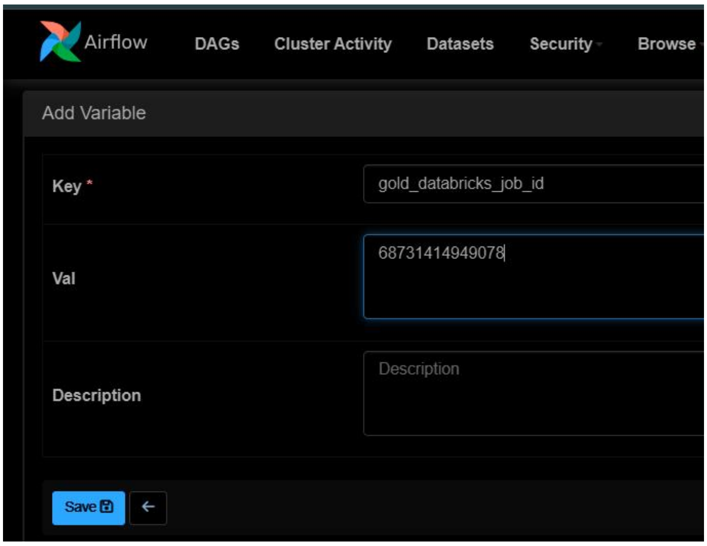


Airflow Connection to Databricks:

- Databricks Connection created in Airflow UI: Stores workspace URL, authentication token



- Databricks Job ID stored as an Airflow Variable:



Gold Layer Execution:

```python
# Task 4: Gold Layer (Databricks)
gold_analytics = DatabricksRunNowOperator(
    task_id="gold_layer_analytics",
    databricks_conn_id="databricks_default",
    job_id=68731414949078,
    on_execute_callback=gold_on_execute,
    on_success_callback=gold_on_success,
    on_failure_callback=gold_on_failure
)
```

- Airflow triggers Databricks REST API
- Databricks executes the configured job
- Job runs Spark-based analytics on Silver data
- Execution status is tracked by Airflow

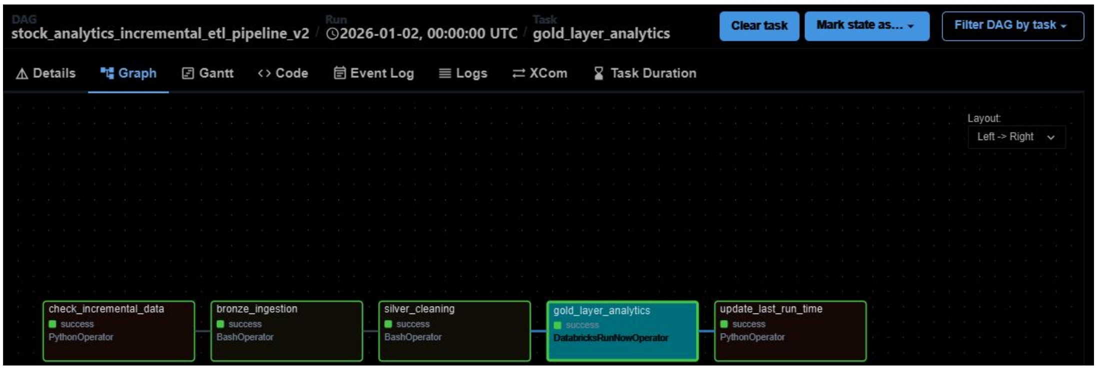
Logging:

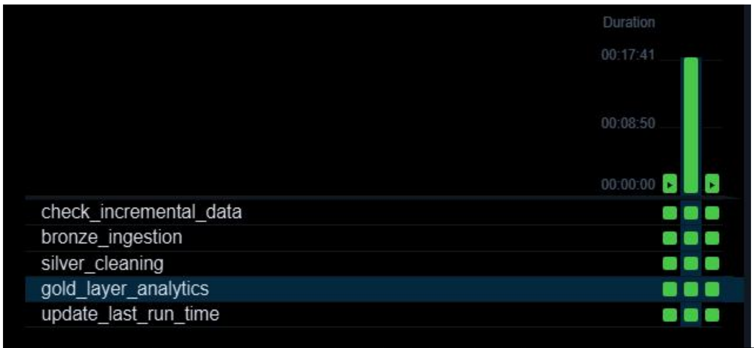
```
[2026-01-03, 06:13:44 UTC] {databricks.py:77} INFO - gold_layer_analytics completed successfully.
[2026-01-03, 06:13:44 UTC] {databricks.py:78} INFO - View run status, Spark UI, and logs at https://adb-7405617267455896.16.azuredatabricks.net/?o=7405617267455896#job/68731414949078/
[2026-01-03, 06:13:44 UTC] {taskinstance.py:340} ▶ Post task execution logs
```

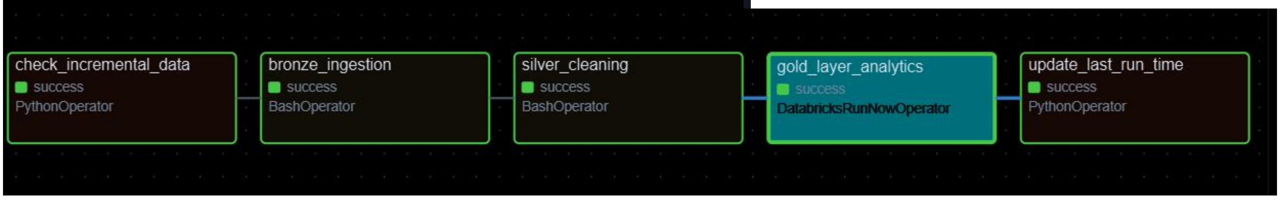
Monitoring & Observability

Airflow UI Monitoring:

- DAG Graph View
- Gantt Chart
- Task Duration
- Success / Failure states







Task-Level Logs:

- Each task produces Airflow logs automatically, Accessible via Airflow Web UI
- Application Logs: Bronze: bronze.log, Silver: silver_layer.log

Retries & Fault Tolerance

- Configured via default_args:

```
default_args = {
    "owner": "airflow",
    "depends_on_past": False,
    "email": [redacted]
    "email_on_failure": True,
    "email_on_retry": False,
    "retries": 2,
    "retry_delay": timedelta(minutes=5)
}
```

• Temporary failures don't break the pipeline, Automatic retry without manual intervention
• Production-grade fault tolerance

Trigger the pipeline:
• **Stop Airflow:** Stops all running Airflow containers and services using docker-compose down.
• **Start Airflow:** Starts all Airflow services in the background using docker-compose up -d and makes the UI available.
• **Verify Airflow:** Verifies that all Airflow containers are running and healthy using docker ps.
• **Trigger DAG:** Manually triggers the ETL pipeline from the Airflow UI or automatically via the configured schedule.

```
C:\Users\mpran\airflow>docker-compose down
[+] Running 8/8
 ✔Container airflow-airflow-webserver-1  Removed                                                    5.4s
 ✔Container airflow-airflow-scheduler-1  Removed                                                    5.4s
 ✔Container airflow-airflow-worker-1     Removed                                                    5.4s
 ✔Container airflow-airflow-triggerer-1  Removed                                                    3.8s
 ✔Container airflow-airflow-init-1       Removed                                                    0.6s
 ✔Container airflow-postgres-1           Removed                                                    1.0s
 ✔Container airflow-redis-1              Removed                                                    0.9s
 ✔Network airflow_default                Removed                                                    0.4s

C:\Users\mpran\airflow>docker-compose up -d
[+] Running 8/8
 ✔Network airflow_default                Created                                                    0.1s
 ✔Container airflow-redis-1              Healthy                                                    8.8s
 ✔Container airflow-postgres-1           Healthy                                                    8.8s
 ✔Container airflow-airflow-init-1       Exited                                                    44.2s
 ✔Container airflow-airflow-scheduler-1  Started                                                   44.5s
 ✔Container airflow-airflow-worker-1     Started                                                   44.5s
 ✔Container airflow-airflow-triggerer-1  Started                                                   44.7s
 ✔Container airflow-airflow-webserver-1  Started                                                   44.7s
```

Automated Failure Notifications (Email Alerts)

The pipeline includes **proactive failure notifications** using Apache Airflow's native email alerting mechanism.Whenever a critical task (Gold layer analytics) fails, Airflow automatically sends an email notification to the configured recipients. This ensures quick awareness of pipeline issues without requiring manual monitoring.

Implementation Details

- Email alerts are enabled using Airflow's built-in configuration:
  - email_on_failure = True
  - email_on_retry = False
- Alerts are triggered **only after task failure**, avoiding unnecessary notifications
- SMTP configuration is managed at the Docker environment level

Benefits

- Immediate visibility into pipeline failures
- Reduced operational risk
- Industry-standard alerting behavior
- No custom email logic inside DAGs

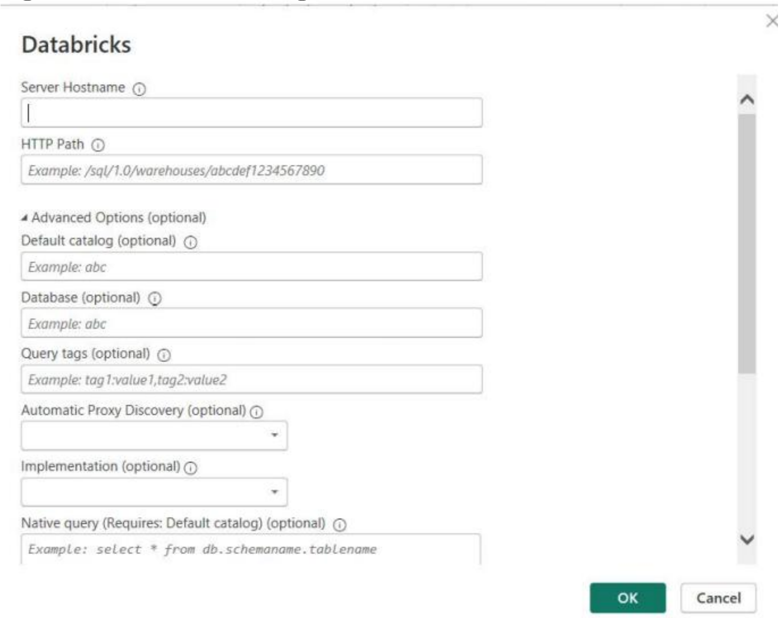
Connection To PowerBI:

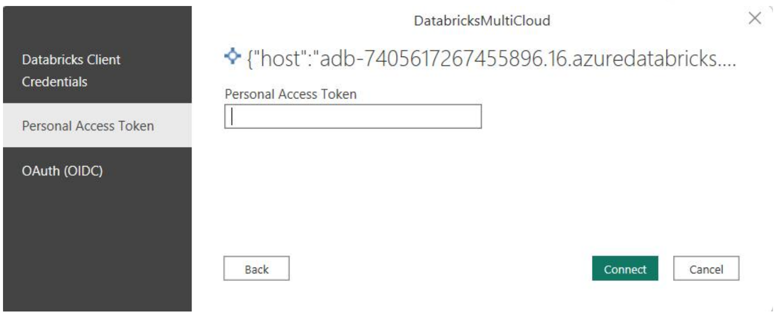
STEP 1 :Get Databricks Connection Details

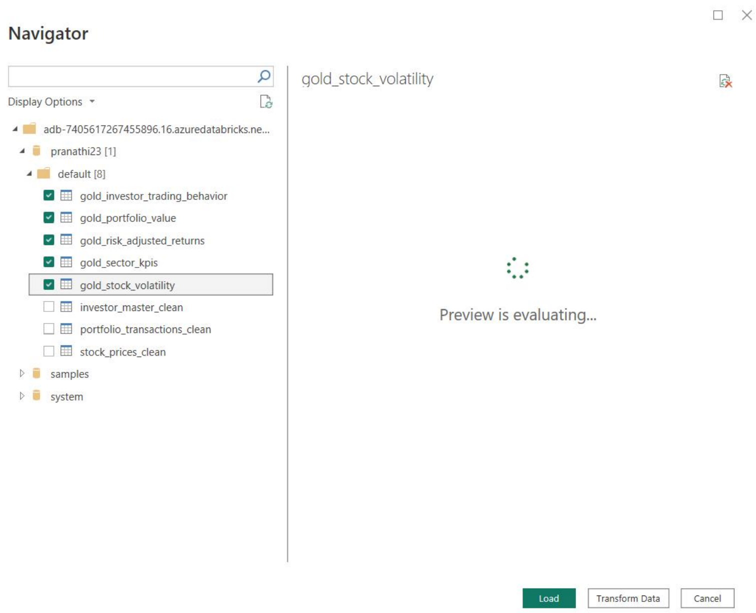
From Databricks: Go to Compute → Your Cluster
Copy: Server hostname, HTTP Path
Go to User Settings → Access Tokens
Create a Personal Access Token

STEP 2: Open Power BI Desktop

Open Power BI Desktop ->Click Get Data ->Search: Azure Databricks ->Click Connect



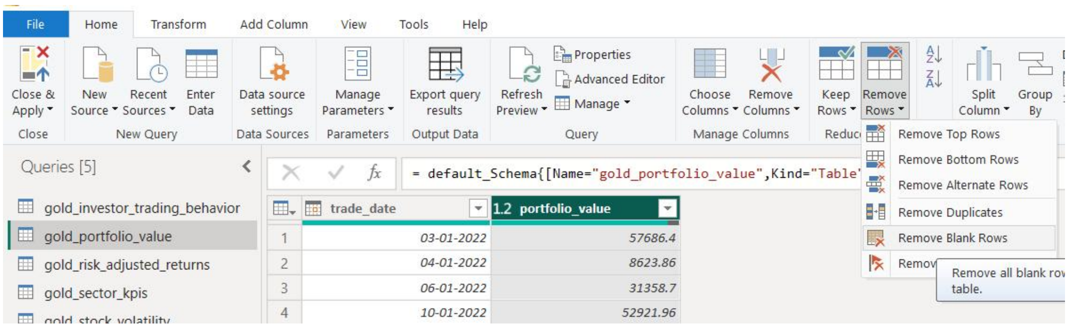




Transforming the Data:

The first step before visualization is to ensure that the data is clean and the attributes follow the right datatype . In this step, conditional columns are created and the redundant columns are removed in order to enhance our visualizations.

1.Remove Nulls & Duplicates

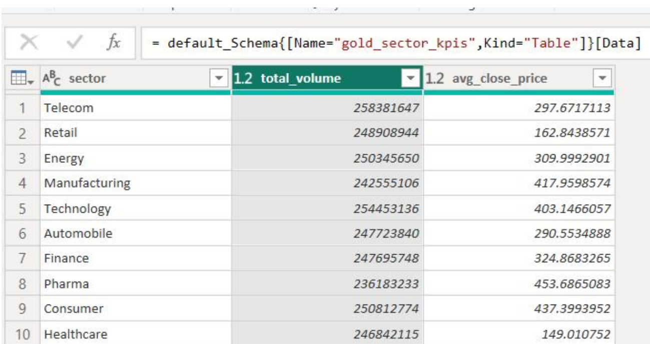


Null portfolio values were removed to avoid incorrect portfolio trend analysis.

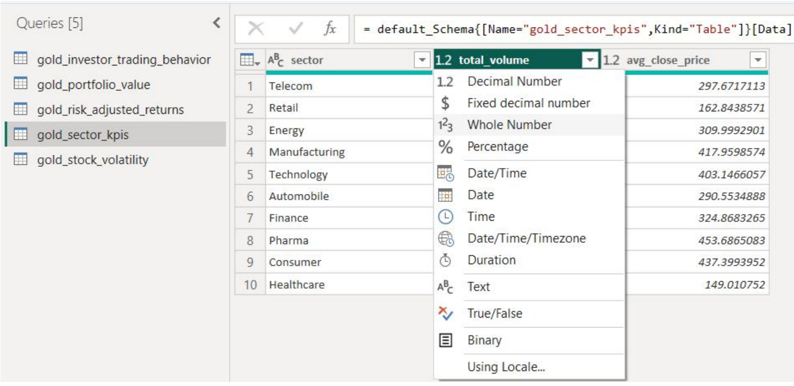
- Select portfolio_value
- Home → Remove Rows → Remove Blank Rows

2.Changing The Datatypes:

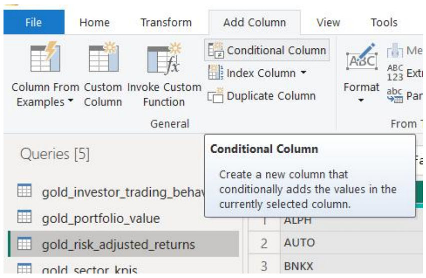
Before performing vizualizations it is very essential to check the attributes are of the right datatypes. Initially, the total_volume in the gold_sectors_kpi was of the decimal datatype . After changing the datatype it is of the whole Number datatype.



	sector	total_volume	avg_close_price
1	Telecom	258381647	297.6717113
2	Retail	248908944	162.8438571
3	Energy	250345650	309.9992901
4	Manufacturing	242555106	417.9598574
5	Technology	254453136	403.1466057
6	Automobile	247723840	290.5534888
7	Finance	247695748	324.8683265
8	Pharma	236183233	453.6865083
9	Consumer	250812774	437.3993952
10	Healthcare	246842115	149.010752

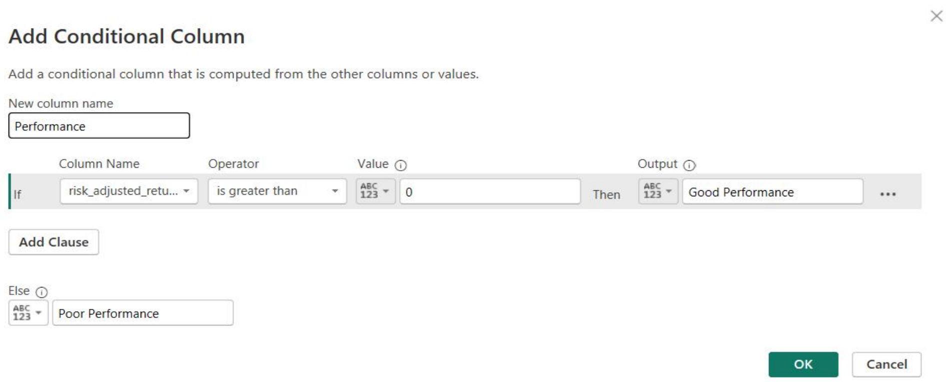


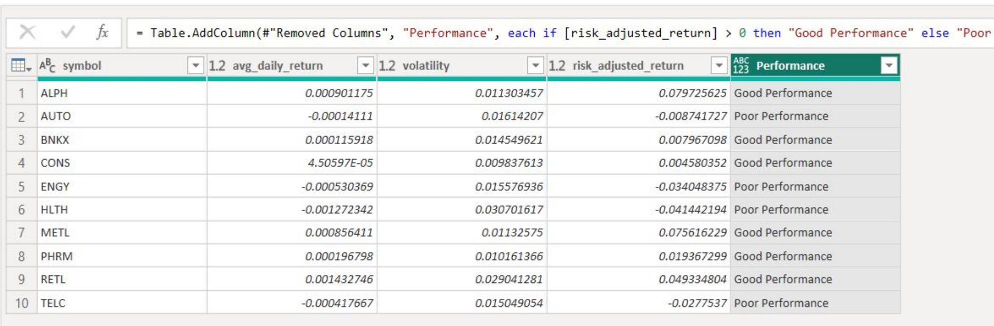
3. Adding Conditional Columns:



A. Performance Category (Risk-Adjusted Returns)

In gold_risk_adjusted_returns:
If risk_adjusted_return > 0 → "Good Performance"
Else → "Poor Performance"
Non-technical users understand it immediately

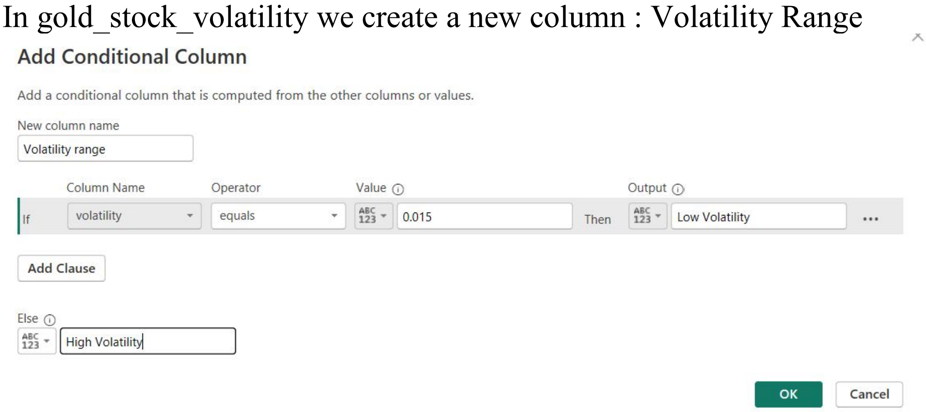




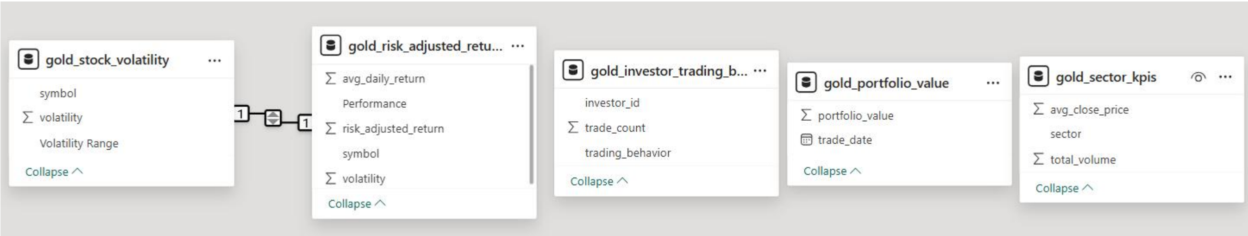
	symbol	avg_daily_return	volatility	risk_adjusted_return	Performance
1	ALPH	0.000901175	0.011303457	0.079725625	Good Performance
2	AUTO	-0.00014111	0.01614207	-0.008741727	Poor Performance
3	BNKX	0.000115918	0.014549621	0.007967098	Good Performance
4	CONS	4.50597E-05	0.009837613	0.004580352	Good Performance
5	ENGY	-0.000530369	0.015576936	-0.034048375	Poor Performance
6	HLTH	-0.001272342	0.030701617	-0.041442194	Poor Performance
7	METL	0.000856411	0.01132575	0.075616229	Good Performance
8	PHRM	0.000196798	0.010161366	0.019367299	Good Performance
9	RETL	0.001432746	0.029041281	0.049334804	Good Performance
10	TELC	-0.000417667	0.015049054	-0.0277537	Poor Performance

B.Volatility Range

In gold_stock_volatility we create a new column : Volatility Range



Tables and Schema:



DAX MEASURES

```
Total Portfolio Value =
SUM(gold_portfolio_value[portfolio_value])
```

```
Risk Efficiency Score =
DIVIDE(
    AVERAGE(gold_risk_adjusted_returns[risk_adjusted_return]),
    AVERAGE(gold_stock_volatility[volatility])
)
```

```
Portfolio Momentum Index =
VAR LatestValue =
    CALCULATE(
        MAX(gold_portfolio_value[portfolio_value]),
        LASTDATE(gold_portfolio_value[trade_date])
    )
VAR AvgValue =
    AVERAGE(gold_portfolio_value[portfolio_value])
RETURN
DIVIDE(LatestValue - AvgValue, AvgValue)
```

```
Sector Dominance Index =
DIVIDE(
    SUM(gold_sector_kpis[total_volume]),
    CALCULATE(SUM(gold_sector_kpis[total_volume]), ALL(gold_sector_kpis))
)
```

```
Investor Trading Intensity =
DIVIDE(
    SUM(gold_investor_trading_behavior[trade_count]),
    DISTINCTCOUNT(gold_investor_trading_behavior[investor_id])
)
```

```
Risk Adjusted Performance Index =
DIVIDE(
    AVERAGE(gold_risk_adjusted_returns[avg_daily_return]),
    AVERAGE(gold_risk_adjusted_returns[volatility])
)
```

```
Sector Risk Contribution % =
DIVIDE(
    SUM(gold_sector_kpis[total_volume]),
    CALCULATE(SUM(gold_sector_kpis[total_volume]), ALL(gold_sector_kpis))
)
```

```
Portfolio Volatility Exposure =
AVERAGE(gold_stock_volatility[volatility]) *
SUM(gold_portfolio_value[portfolio_value])
```

Dashboard 1: Stock Price Trend Dashboard

This dashboard provides a market-level view of stock price behavior across symbols and sectors. It helps analysts understand price trends, sector dominance, trading volume concentration, and volatility patterns in the market.

Chart Type	Chart Title	Attributes Used	Interpretation
Card	Total Trading Volume	**Value:** SUM(total_volume)	Displays the overall trading activity across all stocks, giving a quick sense of market participation and liquidity.
Card	Average Closing Price	**Value:** AVERAGE(avg_close_price)	Represents the average stock price across all symbols, providing a high-level view of market price levels.
Card	Number of Active Stocks	**Value:** DISTINCTCOUNT(symbol)	Indicates how many unique stocks are actively involved in trading, reflecting market breadth.
Slicer	Symbol	**Field:** symbol	Allows users to filter the dashboard for specific stocks, enabling focused analysis at the stock level.
Slicer	Sector	**Field:** sector	Enables sector-wise filtering to compare stock behavior across different industries.
Treemap	Average Stock Price by Sector	**Group:** sector **Values:** AVERAGE(avg_close_price)	Visually compares average stock prices across sectors, highlighting sectors with relatively higher or lower price levels.
Clustered Column Chart	Average Closing Price by Sector	**X-axis:** sector **Y-axis:** AVERAGE(avg_close_price)	Clearly ranks sectors based on average closing price, making it easy to identify top- and bottom-performing sectors by price.
Table	Stock-wise Volatility Classification	**Columns:** symbol, SUM(volatility), Volatility Range	Provides a detailed, stock-level breakdown of volatility, categorizing stocks into low or high volatility for risk assessment.
Pie Chart	Sector-wise Trading Volume Distribution	**Legend:** sector **Values:** SUM(total_volume)	Shows how total trading volume is distributed across sectors, revealing which sectors dominate market activity.
Scatter Plot	Price vs Volatility Relationship by Sector	**X-axis:** avg_close_price **Y-axis:** volatility **Legend:** sector	Analyzes the relationship between price levels and volatility, helping identify sectors that combine high prices with higher or lower risk.

Dashboard 2: Portfolio Performance & Return Analysis

This dashboard explains how the portfolio is performing, whether it is growing or declining, and what factors are influencing its value. It majorly focuses on on returns, momentum, and value changes.

Chart Type	Chart Title	Attributes Used	Interpretation
Line Chart	Average Portfolio Value by Year and Month	**X-axis:** Year, Month **Y-axis:** AVERAGE(portfolio_value)	Shows how the portfolio's average value changes over time, helping identify growth phases, volatility periods, and overall performance trends.
Gauge Chart	Portfolio Momentum Trend Over Time	**Value:** Portfolio Momentum Index **Min–Max:** Calculated momentum range	Indicates the direction and strength of portfolio momentum. A negative value reflects weakening momentum and potential underperformance.
Pie Chart	Portfolio Performance Distribution	**Legend:** Performance Category (Good / Poor) **Values:** COUNT(stocks)	Displays the proportion of stocks with good versus poor performance, providing a quick health check of the overall portfolio.
Card	Total Portfolio Value	**Value:** SUM(portfolio_value)	Represents the cumulative value of all holdings, indicating the total size of the portfolio.
Card	Average Portfolio Value	**Value:** AVERAGE(portfolio_value)	Shows the mean portfolio value, useful for understanding typical investment exposure over time.
Card	Risk Efficiency Score	**Value:** Risk Efficiency Score (DAX Measure)	Reflects how efficiently the portfolio converts risk into returns, combining volatility and performance into a single indicator.

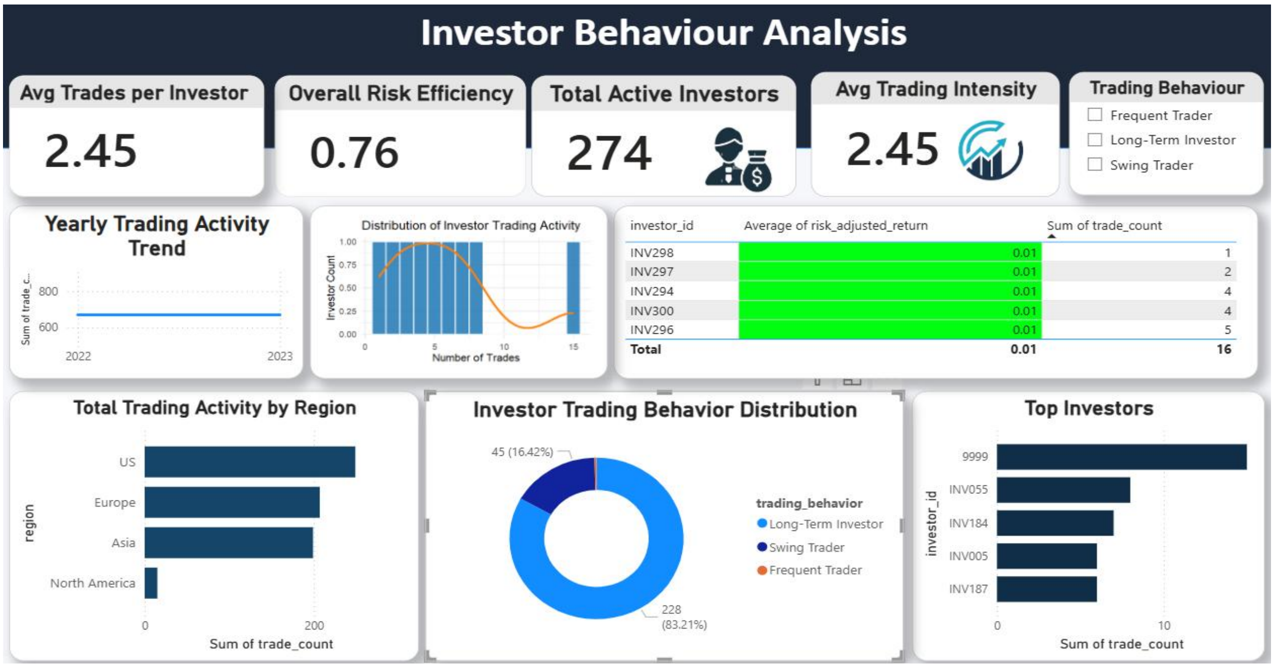
Chart Type	Chart Title	Attributes Used	Interpretation
Column Chart	Average Daily Return by Stock	**X-axis:** symbol **Y-axis:** AVERAGE(daily_return)	Compares daily returns across stocks, identifying top performers and consistently underperforming stocks.
Card	Portfolio Momentum Index	**Value:** Portfolio Momentum Index (DAX Measure)	Quantifies the overall trend strength of portfolio returns; negative values indicate declining momentum.
Card	Sum of Risk-Adjusted Return	**Value:** SUM(risk_adjusted_return)	Aggregates risk-adjusted returns to assess total performance after accounting for volatility.
Area Chart	Drivers of Portfolio Value Change by Year	**X-axis:** Year **Y-axis:** SUM(portfolio_value)	Highlights how portfolio value changes year-over-year, emphasizing periods that contributed most to growth or decline.

Dashboard 3: Investor Behaviour Analysis

This dashboard analyzes how investors trade, focusing on activity intensity, behavior types, and top contributors.

Chart Type	Chart Title	Attributes Used	Interpretation
Card	Avg Trades per Investor	**Value:** AVERAGE(trade_count)	Shows the average number of trades executed by each investor, indicating overall investor activity levels.
Card	Overall Risk Efficiency	**Value:** AVERAGE(risk_efficiency_sco	Represents how efficiently investors are managing risk across the

Chart Type	Chart Title	Attributes Used	Interpretation
		re)	portfolio, combining return and risk factors.
Card	Total Active Investors	**Value:** DISTINCTCOUNT(investor_id)	Displays the total number of unique investors actively participating in trading activities.
Card	Avg Trading Intensity	**Value:** AVERAGE(trading_intensity)	Reflects how frequently investors trade on average, helping assess aggressiveness or passiveness in trading behavior.
Slicer	Trading Behaviour	**Field:** trading_behavior (Frequent Trader, Long-Term Investor, Swing Trader)	Allows filtering the dashboard by investor type to analyze behavior patterns for specific trading styles.
Line Chart	Yearly Trading Activity Trend	**X-axis:** year **Y-axis:** SUM(trade_count)	Shows how overall trading activity changes over time, helping identify growth, decline, or stability in investor participation.
R Visual (Histogram + Density)	Distribution of Investor Trading Activity	**Values:** trade_count	Displays the distribution of trades across investors, highlighting common trading frequencies and identifying outliers.
Table	Investor Risk & Trading Summary	**Columns:** investor_id, AVERAGE(risk_adjusted_return), SUM(trade_count)	Provides a detailed investor-level view combining risk-adjusted performance with trading volume.
Bar Chart	Total Trading Activity by Region	**Y-axis:** region **X-axis:** SUM(trade_count)	Compares trading activity across geographic regions, revealing where investor participation is highest.
Donut Chart	Investor Trading Behavior Distribution	**Legend:** trading_behavior **Values:** COUNT(investor_id)	Shows the proportion of investors by trading style, giving insight into dominant investment behavior patterns.
Horizontal Bar Chart	Top Investors	**Y-axis:** investor_id **X-axis:** SUM(trade_count)	Highlights the most active investors based on total trades, useful for identifying high-engagement participants.



Dashboard 4: Risk & Volatility Indicators

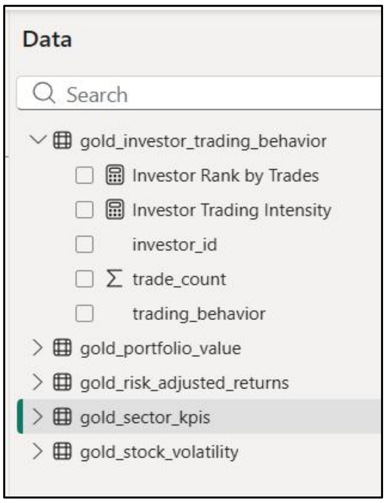
This dashboard provides a comprehensive view of market and portfolio risk, showing how volatility and returns interact across stocks and sectors. It supports risk monitoring, portfolio optimization, and informed investment decisions.

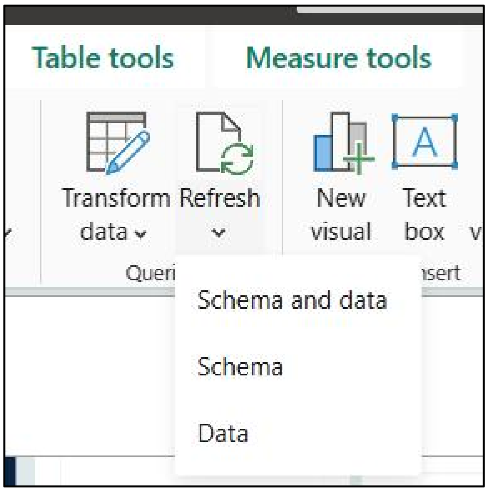
Chart Type	Chart Title	Attributes Used	Interpretation
Card	Risk-Adjusted Performance Index	**Value:** Risk-Adjusted Performance Index (DAX)	Represents overall portfolio performance after adjusting for risk. A low value indicates modest returns relative to volatility.
Column Chart	Stock Distribution by Volatility Category	**X-axis:** Volatility Range (High / Low) **Y-axis:** COUNT(symbol)	Shows how many stocks fall into high vs low volatility categories, helping assess overall portfolio risk composition.
Card	Risk Efficiency Score	**Value:** Risk Efficiency Score (DAX)	Measures how efficiently the portfolio converts risk into returns. Higher values indicate better risk utilization.
Pie Chart	Average Volatility Contribution by Sector	**Legend:** sector **Values:** AVERAGE(volatility)	Highlights sector-wise contribution to portfolio volatility, identifying sectors that introduce higher risk.
Gauge Chart	Overall Portfolio Volatility Level	**Value:** AVERAGE(volatility) **Min–Max:** Volatility range	Provides a single consolidated indicator of overall portfolio volatility relative to expected bounds.
Table	Stock-level Risk & Return Summary	**Columns:** symbol, volatility, risk_adjusted_return, Risk Efficiency Score	Enables detailed stock-level comparison of volatility, returns, and risk efficiency for informed investment decisions.
Scatter Plot	Risk vs	**X-axis:** volatility **Y-axis:**	Visualizes the risk–return tradeoff for

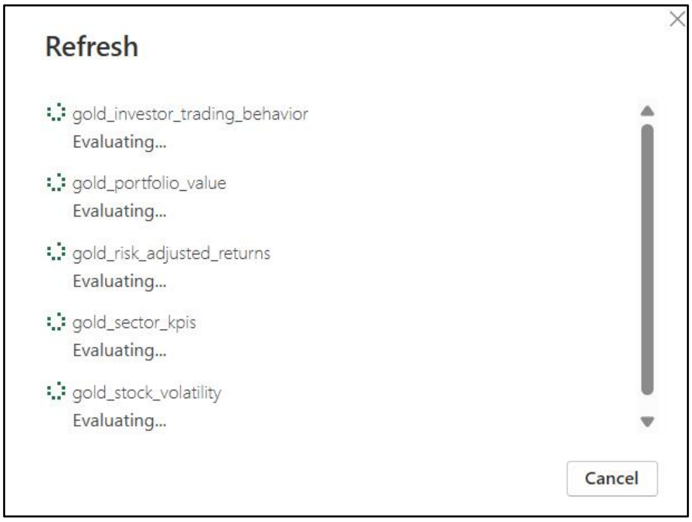
Chart Type	Chart Title	Attributes Used	Interpretation
	Return Distribution by Stock	risk_adjusted_return **Legend:** symbol	individual stocks, identifying efficient and inefficient risk profiles.
Column Chart	Risk Efficiency Score by Stock	**X-axis:** symbol **Y-axis:** Risk Efficiency Score	Compares stocks based on risk efficiency, highlighting top-performing stocks and those with negative risk-adjusted outcomes.
Slicers	Sector, Symbol, Volatility Range	**Filters:** sector, symbol, Volatility Range	Allow interactive exploration of risk and volatility patterns across specific sectors, stocks, or volatility levels.

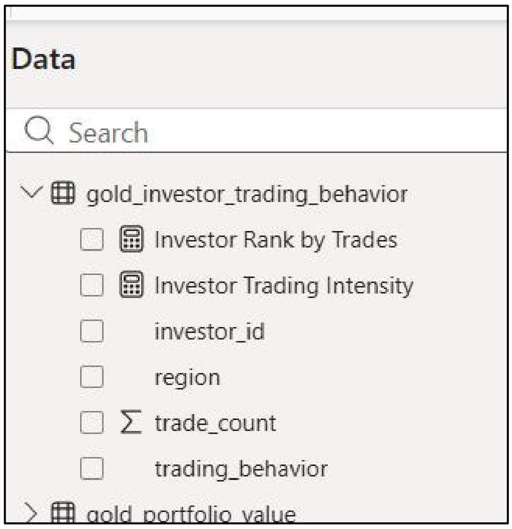
Refresh Behaviour:

In order to test the refresh behaviour in PowerBI , the region column is added in the gold layer-gold_investor_trading_behaviour table. The data and schema is refreshed and now a new attribute for our visualization is enabled- region







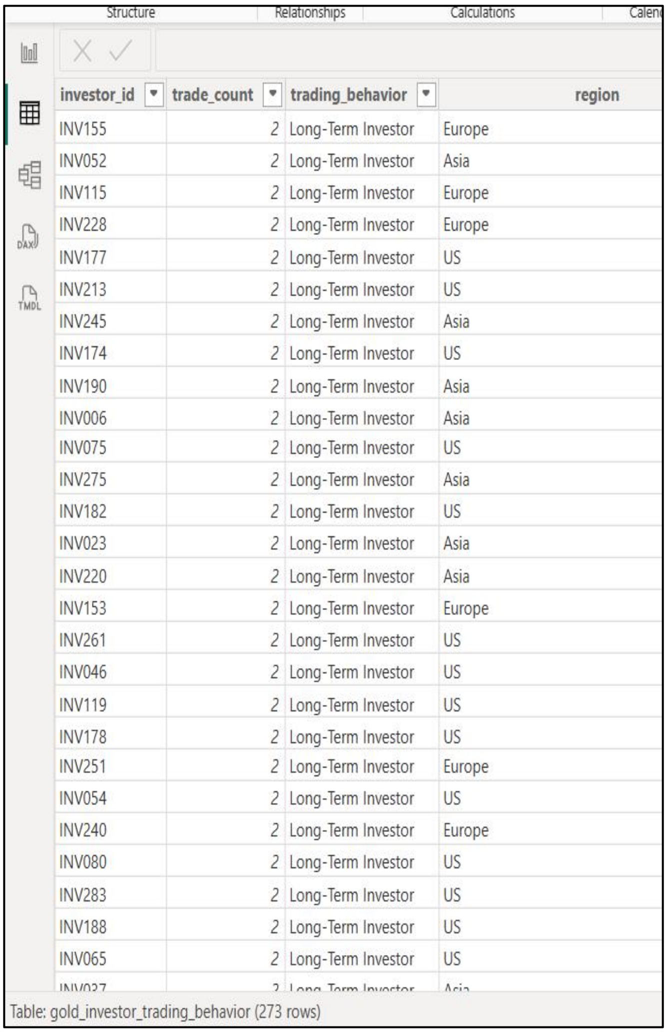


Data Refresh :
Here we are adding a new record in our gold_investor_trading_behaviour table. Before the refresh there were 273 rows and after data refresh there are 274 records.

Inserting a record to test manual refresh behaviour

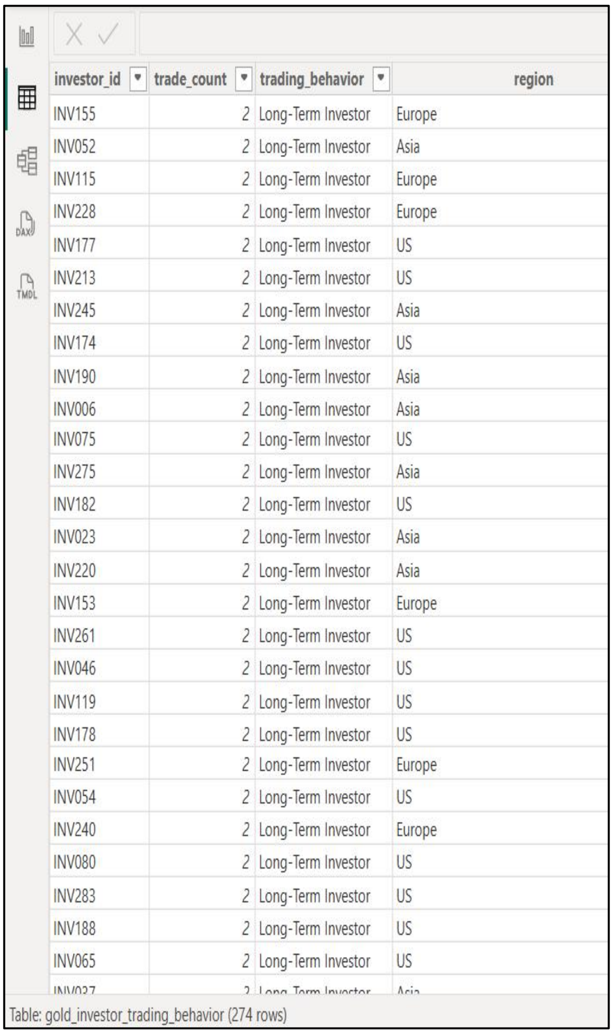
✓ 1.29s 41

```
%sql
INSERT INTO gold_investor_trading_behavior
VALUES (
    9999,
    15,
    'Frequent Trader',
    'North America'
);
```



investor_id	trade_count	trading_behavior	region
INV155	2	Long-Term Investor	Europe
INV052	2	Long-Term Investor	Asia
INV115	2	Long-Term Investor	Europe
INV228	2	Long-Term Investor	Europe
INV177	2	Long-Term Investor	US
INV213	2	Long-Term Investor	US
INV245	2	Long-Term Investor	Asia
INV174	2	Long-Term Investor	US
INV190	2	Long-Term Investor	Asia
INV006	2	Long-Term Investor	Asia
INV075	2	Long-Term Investor	US
INV275	2	Long-Term Investor	Asia
INV182	2	Long-Term Investor	US
INV023	2	Long-Term Investor	Asia
INV220	2	Long-Term Investor	Asia
INV153	2	Long-Term Investor	Europe
INV261	2	Long-Term Investor	US
INV046	2	Long-Term Investor	US
INV119	2	Long-Term Investor	US
INV178	2	Long-Term Investor	US
INV251	2	Long-Term Investor	Europe
INV054	2	Long-Term Investor	US
INV240	2	Long-Term Investor	Europe
INV080	2	Long-Term Investor	US
INV283	2	Long-Term Investor	US
INV188	2	Long-Term Investor	US
INV065	2	Long-Term Investor	US
INV027	2	Long-Term Investor	Asia

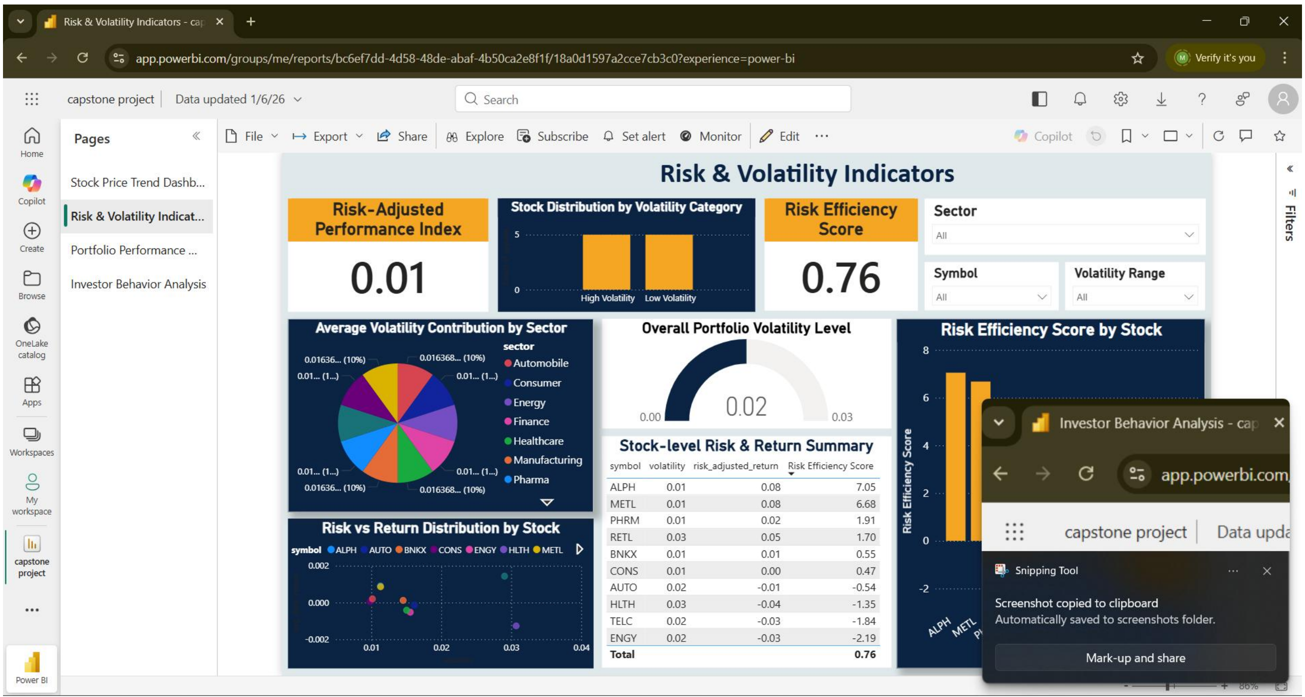
Table: gold_investor_trading_behavior (273 rows)

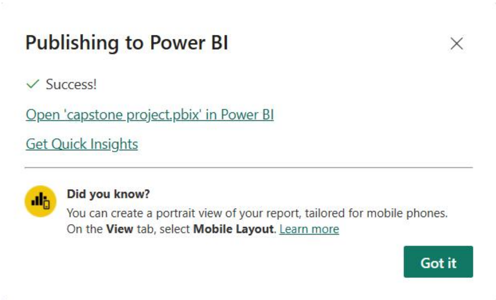


investor_id	trade_count	trading_behavior	region
INV155	2	Long-Term Investor	Europe
INV052	2	Long-Term Investor	Asia
INV115	2	Long-Term Investor	Europe
INV228	2	Long-Term Investor	Europe
INV177	2	Long-Term Investor	US
INV213	2	Long-Term Investor	US
INV245	2	Long-Term Investor	Asia
INV174	2	Long-Term Investor	US
INV190	2	Long-Term Investor	Asia
INV006	2	Long-Term Investor	Asia
INV075	2	Long-Term Investor	US
INV275	2	Long-Term Investor	Asia
INV182	2	Long-Term Investor	US
INV023	2	Long-Term Investor	Asia
INV220	2	Long-Term Investor	Asia
INV153	2	Long-Term Investor	Europe
INV261	2	Long-Term Investor	US
INV046	2	Long-Term Investor	US
INV119	2	Long-Term Investor	US
INV178	2	Long-Term Investor	US
INV251	2	Long-Term Investor	Europe
INV054	2	Long-Term Investor	US
INV240	2	Long-Term Investor	Europe
INV080	2	Long-Term Investor	US
INV283	2	Long-Term Investor	US
INV188	2	Long-Term Investor	US
INV065	2	Long-Term Investor	US
INV027	2	Long-Term Investor	Asia

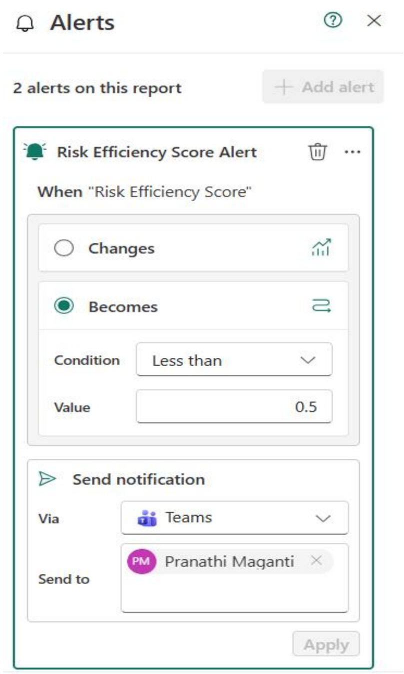
Table: gold_investor_trading_behavior (274 rows)

Publishing to Power BI service:

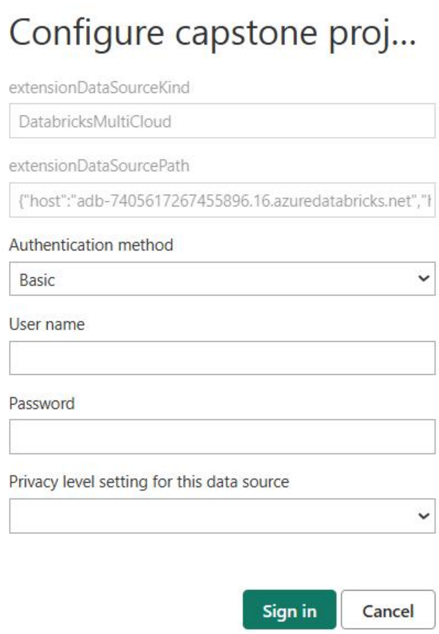


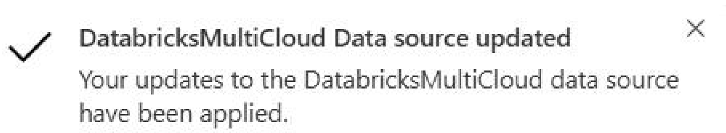


Managing Alerts in PowerBI:
An alert is created for Risk Efficiency Score KPI . if the value of this attribute decreases below a threshold of value 0.5 then an alert is automatically sent via email and teams.



Setting a connection between Databricks and PowerBI service:
The authentication method is chosen as basic and the databricks token is given as the password for connection.





Enabling Refresh behaviour in PowerBI: