ISM6208 Data Warehousing Final Project  
James Long

USF Summer 2021

Contents

[Executive Summary 2](#_Toc75973342)

[Problem Statement 2](#_Toc75973343)

[Data Collection and Preparation 2](#_Toc75973344)

[Database Design 3](#_Toc75973345)

[Operational Model 16](#_Toc75973346)

[Reporting 18](#_Toc75973347)

[Modeling 23](#_Toc75973348)

[Storytelling 31](#_Toc75973349)

[Conclusions 35](#_Toc75973350)

# Executive Summary

This paper presents the processes for and results of building a very basic predictive model for major U.S. stock indexes based on core economic data. The model is dimensional and based on a relational star schema. I am particularly interested in the influence the Federal Reserve has on the markets, so my model uses predictors aligned with the Fed's mandate (employment and inflation) along with the Federal Funds rate.

This basic model could evolve to include something like a “Fed Sentiment” fact table. The purpose would be to capture and quantify how hawkish or dovish Fed comments are whenever a member of the Fed makes public comments. Natural Language Processing (NLP) or other Artificial Intelligence (AI) techniques could be used to assign a numeric value to the sentiment indicator for each Fed comment. Due to lack of time and data, I modeled this concept but did not implement it.

This project is the culmination of effort performed over the span of a six-week semester. A large set of financial data was already available within the course DBMS, so I relied on it as my primary data source. This enabled me to focus more on modeling, visualizing, and mining. The project incorporates some of the work assigned earlier in the semester and builds upon it to meet the project requirements. Though given the option to work in a group, I worked alone to maximize my learning outcomes.

# Problem Statement

Accurately predicting stock moves is notoriously difficult yet extremely alluring due to the potential rewards. Everyone that has any investment in stock (401k, IRA, pension, retail broker, etc.) has a motive to solve this problem. Unfortunately, countless variables influence stock market movements, and the influence of each variable changes daily or even hourly. This creates both opportunity and risk every trading day in every stock market around the globe. For many families, solving this puzzle with even modest reliability could mean the difference between private transportation or public transit, buying a home or renting an apartment, private or public school. Otherwise, long term investing is the only safe option. While that helps with retirement, it does not pay the bills due today. Data mining and visualization could uncover patterns that help to solve this challenge.

# Data Collection and Preparation

Most of the data I used were already in the FIN schema. I replicated these data to my personal schema and modified them as needed by dropping columns, adding columns, renaming columns, populating columns, etc. In the case of the Employment\_Facts table, I also dropped some rows of older data that preceded the earliest month in the Cal\_Month\_Dim\_MV materialized view.

No data were available in the DMBS for inflation rates. So, I searched the FRED site for relevant series. Many series were indexed to a particular year, so they would not correlate with other data that fluctuated monthly. Fortunately, I found a mean Consumer Price Index (CPI) series produced by the Federal Reserve Bank of Dallas (FRBD) on the monthly basis. Prior to loading that into my schema, I dropped the rows of newer data that succeeded the latest month in the Cal\_Month\_Dim\_MV materialized view (July 2018).

No data were available in the DMBS for Fed fund rates after March 2012. So, I downloaded the same series from the FRED site, dropped the rows that were already in the DBMS, and then inserted new rows into the Fed\_Reserve\_Facts table to fill in missing data from April 2012 to July 2018.

No data were available in the DMBS for BLS unemployment rates after March 2015. So, I downloaded the same series from the FRED site, dropped the rows that were already in the DBMS, and then inserted new rows into the Employments\_Facts table to fill in missing data from April 2015 to July 2018.

# Database Design

The following ERD depicts the dimensional schema I created for this project.

Graphical user interface, application, Teams

Description automatically generated

**Monthly Grain**

I am not interested in quarterly aggregates because I want to know how to trade index ETFs leading up to Fed meetings, public Fed comments, events featuring Fed speakers, etc. Too many such events occur in a quarter for that time span to be useful. So, I only created a monthly aggregate for the date table. Most of the fact tables contain data produced on a monthly basis, so this grain works well for aggregation.

**52 Week Data**

The “52 week” facts in the STK\_Index\_Mon\_Aggs materialized view record the highest and lowest values during each aggregate period (not the average). The “52 week” dates in the STK\_Index\_Dim table are based on the trailing twelve months.

**World Events**

The World\_Events\_Dim table attempts to capture some of the market's psychological factors. This could help explain temporary deviations from the normal correlation of indicator facts to index price movements. For example, Pre\_Recession would indicate the month prior to the formal start of a recession, whereas Post\_Recession would indicate the month after the formal end of a recession. Some of the other attributes probably should be modeled granularly like recession. For now, they serve as placeholders to indicate the types of events that might be relevant.

**Fed Sentiment**

The grain of the Fed\_Sentiment\_Facts table is daily, so we expect the fact table to be sparsely populated. The date alone is insufficient to uniquely identify each row because multiple events can occur on a single day. So, a system-generated identity field is included in the primary key. An event type is included to inform us of the nature of the event at which comments were made, but this is modeled as a degenerate dimension on the assumption we do not need to know anything about specific events. An event weight is assigned to each event type to indicate the relevant importance. This enables us to calculate a weighted average of the sentiment level in the monthly aggregate view. We need this because some comments (e.g., FOMC meeting minutes or the dot plot) have more impact on market psychology than other comments (e.g., a Fed President speaking at a business conference). This sentiment concept was modeled but not instantiated since I had no data.

**SQL Code**

I issued the following SQL commands to instantiate the tables within my personal schema (DW787).

CREATE TABLE

STK\_DATE\_DIM

AS

SELECT

\*

FROM

FIN.STK\_DATE\_DIMS SDD

INNER JOIN

(SELECT

JULIAN\_DAY\_KEY AS DAY\_KEY,

TO\_NUMBER(TO\_CHAR(CAL\_DATE, 'YYYYMM'),'999999') AS CAL\_MONTH\_KEY

FROM

FIN.STK\_DATE\_DIMS) CMK

ON SDD.JULIAN\_DAY\_KEY=CMK.DAY\_KEY;

ALTER TABLE

STK\_DATE\_DIM

DROP COLUMN DAY\_KEY;

ALTER TABLE

STK\_DATE\_DIM

ADD CONSTRAINT

STK\_DATE\_DIM\_PK

PRIMARY KEY

(JULIAN\_DAY\_KEY);

ALTER TABLE

STK\_DATE\_DIM

ADD CONSTRAINT

STK\_DATE\_MONTH\_FK

FOREIGN KEY

(CAL\_MONTH\_KEY)

REFERENCES

CAL\_MONTH\_DIM\_MV(CAL\_MONTH\_KEY);

CREATE TABLE

STK\_INDEX\_DIM

AS

SELECT

\*

FROM

FIN.STK\_INDEX\_DIMS;

ALTER TABLE

STK\_INDEX\_DIM

ADD CONSTRAINT

STK\_INDEX\_DIM\_PK

PRIMARY KEY

(INDEX\_KEY);

ALTER TABLE

STK\_INDEX\_DIM

ADD HIGH\_52\_WEEK\_DATE DATE;

ALTER TABLE

STK\_INDEX\_DIM

ADD LOW\_52\_WEEK\_DATE DATE;

CREATE TABLE

STK\_INDEX\_FACTS

AS

SELECT

\*

FROM

FIN.STK\_INDEX\_FACTS IF

INNER JOIN

(SELECT

TRADE\_DATE AS TD,

INDEX\_KEY AS IK,

ROUND(MAX(HIGH) OVER (PARTITION BY INDEX\_KEY ORDER BY TRADE\_DATE ASC RANGE INTERVAL '365' DAY(3) PRECEDING),2) AS HIGH\_52\_WEEK,

ROUND(MIN(LOW) OVER (PARTITION BY INDEX\_KEY ORDER BY TRADE\_DATE ASC RANGE INTERVAL '365' DAY(3) PRECEDING),2) AS LOW\_52\_WEEK

FROM

FIN.STK\_INDEX\_FACTS

ORDER BY TRADE\_DATE, INDEX\_KEY) HL

ON IF.TRADE\_DATE=HL.TD AND IF.INDEX\_KEY=HL.IK;

ALTER TABLE

STK\_INDEX\_FACTS

DROP COLUMN TD;

ALTER TABLE

STK\_INDEX\_FACTS

DROP COLUMN IK;

ALTER TABLE

STK\_INDEX\_FACTS

ADD CONSTRAINT

STK\_INDEX\_FACTS\_PK

PRIMARY KEY

(INDEX\_KEY, JULIAN\_DAY\_KEY);

ALTER TABLE

STK\_INDEX\_FACTS

ADD CONSTRAINT

STK\_INDEX\_INDEX\_FK

FOREIGN KEY

(INDEX\_KEY)

REFERENCES

STK\_INDEX\_DIM(INDEX\_KEY);

ALTER TABLE

STK\_INDEX\_FACTS

ADD CONSTRAINT

STK\_INDEX\_JULIAN\_FK

FOREIGN KEY

(JULIAN\_DAY\_KEY)

REFERENCES

STK\_DATE\_DIM(JULIAN\_DAY\_KEY);

CREATE MATERIALIZED VIEW

CAL\_MONTH\_DIM\_MV

BUILD IMMEDIATE

REFRESH FORCE

ON DEMAND

AS

SELECT DISTINCT

CAL\_MONTH\_KEY,

MONTH\_NAME,

MONTH\_ABBREV,

MONTH\_NBR,

YEAR\_NAME,

YEAR\_NBR

FROM

STK\_DATE\_DIM;

ALTER MATERIALIZED VIEW

CAL\_MONTH\_DIM\_MV

ADD CONSTRAINT

CAL\_MONTH\_DIM\_MV\_PK PRIMARY KEY("CAL\_MONTH\_KEY");

ALTER MATERIALIZED VIEW

CAL\_MONTH\_DIM\_MV

ADD CONSTRAINT

CAL\_MONTH\_DIM\_MV\_PK\_NOT\_NULL check("CAL\_MONTH\_KEY" IS NOT NULL) ENABLE;

EXEC DBMS\_STATS.GATHER\_TABLE\_STATS('DW787', 'CAL\_MONTH\_DIM\_MV');

CREATE MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS

BUILD IMMEDIATE

REFRESH FORCE

ON DEMAND

AS

SELECT

CAL\_MONTH\_KEY,

INDEX\_KEY,

ROUND(AVG(OPEN),2) AS AVG\_OPEN,

ROUND(AVG(HIGH),2) AS AVG\_HIGH,

ROUND(AVG(LOW),2) AS AVG\_LOW,

ROUND(AVG(CLOSE),2) AS AVG\_CLOSE,

ROUND(AVG(ADJ\_CLOSE),2) AS AVG\_ADJ\_CLOSE,

ROUND(AVG(VOLUME),0) AS AVG\_VOLUME,

MAX(HIGH\_52\_WEEK) AS HIGH\_52\_WEEK,

MIN(LOW\_52\_WEEK) AS LOW\_52\_WEEK

FROM

STK\_DATE\_DIM DD

INNER JOIN STK\_INDEX\_FACTS IF

ON DD.JULIAN\_DAY\_KEY=IF.JULIAN\_DAY\_KEY

GROUP BY CAL\_MONTH\_KEY, INDEX\_KEY

ORDER BY CAL\_MONTH\_KEY, INDEX\_KEY;

ALTER MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS

ADD CONSTRAINT

SYS\_C12345678 check("CAL\_MONTH\_KEY" IS NOT NULL) ENABLE;

ALTER MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS

ADD CONSTRAINT

STK\_INDEX\_MON\_AGGS\_PK PRIMARY KEY("CAL\_MONTH\_KEY","INDEX\_KEY");

ALTER MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS

ADD CONSTRAINT

STK\_INDEX\_AGGS\_CAL\_MON\_FK

FOREIGN KEY

(CAL\_MONTH\_KEY)

REFERENCES

CAL\_MONTH\_DIM\_MV(CAL\_MONTH\_KEY);

ALTER MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS

ADD CONSTRAINT

STK\_INDEX\_AGGS\_INDEX\_FK

FOREIGN KEY

(INDEX\_KEY)

REFERENCES

STK\_INDEX\_DIM(INDEX\_KEY);

EXEC DBMS\_STATS.GATHER\_TABLE\_STATS('DW787', 'STK\_INDEX\_MON\_AGGS');

UPDATE

STK\_INDEX\_DIM

SET

HIGH\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=101

AND TRADE\_DATE >='11-JUL-17'

AND HIGH\_52\_WEEK=(SELECT

MAX(HIGH\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=101

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=101;

UPDATE

STK\_INDEX\_DIM

SET

HIGH\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=102

AND TRADE\_DATE >='11-JUL-17'

AND HIGH\_52\_WEEK=(SELECT

MAX(HIGH\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=102

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=102;

UPDATE

STK\_INDEX\_DIM

SET

HIGH\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=103

AND TRADE\_DATE >='11-JUL-17'

AND HIGH\_52\_WEEK=(SELECT

MAX(HIGH\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=103

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=103;

UPDATE

STK\_INDEX\_DIM

SET

LOW\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=101

AND TRADE\_DATE >='11-JUL-17'

AND LOW\_52\_WEEK=(SELECT

MIN(LOW\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=101

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=101;

UPDATE

STK\_INDEX\_DIM

SET

LOW\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=102

AND TRADE\_DATE >='11-JUL-17'

AND LOW\_52\_WEEK=(SELECT

MIN(LOW\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=102

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=102;

UPDATE

STK\_INDEX\_DIM

SET

LOW\_52\_WEEK\_DATE=

(SELECT

MIN(TRADE\_DATE)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=103

AND TRADE\_DATE >='11-JUL-17'

AND LOW\_52\_WEEK=(SELECT

MIN(LOW\_52\_WEEK)

FROM

STK\_INDEX\_FACTS

WHERE

INDEX\_KEY=103

AND TRADE\_DATE >='11-JUL-17'))

WHERE

INDEX\_KEY=103;

CREATE TABLE WORLD\_EVENTS\_DIM (

WORLD\_EVENTS\_KEY NUMBER(4,0) GENERATED BY DEFAULT AS IDENTITY,

PRE\_RECESSION VARCHAR2(5 BYTE),

IN\_RECESSION VARCHAR2(5 BYTE),

POST\_RECESSION VARCHAR2(5 BYTE),

WAR\_DECLARED VARCHAR2(5 BYTE),

WAR\_ENDED VARCHAR2(5 BYTE),

FEDERAL\_ELECTION\_MONTH VARCHAR2(5 BYTE),

FEDERAL\_ELECTION\_YEAR VARCHAR2(5 BYTE),

GEOPOLITICAL\_EVENT VARCHAR2(5 BYTE),

ENERGY\_EVENT VARCHAR2(5 BYTE),

IN\_PANDEMIC VARCHAR2(5 BYTE)

);

ALTER TABLE

WORLD\_EVENTS\_DIM

ADD CONSTRAINT

WORLD\_EVENTS\_DIM\_PK PRIMARY KEY ("WORLD\_EVENTS\_KEY");

INSERT INTO WORLD\_EVENTS\_DIM (

PRE\_RECESSION,

IN\_RECESSION,

POST\_RECESSION,

WAR\_DECLARED,

WAR\_ENDED,

FEDERAL\_ELECTION\_MONTH,

FEDERAL\_ELECTION\_YEAR,

GEOPOLITICAL\_EVENT,

ENERGY\_EVENT,

IN\_PANDEMIC)

VALUES

(NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);

CREATE TABLE

EMPLOYMENT\_FACTS

AS

SELECT

CAL\_MONTH\_KEY,

UNRATE\_PERCENT AS BLS\_UNEMP\_RATE,

CAST(NULL AS NUMBER) AS BLS\_PARTICIPATION\_RATE,

CAST(NULL AS NUMBER) AS BLS\_NONFARM\_PAYROLL\_EMP,

CAST(NULL AS NUMBER) AS ADP\_NATIONAL\_EMP,

1 AS WORLD\_EVENTS\_KEY

FROM

FIN.FRED\_UNRATE;

ALTER TABLE

EMPLOYMENT\_FACTS

ADD CONSTRAINT

EMPLOYMENT\_FACTS\_PK PRIMARY KEY ("CAL\_MONTH\_KEY");

DELETE

FROM

EMPLOYMENT\_FACTS

WHERE

CAL\_MONTH\_KEY IN (194801, 194802, 194803, 194804, 194805, 194806, 194807, 194808, 194809, 194810, 194811, 194812, 194901, 194902, 194903, 194904, 194905, 194906, 194907, 194908, 194909, 194910, 194911, 194912);

ALTER TABLE

EMPLOYMENT\_FACTS

ADD CONSTRAINT

EMPLOYMENT\_FACTS\_MONTH\_FK

FOREIGN KEY

("CAL\_MONTH\_KEY")

REFERENCES

"CAL\_MONTH\_DIM\_MV"("CAL\_MONTH\_KEY");

ALTER TABLE

EMPLOYMENT\_FACTS

ADD CONSTRAINT

EMPLOYMENT\_FACTS\_WORLD\_FK

FOREIGN KEY

("WORLD\_EVENTS\_KEY")

REFERENCES

"WORLD\_EVENTS\_DIM"("WORLD\_EVENTS\_KEY");

CREATE TABLE

FED\_RESERVE\_FACTS

AS

SELECT

CAL\_MONTH\_KEY,

FF\_RATE AS FED\_FUND\_RATE,

CAST(NULL AS NUMBER) AS QE\_PURCHASE\_AMT,

1 AS WORLD\_EVENTS\_KEY

FROM

FIN.FRED\_FEDFUNDS;

ALTER TABLE

FED\_RESERVE\_FACTS

ADD CONSTRAINT

FED\_RESERVE\_FACTS\_PK PRIMARY KEY ("CAL\_MONTH\_KEY");

ALTER TABLE

FED\_RESERVE\_FACTS

ADD CONSTRAINT

FRF\_PK\_NOT\_NULL CHECK ("CAL\_MONTH\_KEY" IS NOT NULL) ENABLE;

ALTER TABLE

FED\_RESERVE\_FACTS

ADD CONSTRAINT

FED\_RESERVE\_MONTH\_FK

FOREIGN KEY

("CAL\_MONTH\_KEY")

REFERENCES

"CAL\_MONTH\_DIM\_MV" ("CAL\_MONTH\_KEY");

ALTER TABLE

FED\_RESERVE\_FACTS

ADD CONSTRAINT

FED\_RESERVE\_WORLD\_FK

FOREIGN KEY

("WORLD\_EVENTS\_KEY")

REFERENCES

"WORLD\_EVENTS\_DIM" ("WORLD\_EVENTS\_KEY");

To load inflation data, I downloaded the FRED data in CSV format and created a temporary table to hold the data.

CREATE TABLE INFLATION\_TEMP (

PUB\_DATE DATE,

PCE NUMBER(5,2)

);

Next, I used the import wizard in SQL Developer to load the source data into the INFLATION\_TEMP table. Then I created the INFLATION\_FACTS table.

CREATE TABLE

INFLATION\_FACTS

AS

SELECT

TO\_NUMBER(TO\_CHAR(PUB\_DATE, 'YYYYMM'),'999999') AS CAL\_MONTH\_KEY,

PCE AS FRBD\_TRIMMED\_PCE\_RATE,

CAST(NULL AS NUMBER) AS BLS\_CPI\_RATE,

1 AS WORLD\_EVENTS\_KEY

FROM

INFLATION\_TEMP;

DROP TABLE INFLATION\_TEMP;

ALTER TABLE

INFLATION\_FACTS

ADD CONSTRAINT

INFLATION\_FACTS\_PK PRIMARY KEY ("CAL\_MONTH\_KEY");

ALTER TABLE

INFLATION\_FACTS

ADD CONSTRAINT

INFLATION\_FACTS\_PK\_NOT\_NULL CHECK ("CAL\_MONTH\_KEY" IS NOT NULL) ENABLE;

ALTER TABLE

INFLATION\_FACTS

ADD CONSTRAINT

INFLATION\_FACTS\_MONTH\_FK

FOREIGN KEY

("CAL\_MONTH\_KEY")

REFERENCES

"CAL\_MONTH\_DIM\_MV" ("CAL\_MONTH\_KEY");

ALTER TABLE

INFLATION\_FACTS

ADD CONSTRAINT

INFLATION\_FACTS\_WORLD\_FK

FOREIGN KEY

("WORLD\_EVENTS\_KEY")

REFERENCES

"WORLD\_EVENTS\_DIM" ("WORLD\_EVENTS\_KEY");

To load additional rows of newer Fed funds rate data, I downloaded the FRED data in CSV format and then used Excel functions to extract the year and month from the date field. These were concatenated into a Cal\_Month\_Key field in the CSV file. I also created a World\_Events\_Key field in the CSV file with a value of 1 in each row. I then used the import wizard in SQL Developer to load newer data into the Fed\_Reserve\_Facts table.

To load additional rows of newer BLS unemployment rate data, I followed the same procedure as for the Fed funds rate data except the data were loaded into the Employment\_Facts table.

Following the June 21 lecture, I issued the following SQL statements.

ALTER MATERIALIZED VIEW

CAL\_MONTH\_DIM\_MV ENABLE QUERY REWRITE;

ALTER MATERIALIZED VIEW

STK\_INDEX\_MON\_AGGS ENABLE QUERY REWRITE;

# Operational Model

The operational systems that generate trade fact data might look something like the following ERD. This partial model is shown with a high-level view of the ETL process.

Diagram

Description automatically generated

# Reporting

**Query 1 - Copied from Assignment 2**

This query examines the correlation of the Fed Funds rate with the BLS unemployment rate on a calendar year basis and reports the strongest negative and positive correlations along with the total number of negative and positive correlations.

select

\*

from

(SELECT

MIN(rate\_corr) as "strongest neg corr",

COUNT(\*) as "number of neg corr"

FROM

(SELECT

year\_nbr,

round(corr(fed\_fund\_rate, bls\_unemp\_rate),4) as rate\_corr

FROM

employment\_facts ef

inner join cal\_month\_dim\_mv md

on ef.cal\_month\_key=md.cal\_month\_key

inner join fed\_reserve\_facts frf

on md.cal\_month\_key=frf.cal\_month\_key

group by year\_nbr

order by rate\_corr)

where rate\_corr < 0)

full join

(SELECT

MAX(rate\_corr) as "strongest pos correlation",

COUNT(\*) as "number of pos correlation"

FROM

(SELECT

year\_nbr,

round(corr(fed\_fund\_rate, bls\_unemp\_rate),4) as rate\_corr

FROM

employment\_facts ef

inner join cal\_month\_dim\_mv md

on ef.cal\_month\_key=md.cal\_month\_key

inner join fed\_reserve\_facts frf

on md.cal\_month\_key=frf.cal\_month\_key

group by year\_nbr

order by rate\_corr)

where rate\_corr > 0)

on 1=0;

**Query 2 - Adapted from Assignment 2**

This query examines the correlation of stock index price range with volume on an index, monthly, and yearly basis.

SELECT

\*

FROM

(SELECT

ROUND(CORR(avg\_range, avg\_volume),4) as Index\_Corr

FROM

(SELECT

index\_name,

(CASE

WHEN GROUPING(year\_nbr) = 1 THEN 'Avg'

ELSE TO\_CHAR(year\_nbr)

END) as year,

(CASE

WHEN GROUPING(month\_nbr) = 1 THEN 'Avg'

ELSE TO\_CHAR(month\_nbr)

END) as month,

(ROUND(AVG(high),2)-ROUND(AVG(low),2)) as avg\_range,

ROUND(AVG(volume)) as avg\_volume

FROM

stk\_date\_dim dd

inner join stk\_index\_facts if

on dd.julian\_day\_key = if.julian\_day\_key

inner join stk\_index\_dim id

on if.index\_key = id.index\_key

GROUP BY index\_name, ROLLUP(year\_nbr, month\_nbr)

ORDER BY index\_name, year\_nbr, month\_nbr)

where year = 'Avg')

full join

(SELECT

ROUND(CORR(avg\_range, avg\_volume),4) as Yearly\_Corr

FROM

(SELECT

index\_name,

(CASE

WHEN GROUPING(year\_nbr) = 1 THEN 'Avg'

ELSE TO\_CHAR(year\_nbr)

END) as year,

(CASE

WHEN GROUPING(month\_nbr) = 1 THEN 'Avg'

ELSE TO\_CHAR(month\_nbr)

END) as month,

(ROUND(AVG(high),2)-ROUND(AVG(low),2)) as avg\_range,

ROUND(AVG(volume)) as avg\_volume

FROM

stk\_date\_dim dd

inner join stk\_index\_facts if

on dd.julian\_day\_key = if.julian\_day\_key

inner join stk\_index\_dim id

on if.index\_key = id.index\_key

GROUP BY index\_name, year\_nbr, ROLLUP(month\_nbr)

ORDER BY index\_name, year\_nbr, month\_nbr)

where month = 'Avg'

and avg\_volume != 0)

on 1=0

full join

(SELECT

ROUND(CORR(avg\_range, avg\_volume),4) as Monthly\_Corr

FROM

(SELECT

index\_name,

year\_nbr as year,

month\_nbr as month,

(ROUND(AVG(high),2)-ROUND(AVG(low),2)) as avg\_range,

ROUND(AVG(volume)) as avg\_volume

FROM

stk\_date\_dim dd

inner join stk\_index\_facts if

on dd.julian\_day\_key = if.julian\_day\_key

inner join stk\_index\_dim id

on if.index\_key = id.index\_key

GROUP BY index\_name, year\_nbr, month\_nbr

ORDER BY index\_name, year\_nbr, month\_nbr)

where avg\_volume != 0)

on 1=0;

**Query 3**

This query set reports the three best and worst performing months for each stock index.

/\* create view to avoid repeatedly generating base data \*/

CREATE VIEW MONTHLY\_AVGS

AS

SELECT

index\_name,

(CASE

WHEN month\_nbr = 1 THEN 'Jan'

WHEN month\_nbr = 2 THEN 'Feb'

WHEN month\_nbr = 3 THEN 'Mar'

WHEN month\_nbr = 4 THEN 'Apr'

WHEN month\_nbr = 5 THEN 'May'

WHEN month\_nbr = 6 THEN 'Jun'

WHEN month\_nbr = 7 THEN 'Jul'

WHEN month\_nbr = 8 THEN 'Aug'

WHEN month\_nbr = 9 THEN 'Sep'

WHEN month\_nbr = 10 THEN 'Oct'

WHEN month\_nbr = 11 THEN 'Nov'

WHEN month\_nbr = 12 THEN 'Dec'

END) AS MONTH\_NAME,

avg\_high,

avg\_low

FROM

(SELECT

index\_name,

(CASE

WHEN GROUPING(year\_nbr) = 1 THEN 'Avg'

ELSE TO\_CHAR(year\_nbr)

END) as year,

month\_nbr,

ROUND(AVG(high),2) as avg\_high,

ROUND(AVG(low),2) as avg\_low

FROM

stk\_date\_dim dd

inner join stk\_index\_facts if

on dd.julian\_day\_key = if.julian\_day\_key

inner join stk\_index\_dim id

on if.index\_key = id.index\_key

GROUP BY index\_name, ROLLUP(year\_nbr), month\_nbr

ORDER BY index\_name, year\_nbr, month\_nbr)

where year = 'Avg'

ORDER BY INDEX\_NAME;

/\* best performing months \*/

SELECT

\*

FROM

((SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_high desc) as high\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'Dow Jones Industrial Average'

ORDER BY HIGH\_RANK

FETCH FIRST 3 ROWS ONLY)

UNION

(SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_high desc) as high\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'S&' || 'P 500'

ORDER BY HIGH\_RANK

FETCH FIRST 3 ROWS ONLY)

UNION

(SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_high desc) as high\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'Russell 2000'

ORDER BY HIGH\_RANK

FETCH FIRST 3 ROWS ONLY))

ORDER BY HIGH\_RANK, INDEX\_NAME;

/\* worst performing months \*/

SELECT

\*

FROM

((SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_low asc) as low\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'Dow Jones Industrial Average'

ORDER BY LOW\_RANK

FETCH FIRST 3 ROWS ONLY)

UNION

(SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_low asc) as low\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'S&' || 'P 500'

ORDER BY LOW\_RANK

FETCH FIRST 3 ROWS ONLY)

UNION

(SELECT

index\_name,

MONTH\_NAME,

RANK() OVER (PARTITION BY index\_name ORDER BY avg\_low asc) as low\_rank

FROM

MONTHLY\_AVGS

WHERE index\_name = 'Russell 2000'

ORDER BY LOW\_RANK

FETCH FIRST 3 ROWS ONLY))

ORDER BY LOW\_RANK, INDEX\_NAME;

# Modeling

For the modeling requirement, I wanted to use the Oracle Data Miner feature. Since I do not have access to it outside of class, this was a good opportunity to gain exposure. Unfortunately, the student data mining accounts did not have the needed permissions to load data into the data mining tablespace. So, I used Orange instead.

First, I created a flattened materialized view of the intermediate calculations I needed. The earliest record date was inconsistent across the fact tables, so I trimmed them to the oldest common date. The resulting view had 40 years of data, which is still plenty for my analysis. In fact, one could argue that the financial markets are sufficiently different today versus 40 years ago to make data that old useless for a predictive model. Nonetheless, I kept all 40 years. I also chose to focus on the S&P 500 index rather than incorporate all of the indexes.

Next, based on the first view, I created another flattened materialized view of the final calculations I needed. Perhaps this could have been accomplished with a single view, but I when I tried, I got SQL windowing errors that I did not have time to resolve. Besides, I think the code is easier to understand with two views, which makes maintenance less error prone. Also, the first view can be used by other derivative views, thus forming the basis of a modular schema to support future data mining efforts.

CREATE MATERIALIZED VIEW

DM\_INTERMED\_MV

AS

SELECT

IMA.CAL\_MONTH\_KEY,

INDEX\_NAME,

AVG\_ADJ\_CLOSE,

FED\_FUND\_RATE AS FED\_FUNDS,

LAG(FED\_FUND\_RATE, 1, FED\_FUND\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC)

AS FED\_FUNDS\_LAG1,

FED\_FUND\_RATE - (LAG(FED\_FUND\_RATE, 1, FED\_FUND\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC))

AS FED\_FUNDS\_CHG,

FRBD\_TRIMMED\_PCE\_RATE AS INFLATION,

LAG(FRBD\_TRIMMED\_PCE\_RATE, 1, FRBD\_TRIMMED\_PCE\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC)

AS INFLATION\_LAG1,

FRBD\_TRIMMED\_PCE\_RATE - (LAG(FRBD\_TRIMMED\_PCE\_RATE, 1, FRBD\_TRIMMED\_PCE\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC))

AS INFLATION\_CHG,

BLS\_UNEMP\_RATE AS UNEMPLOYMENT,

LAG(BLS\_UNEMP\_RATE, 1, BLS\_UNEMP\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC)

AS UNEMPLOYMENT\_LAG1,

BLS\_UNEMP\_RATE - (LAG(BLS\_UNEMP\_RATE, 1, BLS\_UNEMP\_RATE)

OVER (ORDER BY IMA.CAL\_MONTH\_KEY ASC))

AS UNEMPLOYMENT\_CHG

FROM

STK\_INDEX\_MON\_AGGS IMA

INNER JOIN FED\_RESERVE\_FACTS FRF

ON IMA.CAL\_MONTH\_KEY=FRF.CAL\_MONTH\_KEY

INNER JOIN INFLATION\_FACTS IF

ON IMA.CAL\_MONTH\_KEY=IF.CAL\_MONTH\_KEY

INNER JOIN EMPLOYMENT\_FACTS EF

ON IMA.CAL\_MONTH\_KEY=EF.CAL\_MONTH\_KEY

INNER JOIN STK\_INDEX\_DIM ID

ON IMA.INDEX\_KEY=ID.INDEX\_KEY

WHERE

INDEX\_NAME='S&'||'P 500'

AND IMA.CAL\_MONTH\_KEY >= 197801;

EXEC DBMS\_STATS.GATHER\_TABLE\_STATS('DW787', 'DM\_INTERMED\_MV');

CREATE MATERIALIZED VIEW

DM\_STK\_INDEX\_MV

AS

SELECT

CAL\_MONTH\_KEY,

INDEX\_NAME,

AVG\_ADJ\_CLOSE,

FED\_FUNDS,

FED\_FUNDS\_LAG1,

FED\_FUNDS\_CHG,

ROUND(FED\_FUNDS\_CHG / FED\_FUNDS\_LAG1,2) AS FED\_FUNDS\_PCT\_CHG,

ROUND(AVG(FED\_FUNDS\_CHG / FED\_FUNDS\_LAG1)

OVER (ORDER BY CAL\_MONTH\_KEY ASC ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2)

AS FED\_FUNDS\_PCT\_CHG\_SMA,

INFLATION,

INFLATION\_LAG1,

INFLATION\_CHG,

ROUND(INFLATION\_CHG / INFLATION\_LAG1,2) AS INFLATION\_PCT\_CHG,

ROUND(AVG(INFLATION\_CHG / INFLATION\_LAG1)

OVER (ORDER BY CAL\_MONTH\_KEY ASC ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2)

AS INFLATION\_PCT\_CHG\_SMA,

UNEMPLOYMENT,

UNEMPLOYMENT\_LAG1,

UNEMPLOYMENT\_CHG,

ROUND(UNEMPLOYMENT\_CHG / UNEMPLOYMENT\_LAG1,2) AS UNEMPLOYMENT\_PCT\_CHG,

ROUND(AVG(UNEMPLOYMENT\_CHG / UNEMPLOYMENT\_LAG1)

OVER (ORDER BY CAL\_MONTH\_KEY ASC ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2)

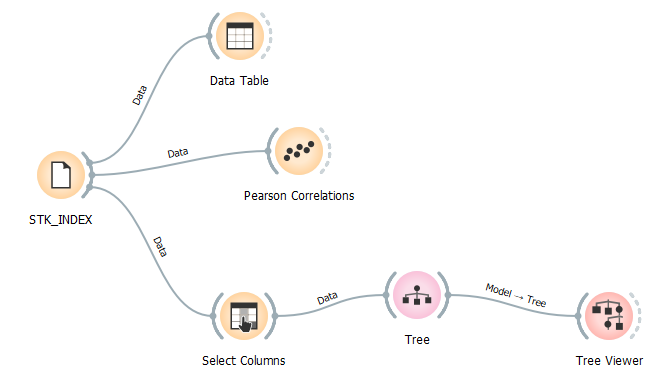
AS UNEMPLOYMENT\_PCT\_CHG\_SMA

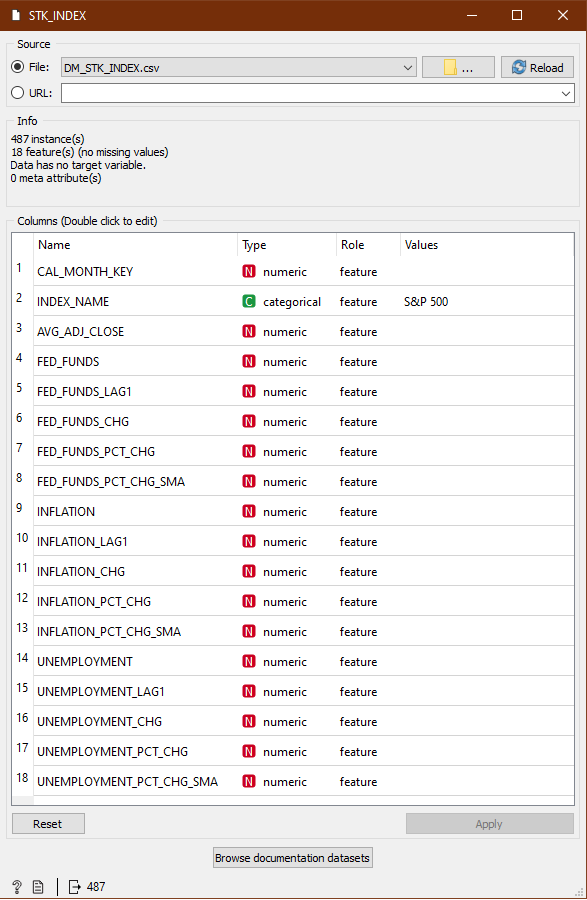
FROM

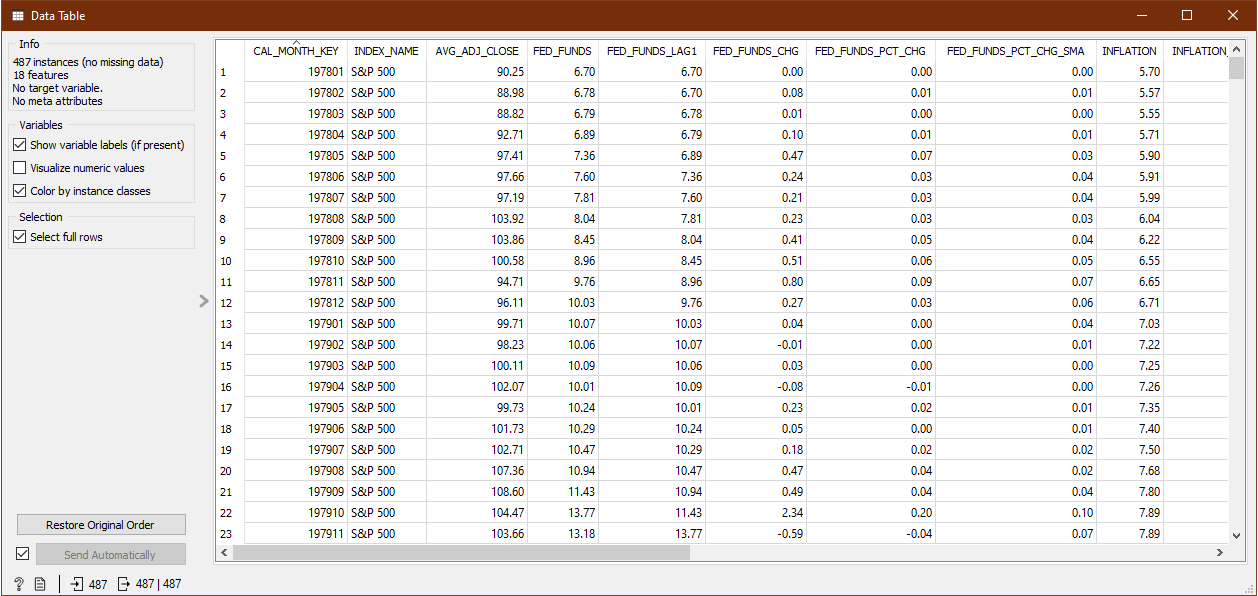
DM\_INTERMED\_MV;

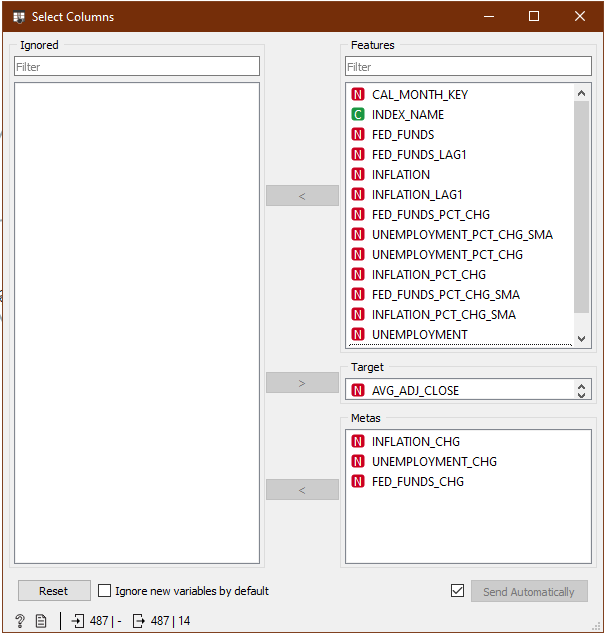
EXEC DBMS\_STATS.GATHER\_TABLE\_STATS('DW787', 'DM\_STK\_INDEX\_MV');

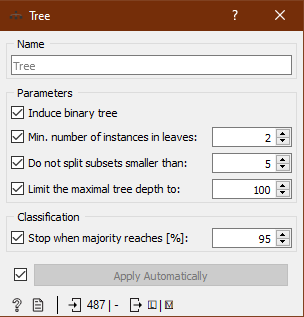
I exported the second view to a CSV file and imported into Orange. Then I followed the same decision tree analysis that was demonstrated in class. My target was AVG\_ADJ\_CLOSE, and my initial meta was FED\_FUNDS\_PCT\_CHG\_SMA. However, the resulting tree had 347 nodes, 174 leaves, and was 14 levels deep. So, I tried various other metas alone and in combinations. The results were all similar.

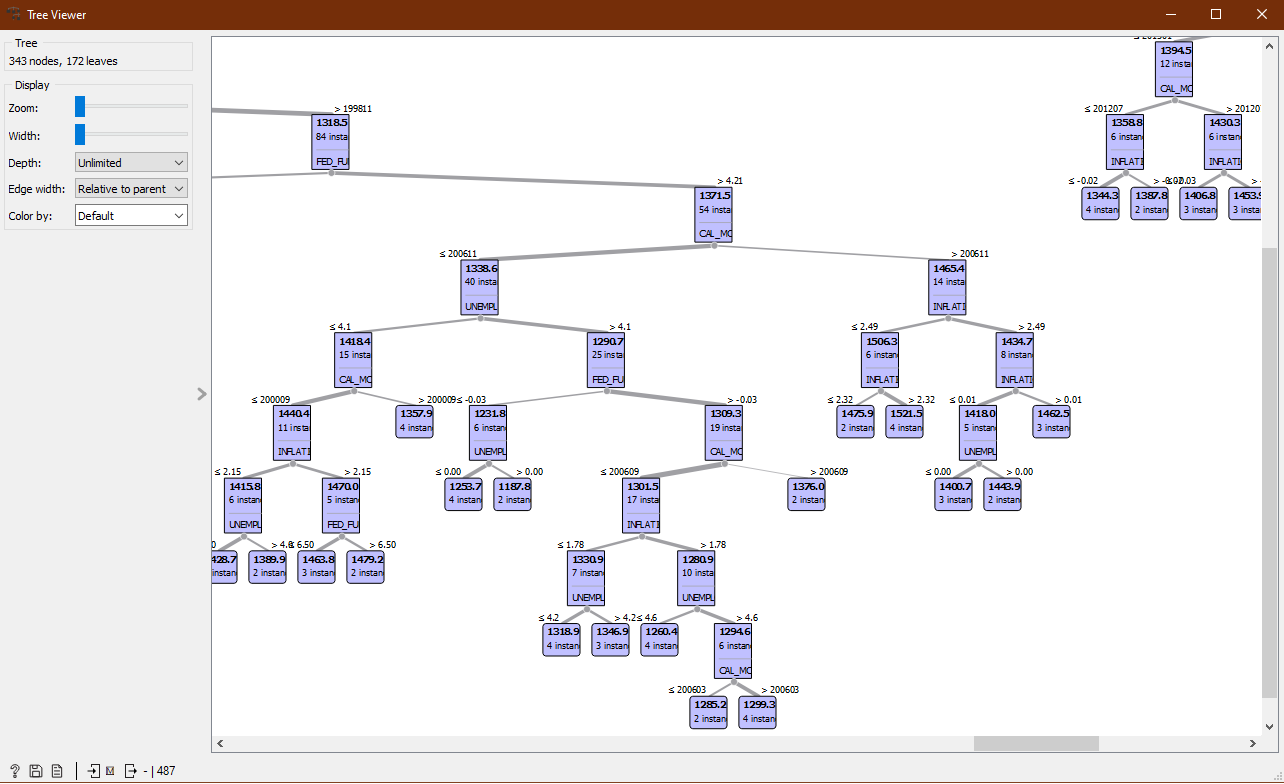




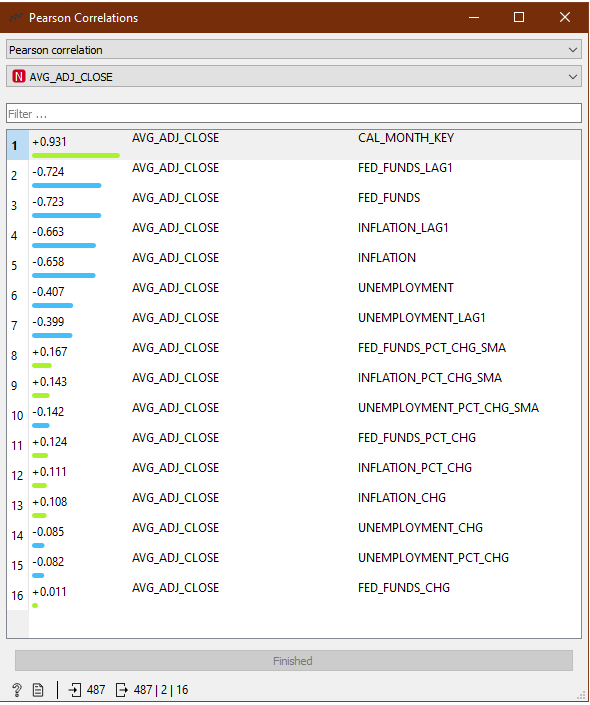








Then I ran some Pearson correlations to see if any useful insights could be gleaned. The FED\_FUNDS\_LAG1 had the strongest correlation, but it was not strong enough to be considered reliable. Just for kicks, I re-ran the decision tree using FED\_FUNDS\_LAG1 as the only meta, but the results were essentially the same as the other trees. I also tried ignoring all features except CAL\_MONTH\_KEY, but the tree results were still essentially the same.

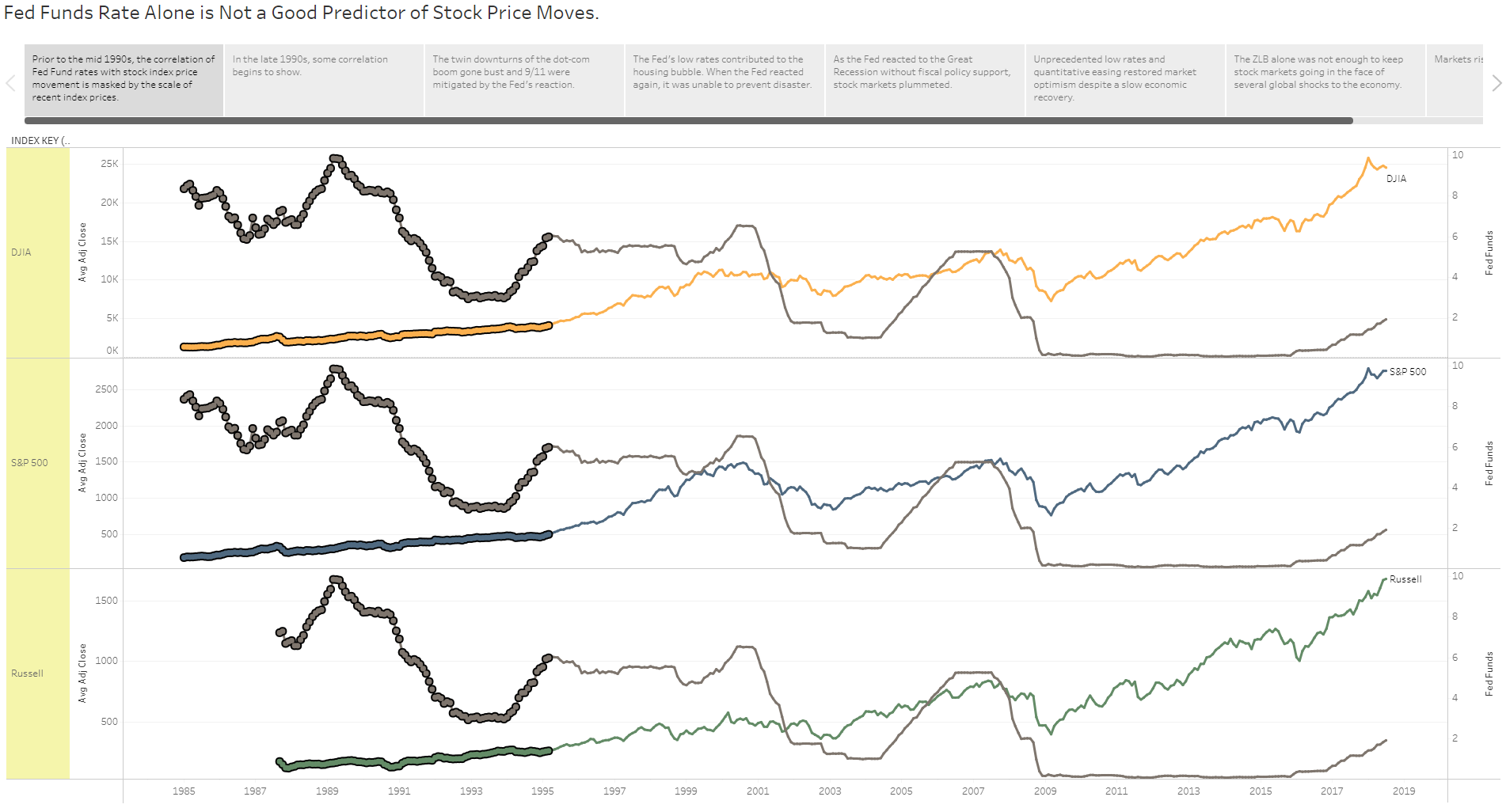


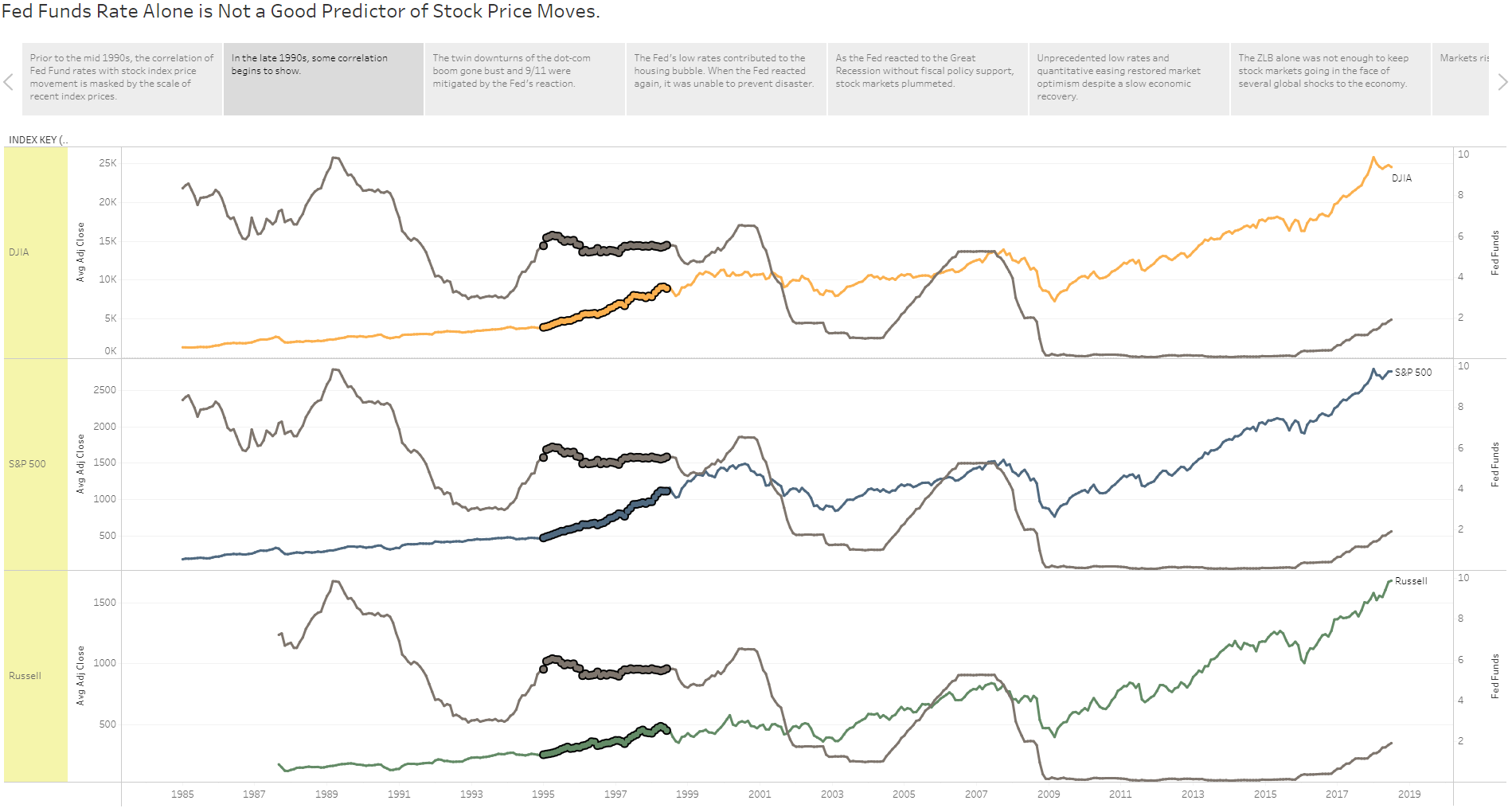
I have not yet taken the Data Mining course, so I am not completely certain how to interpret the decision tree results. However, the large number of leaf nodes seems to suggest none of the chosen economic data are reliable predictors of S&P 500 price movement. The low correlations are certainly evidence of a weak predictive model. These observations strengthen the argument for incorporating other indicators, such as a “Fed Sentiment” indicator, into the model. In my experience, the market usually reacts to Fed comments and other indicators of the Fed’s mindset, which always precede Fed actions.

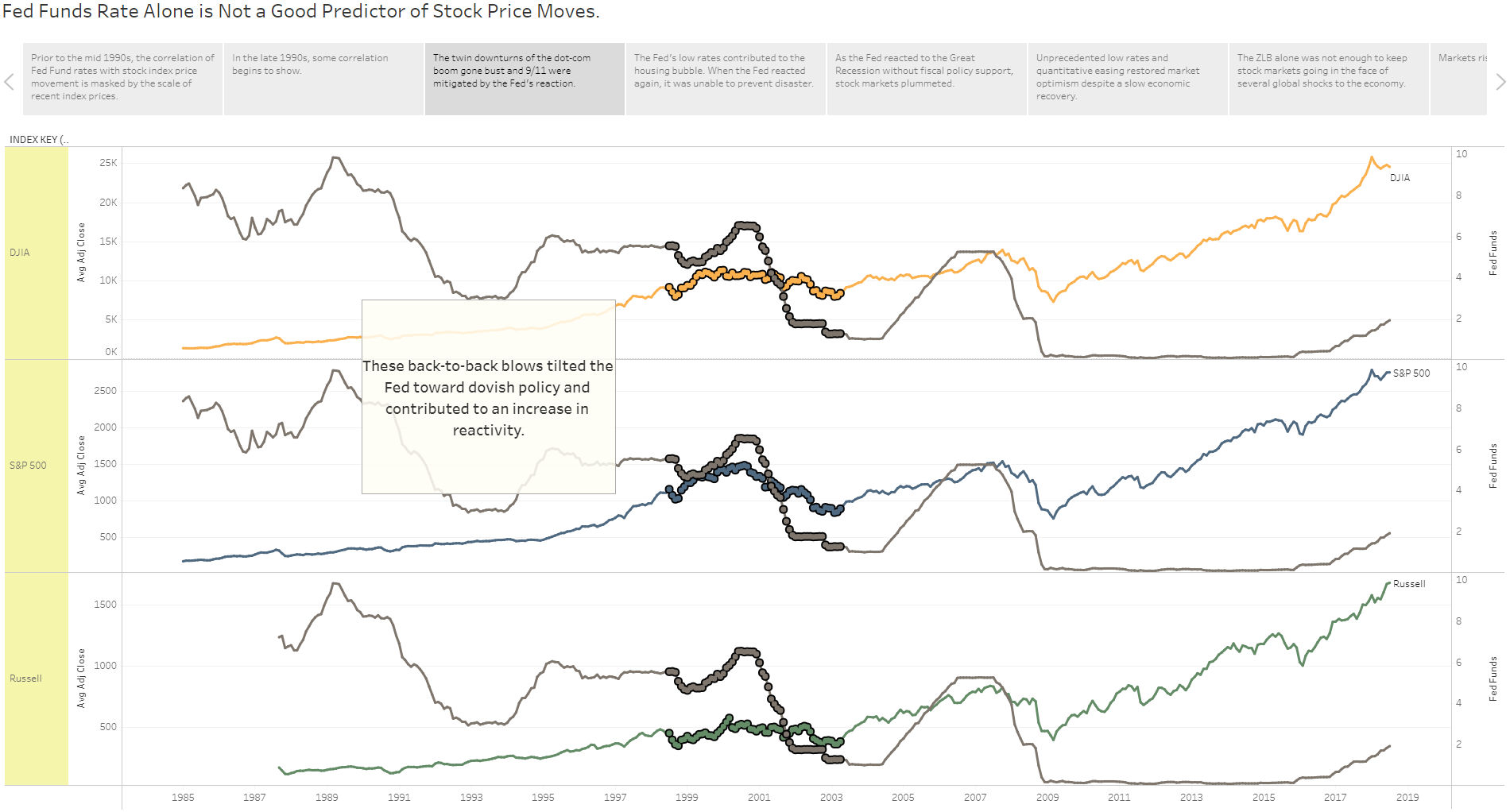
# Storytelling

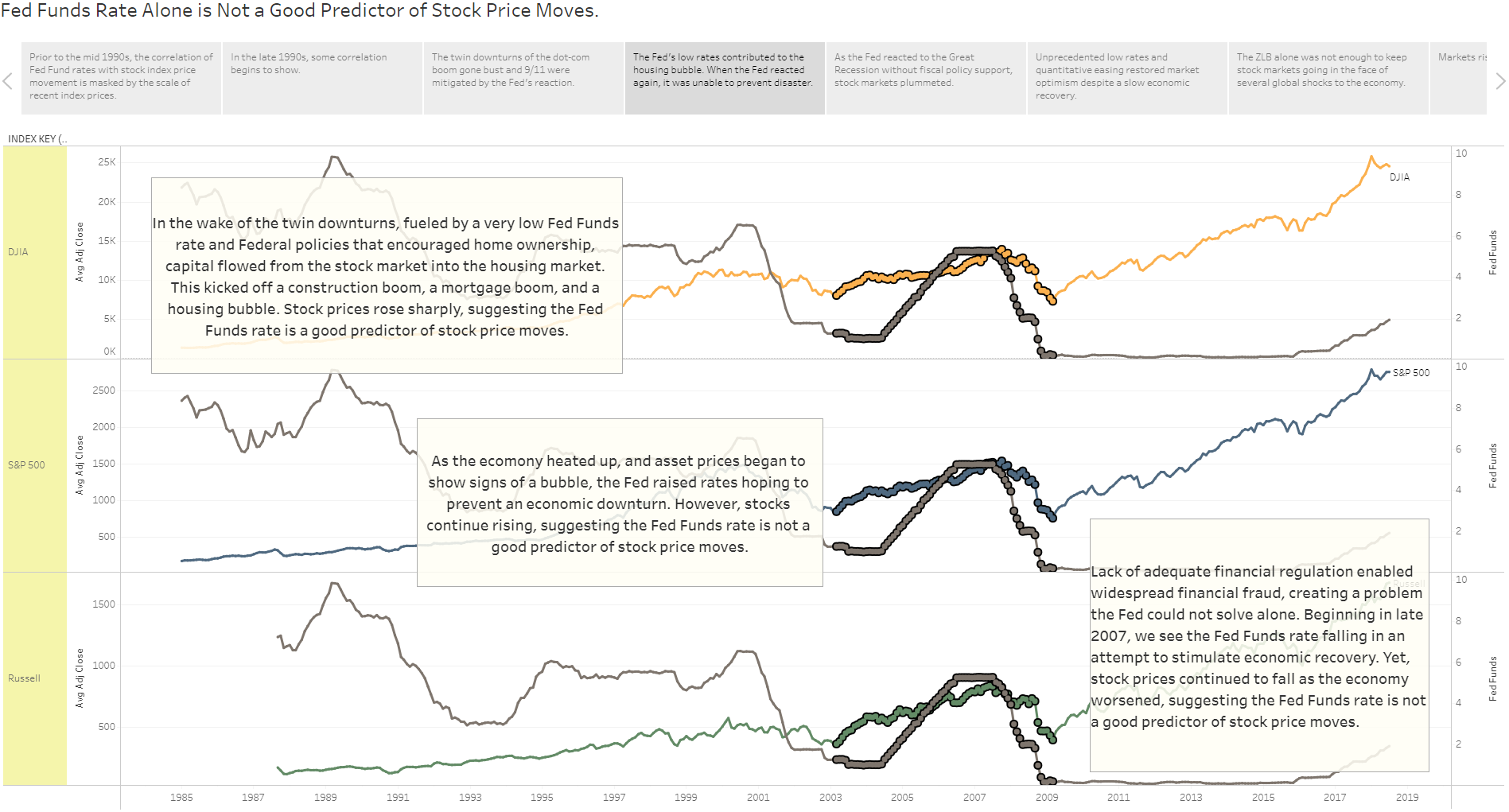
For the storytelling requirement, I created a visualization in Tableau using the story telling feature and then [published it to Tableau Public](https://public.tableau.com/app/profile/james.long3817/viz/ISM6208FinalProjectViz/FedFundsStory). Screenshots are included herein should the published workbook not stand the test of time. I would like to have shown the monthly data in more detail to reveal more volatility in the stock indexes. However, with decades of data, the story would have become a saga, and the theme would have been overwhelmed by detail. Despite the smoothing, some coarse correlations (or lack thereof) are visible that reveal an interesting story.

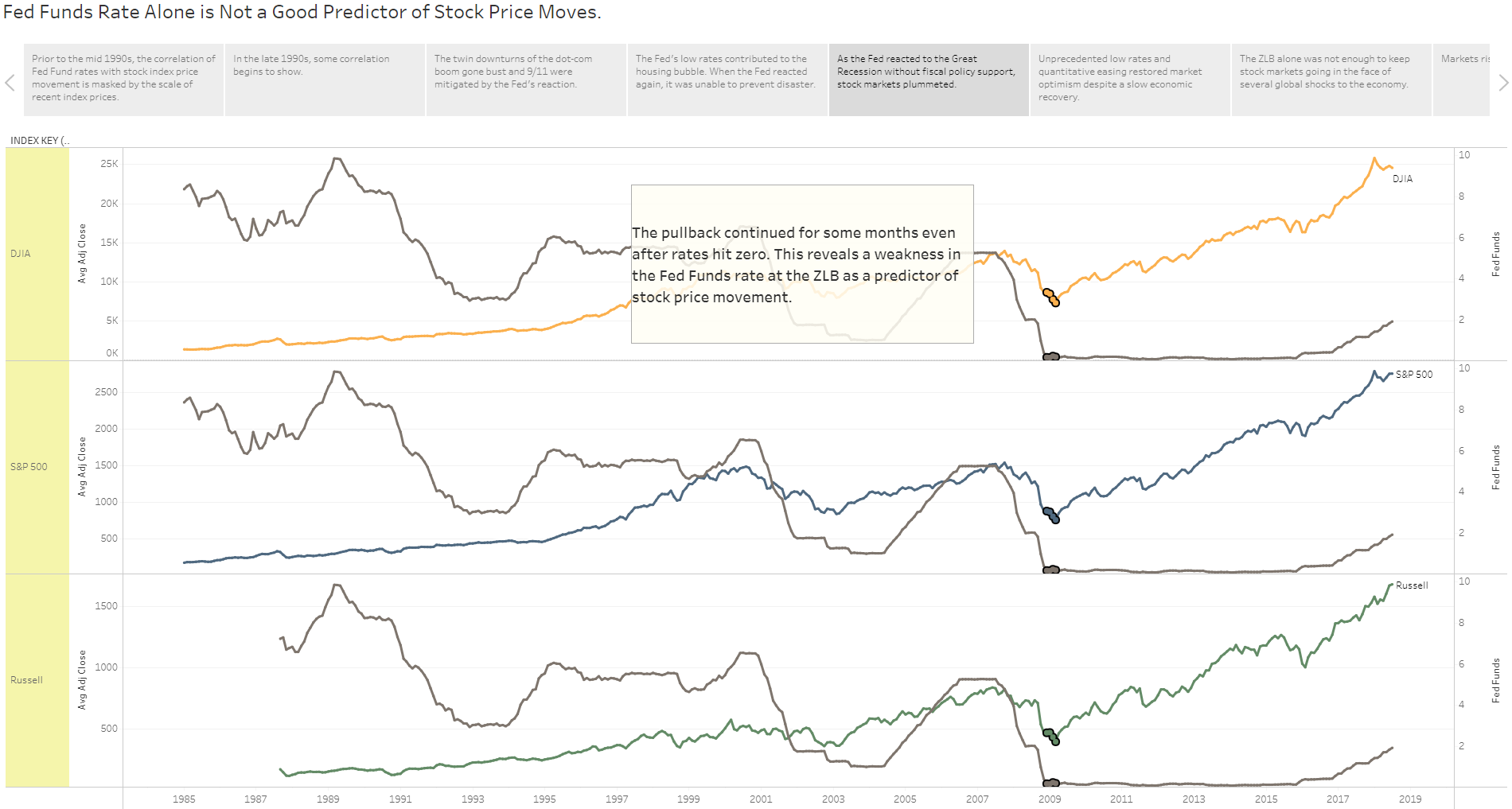
Rather than incorporate all of the candidate predictors, I focused on the Fed Funds rate. Since the data mining effort revealed the strongest correlation to be with the Fed Funds rate, I thought it provided the best opportunity for a useful story to be revealed. In a longer semester, I would have had time to explore the story more fully with multiple predictors.

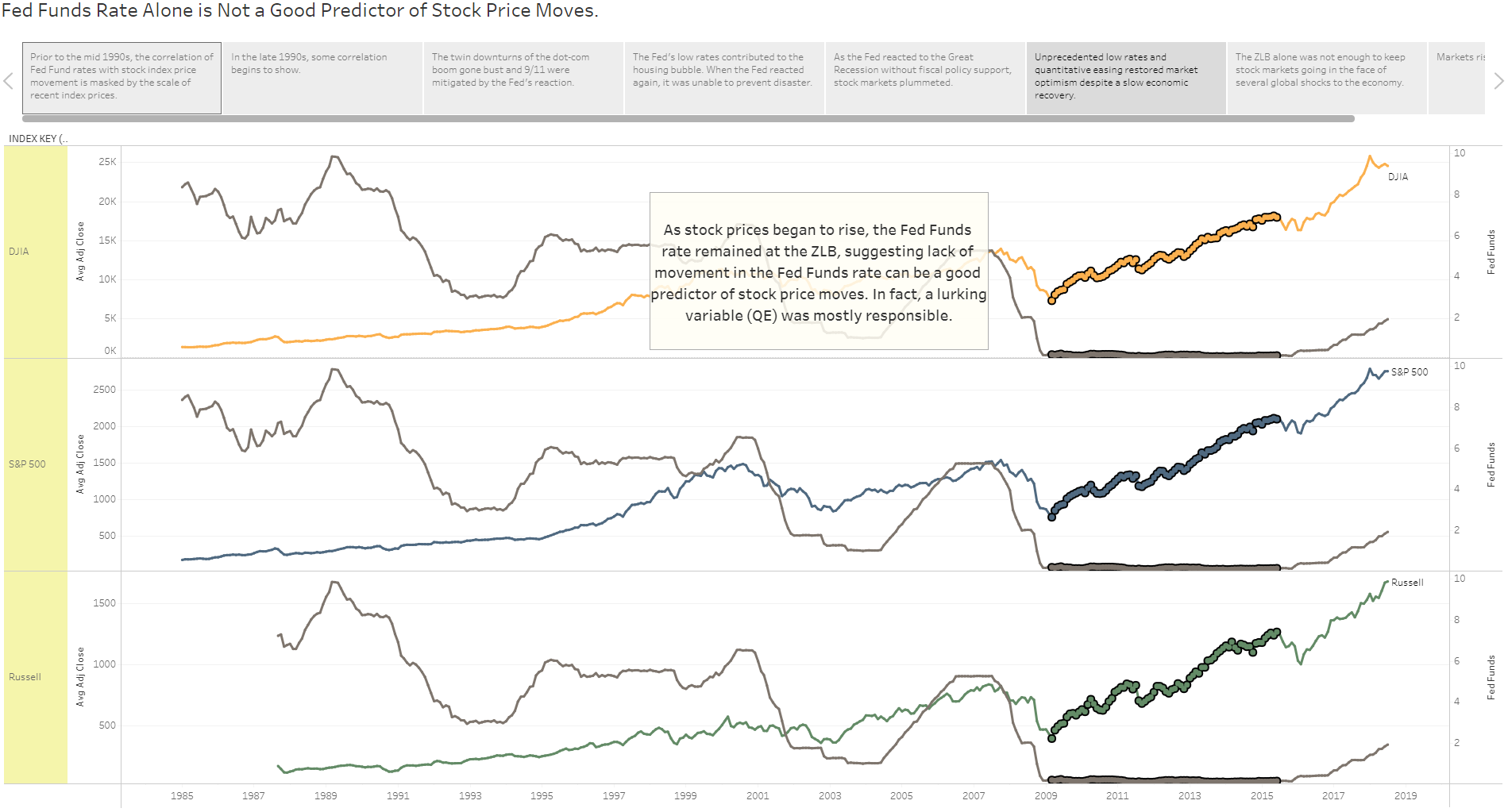


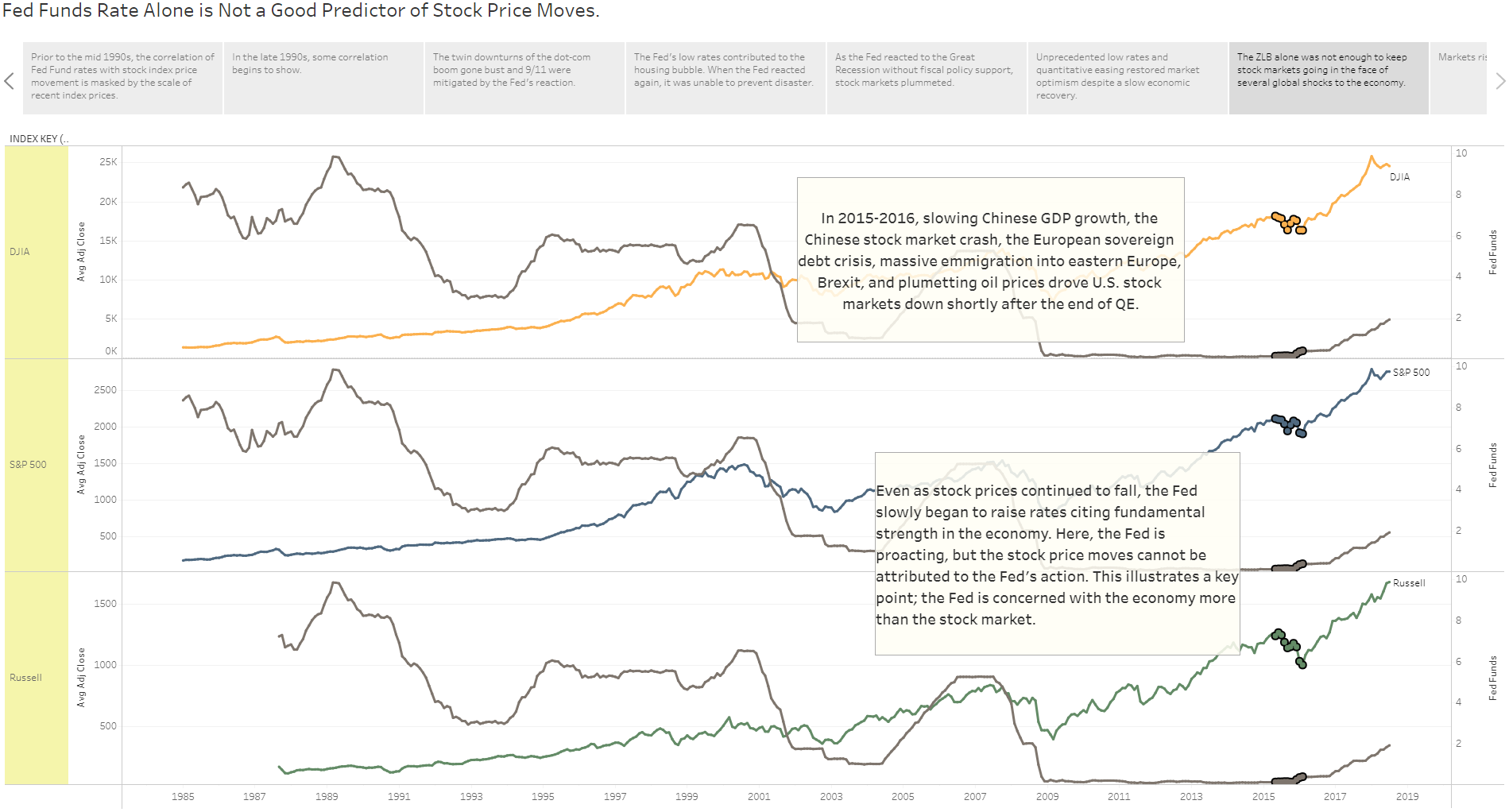


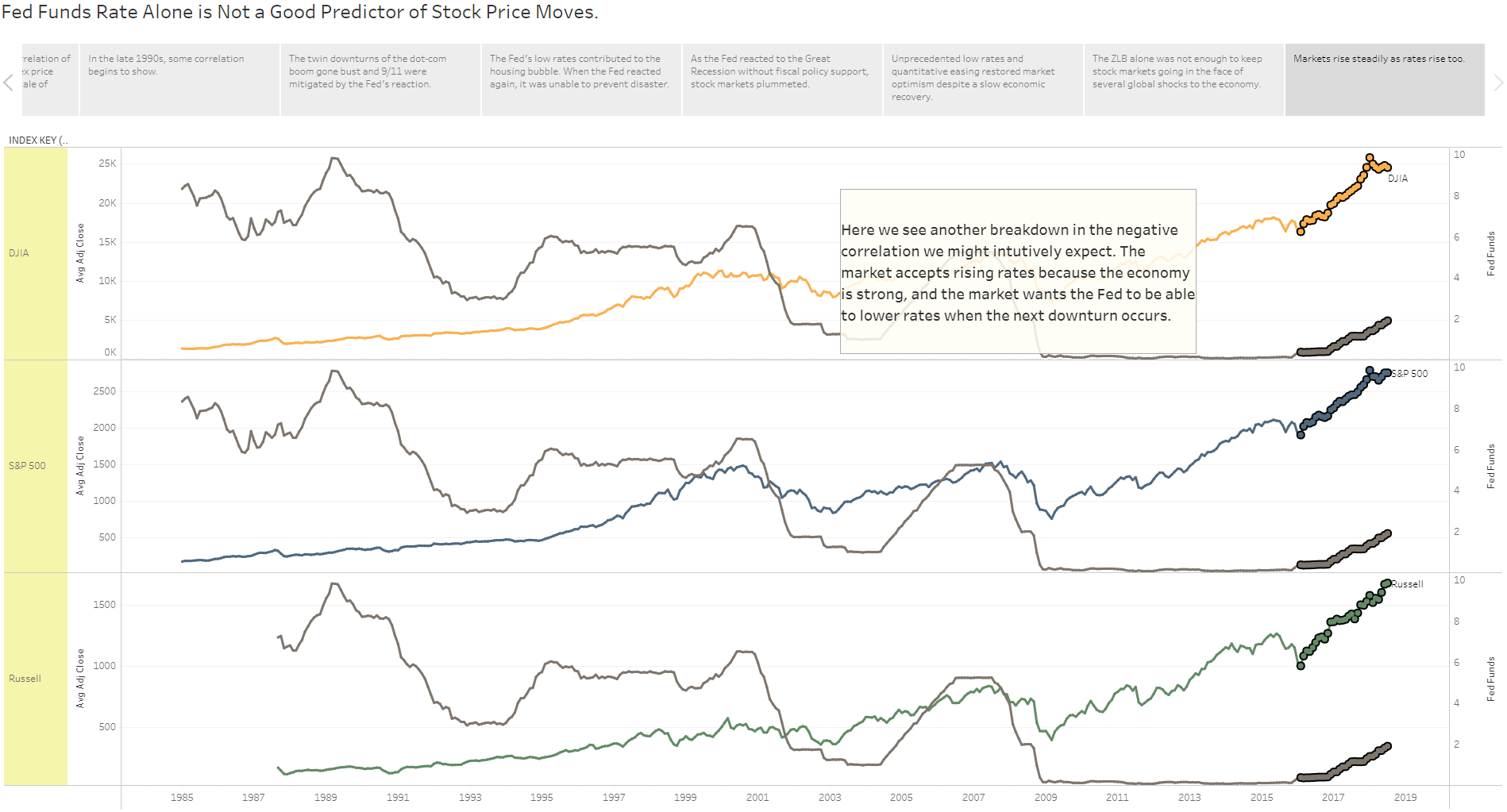












The key take-aways are:

1. The Fed Funds rate alone is not a good predictor of stock index price movement. This visualization corroborates the data mining results.
2. Other Fed tools, like Quantitative Easing (QE), can have even greater influence on stock markets than rate policy.
3. The influence of Fed Funds rate movement can only be properly interpreted by considering other factors such as inflation rates, employment rates, QE activity, world events, financial markets outside the U.S., and so on.

# Conclusions

This project was valuable to me because I learned much about data warehousing. But the value was increased by taking the granular facts all the way through mining and visualization. I can now appreciate the importance of schema design and ETL processes for a quality finished product. And I see how difficult it would be to build a reliable prediction model for anything of even modest complexity.

Perhaps the most valuable lesson is that a model is only as good as its data. Without the right inputs, everything else is a waste of time. In my model, a key data set was missing. The Fed’s communication precedes and has more influence on stock market psychology than Fed actions (such as adjusting the Fed Funds rate). Without this (and perhaps other) key data, my model is a weak predictor of stock index price movement.