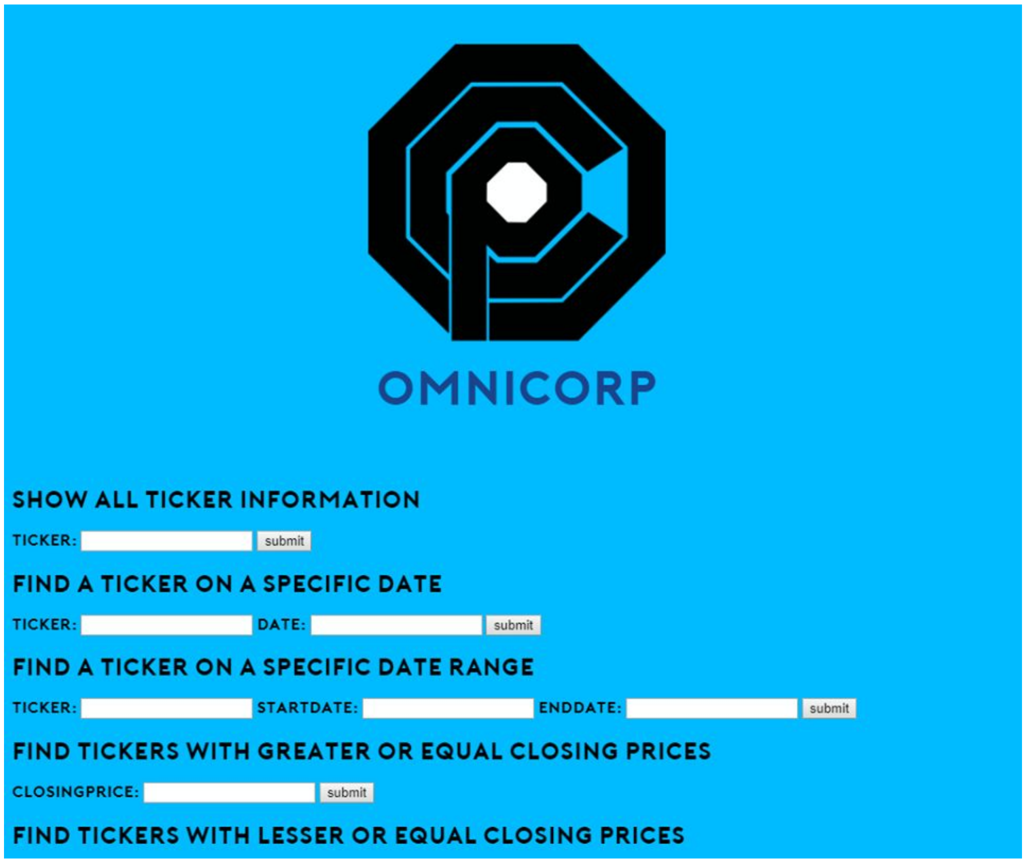
OmniCorp

1. Cover Page
    a. Team Members
        i. Gary Qian
        ii. Eric Tsai
    b. Section
        i. 315
        ii. 315
    c. Email
        i. gqian1@jhu.edu
        ii. etsai7@jhu.edu
    d. Description of application domain
        i. Stock Market data analysis system where users will be able to query the information of a certain stock and its prices depending on the given parameters such as high/low prices or on certain dates. The stocks are also able to provide information regarding the company such as its location and number of employees. Stock data is massive due to its long history.
    e. https://github.com/GaryQian/Omnicorp
2. For our project, we will be using financial stock data with its high/low prices as well as adjusted prices to populate the data. Additional data includes the volume of the stock on each day as well as split and dividend information. The data comes from various files some with the prices and others with information related to the company of that stock. With this information, we can query for specific stocks meeting a certain price range or stock values within a date range. Certain calculations can also be queried. Our system allows full access to all stock data dating back almost a century. Due to the large domain, came up with a fragmentation system to greatly improve query speeds.
3. Source of Data
    a. https://www.quandl.com/data/EOD-End-of-Day-US-Stock-Prices
    b. The dataset was provided through .csv files which we parsed and extracted the financial data through a python script. The data was fragmented across multiple tables with each row storing a shard of the data for the stock on a given day.
    c. Metadata of companies was manually curated for a subset of the companies to provide supplementary info regarding location, CEO, employee count and more.
4. Running the code:
    a. To generate the SQL, the user can run the parse.py which will take an argument which will populate the database up to what the user specifies. For example, to prevent exceeding the storage limit, we test with 1,000,000 data inserts. Make sure to download the data, which is 1.7 GB. Note: Since we dealt with big data, the data insertion and preprocessing steps can take significant amounts of time. Our shard of 1,000,000 csv rows took 6hrs to insert using the mysql command line. This can be improved if there is no buffer to print to.
    b. Start the apache server

    c. Open up Home.html and enter the values to the appropriate field
5. We specialize in data-mining and extraction of real data from online sources.The financial data is stock data over the course of all time for every stock listed on the NASDAQ and NYSE. This dataset is massive which requires an efficient system to parse and load into the database within a reasonable time limit. Our queries allow penetration to extract highly specific information rapidly. To make these queries efficient, we fragement the data across multiple tables to reduce tuple size, disk IO, memory usage, and increase efficiency
6. Selling Points
    a. Full access to vast amounts of data
    b. Vertical Fragmentation across multiple tables to reduce tuple size which means faster joins and lower memory usage.
    c. Intermediate access to SQL insertion
    d. Rapid updatability - able to add new data on easily and will stay up to date with stock trends by simply appending to current dataset.
7. Limitations
    a. Due to fragmenting the data, the database requires more storage for storing all the keys redundantly. This adds extra storage overhead as well as requiring multiple insertions of the same data into the database.
8. DB Tables
    a.
```
drop table Prices;
create table Prices (
    tick     VARCHAR(5),
    date     VARCHAR(10),
    open     FLOAT,
    close    FLOAT,
    high     FLOAT,
    low      FLOAT
);

drop table Volume;
create table Volume (
    tick       VARCHAR(5),
    date       VARCHAR(10),
    volume     FLOAT,
    adjvolume  FLOAT
);

drop table Misc;
create table Misc (
    tick       VARCHAR(5),
    date       VARCHAR(10),
    divi       FLOAT,
    splitratio FLOAT
);

drop table AdjPrices;
create table AdjPrices (
      tick     VARCHAR(5),
      date     VARCHAR(10),
      aopen    FLOAT,
      aclose   FLOAT,
      ahigh    FLOAT,
      alow     FLOAT
);

drop table Company;
create table Company (
      tick       VARCHAR(5),
      name       VARCHAR(25),
      hqkey      VARCHAR(10),
      employees  INTEGER,
      CEO        VARCHAR(20),
      founddate  VARCHAR(10)
);

drop table Location;
create table Location (
      hqkey      VARCHAR(10),
      city     VARCHAR(15),
      state    VARCHAR(10),
      country  VARCHAR(10)
);
```

9.
Front Page:



SHOW ALL TICKER INFORMATION

TICKER: submit

FIND A TICKER ON A SPECIFIC DATE

TICKER: DATE: submit

FIND A TICKER ON A SPECIFIC DATE RANGE

TICKER: STARTDATE: ENDDATE: submit

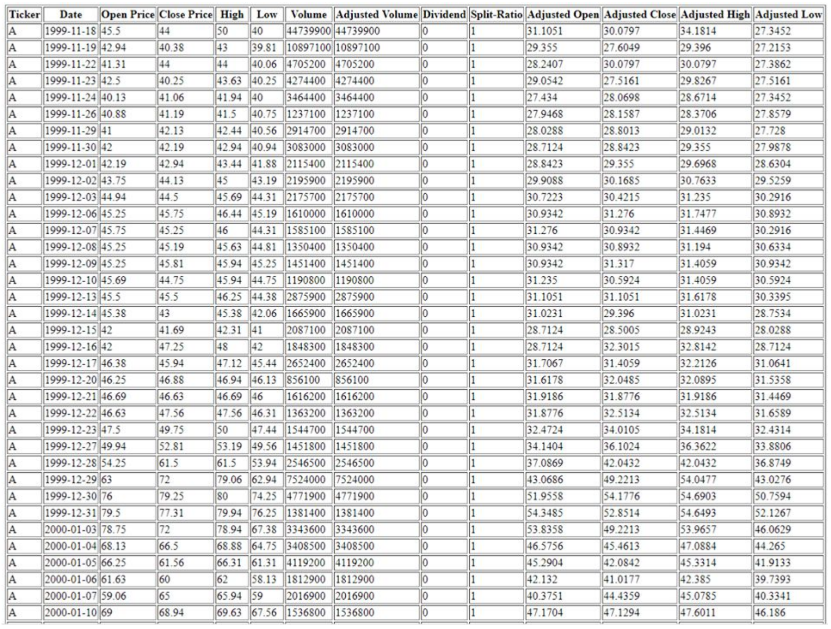
FIND TICKERS WITH GREATER OR EQUAL CLOSING PRICES

CLOSINGPRICE: submit

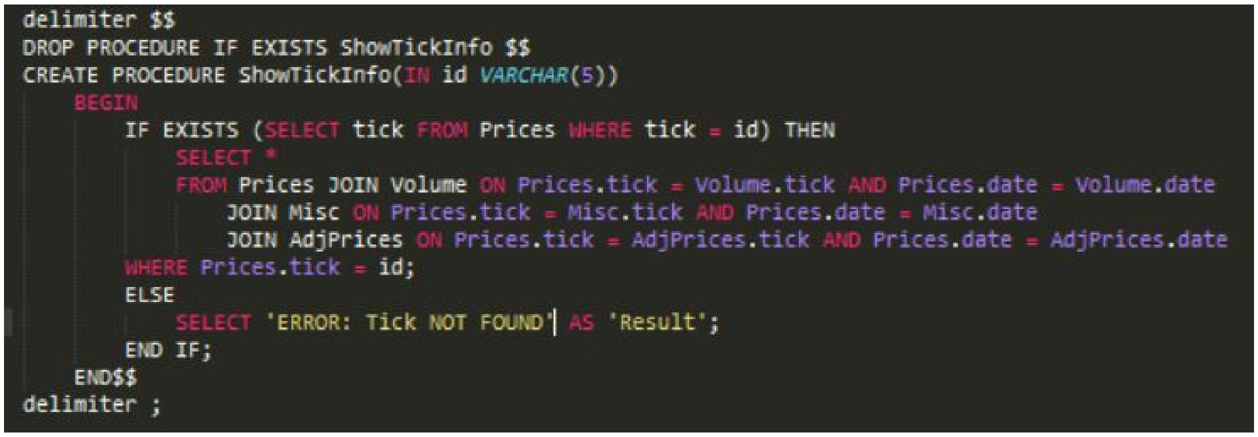
FIND TICKERS WITH LESSER OR EQUAL CLOSING PRICES

Outputs:

a. All Ticker

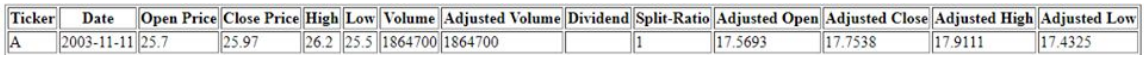


Ticker	Date	Open Price	Close Price	High	Low	Volume	Adjusted Volume	Dividend	Split-Ratio	Adjusted Open	Adjusted Close	Adjusted High	Adjusted Low
A	1999-11-18	45.5	44	50	40	44739900	44739900	0	1	31.1051	30.0797	34.1814	27.3452
A	1999-11-19	42.94	40.38	43	39.81	10897100	10897100	0	1	29.355	27.6049	29.396	27.2153
A	1999-11-22	41.31	44	44	40.06	4705200	4705200	0	1	28.2407	30.0797	30.0797	27.3862
A	1999-11-23	42.5	40.25	43.63	40.25	4274400	4274400	0	1	29.0542	27.5161	29.8267	27.5161
A	1999-11-24	40.13	41.06	41.94	40	3464400	3464400	0	1	27.434	28.0698	28.6714	27.3452
A	1999-11-26	40.88	41.19	41.5	40.75	1237100	1237100	0	1	27.9468	28.1587	28.3706	27.8579
A	1999-11-29	41	42.13	42.44	40.56	2914700	2914700	0	1	28.0288	28.8013	29.0132	27.728
A	1999-11-30	42	42.19	42.94	40.94	3083000	3083000	0	1	28.7124	28.8423	29.355	27.9878
A	1999-12-01	42.19	42.94	43.44	41.88	2115400	2115400	0	1	28.8423	29.355	29.6968	28.6304
A	1999-12-02	43.75	44.13	45	43.19	2195900	2195900	0	1	29.9088	30.1685	30.7633	29.5259
A	1999-12-03	44.94	44.5	45.69	44.31	2175700	2175700	0	1	30.7223	30.4215	31.235	30.2916
A	1999-12-06	45.25	45.75	46.44	45.19	1610000	1610000	0	1	30.9342	31.276	31.7477	30.8932
A	1999-12-07	45.75	45.25	46	44.31	1585100	1585100	0	1	31.276	30.9342	31.4469	30.2916
A	1999-12-08	45.25	45.19	45.63	44.81	1350400	1350400	0	1	30.9342	30.8932	31.194	30.6334
A	1999-12-09	45.25	45.81	45.94	45.25	1451400	1451400	0	1	30.9342	31.317	31.4059	30.9342
A	1999-12-10	45.69	44.75	45.94	44.75	1190800	1190800	0	1	31.235	30.5924	31.4059	30.5924
A	1999-12-13	45.5	45.5	46.25	44.38	2875900	2875900	0	1	31.1051	31.1051	31.6178	30.3395
A	1999-12-14	45.38	43	45.38	42.06	1665900	1665900	0	1	31.0231	29.396	31.0231	28.7534
A	1999-12-15	42	41.69	42.31	41	2087100	2087100	0	1	28.7124	28.5005	28.9243	28.0288
A	1999-12-16	42	47.25	48	42	1848300	1848300	0	1	28.7124	32.3015	32.8142	28.7124
A	1999-12-17	46.38	45.94	47.12	45.44	2652400	2652400	0	1	31.7067	31.4059	32.2126	31.0641
A	1999-12-20	46.25	46.88	46.94	46.13	856100	856100	0	1	31.6178	32.0485	32.0895	31.5358
A	1999-12-21	46.69	46.63	46.69	46	1616200	1616200	0	1	31.9186	31.8776	31.9186	31.4469
A	1999-12-22	46.63	47.56	47.56	46.31	1363200	1363200	0	1	31.8776	32.5134	32.5134	31.6589
A	1999-12-23	47.5	49.75	50	47.44	1544700	1544700	0	1	32.4724	34.0105	34.1814	32.4314
A	1999-12-27	49.94	52.81	53.19	49.56	1451800	1451800	0	1	34.1404	36.1024	36.3622	33.8806
A	1999-12-28	54.25	61.5	61.5	53.94	2546500	2546500	0	1	37.0869	42.0432	42.0432	36.8749
A	1999-12-29	63	72	79.06	62.94	7524000	7524000	0	1	43.0686	49.2213	54.0477	43.0276
A	1999-12-30	76	79.25	80	74.25	4771900	4771900	0	1	51.9558	54.1776	54.6903	50.7594
A	1999-12-31	79.5	77.31	79.94	76.25	1381400	1381400	0	1	54.3485	52.8514	54.6493	52.1267
A	2000-01-03	78.75	72	78.94	67.38	3343600	3343600	0	1	53.8358	49.2213	53.9657	46.0629
A	2000-01-04	68.13	66.5	68.88	64.75	3408500	3408500	0	1	46.5756	45.4613	47.0884	44.265
A	2000-01-05	66.25	61.56	66.31	61.31	4119200	4119200	0	1	45.2904	42.0842	45.3314	41.9133
A	2000-01-06	61.63	60	62	58.13	1812900	1812900	0	1	42.132	41.0177	42.385	39.7393
A	2000-01-07	59.06	65	65.94	59	2016900	2016900	0	1	40.3751	44.4359	45.0785	40.3341
A	2000-01-10	69	68.94	69.63	67.56	1536800	1536800	0	1	47.1704	47.1294	47.6011	46.186

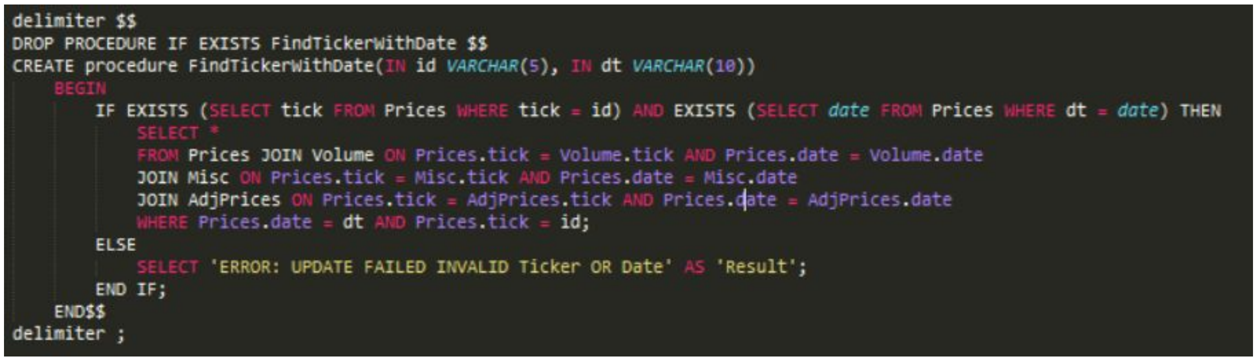


```
delimiter $$
DROP PROCEDURE IF EXISTS ShowTickInfo $$
CREATE PROCEDURE ShowTickInfo(IN id VARCHAR(5))
    BEGIN
        IF EXISTS (SELECT tick FROM Prices WHERE tick = id) THEN
            SELECT *
            FROM Prices JOIN Volume ON Prices.tick = Volume.tick AND Prices.date = Volume.date
                JOIN Misc ON Prices.tick = Misc.tick AND Prices.date = Misc.date
                JOIN AdjPrices ON Prices.tick = AdjPrices.tick AND Prices.date = AdjPrices.date
        WHERE Prices.tick = id;
        ELSE
            SELECT 'ERROR: Tick NOT FOUND' AS 'Result';
        END IF;
    END$$
delimiter ;
```

b. Ticker With Date

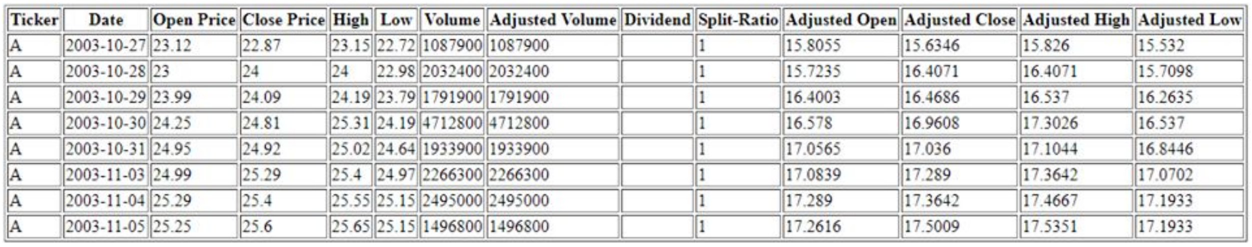


Ticker	Date	Open Price	Close Price	High	Low	Volume	Adjusted Volume	Dividend	Split-Ratio	Adjusted Open	Adjusted Close	Adjusted High	Adjusted Low
A	2003-11-11	25.7	25.97	26.2	25.5	1864700	1864700		1	17.5693	17.7538	17.9111	17.4325

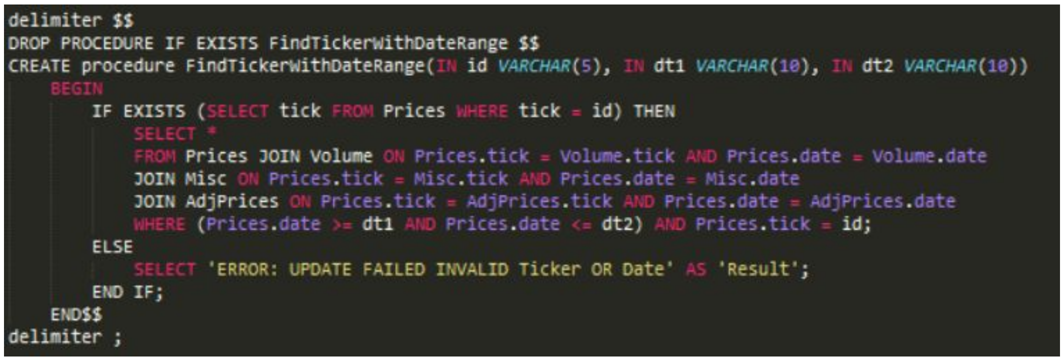


```
delimiter $$
DROP PROCEDURE IF EXISTS FindTickerWithDate $$
CREATE procedure FindTickerWithDate(IN id VARCHAR(5), IN dt VARCHAR(10))
    BEGIN
        IF EXISTS (SELECT tick FROM Prices WHERE tick = id) AND EXISTS (SELECT date FROM Prices WHERE dt = date) THEN
            SELECT *
            FROM Prices JOIN Volume ON Prices.tick = Volume.tick AND Prices.date = Volume.date
            JOIN Misc ON Prices.tick = Misc.tick AND Prices.date = Misc.date
            JOIN AdjPrices ON Prices.tick = AdjPrices.tick AND Prices.date = AdjPrices.date
            WHERE Prices.date = dt AND Prices.tick = id;
        ELSE
            SELECT 'ERROR: UPDATE FAILED INVALID Ticker OR Date' AS 'Result';
        END IF;
    END$$
delimiter ;
```

c. Ticker With Date Range

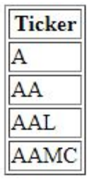


Ticker	Date	Open Price	Close Price	High	Low	Volume	Adjusted Volume	Dividend	Split-Ratio	Adjusted Open	Adjusted Close	Adjusted High	Adjusted Low
A	2003-10-27	23.12	22.87	23.15	22.72	1087900	1087900		1	15.8055	15.6346	15.826	15.532
A	2003-10-28	23	24	24	22.98	2032400	2032400		1	15.7235	16.4071	16.4071	15.7098
A	2003-10-29	23.99	24.09	24.19	23.79	1791900	1791900		1	16.4003	16.4686	16.537	16.2635
A	2003-10-30	24.25	24.81	25.31	24.19	4712800	4712800		1	16.578	16.9608	17.3026	16.537
A	2003-10-31	24.95	24.92	25.02	24.64	1933900	1933900		1	17.0565	17.036	17.1044	16.8446
A	2003-11-03	24.99	25.29	25.4	24.97	2266300	2266300		1	17.0839	17.289	17.3642	17.0702
A	2003-11-04	25.29	25.4	25.55	25.15	2495000	2495000		1	17.289	17.3642	17.4667	17.1933
A	2003-11-05	25.25	25.6	25.65	25.15	1496800	1496800		1	17.2616	17.5009	17.5351	17.1933

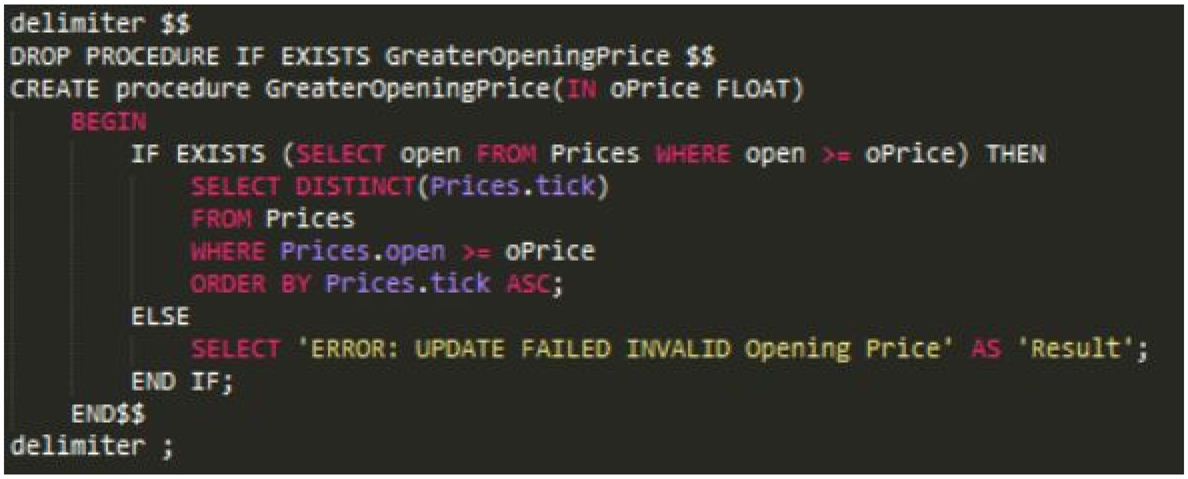


```
delimiter $$
DROP PROCEDURE IF EXISTS FindTickerWithDateRange $$
CREATE procedure FindTickerWithDateRange(IN id VARCHAR(5), IN dt1 VARCHAR(10), IN dt2 VARCHAR(10))
    BEGIN
        IF EXISTS (SELECT tick FROM Prices WHERE tick = id) THEN
            SELECT *
            FROM Prices JOIN Volume ON Prices.tick = Volume.tick AND Prices.date = Volume.date
            JOIN Misc ON Prices.tick = Misc.tick AND Prices.date = Misc.date
            JOIN AdjPrices ON Prices.tick = AdjPrices.tick AND Prices.date = AdjPrices.date
            WHERE (Prices.date >= dt1 AND Prices.date <= dt2) AND Prices.tick = id;
        ELSE
            SELECT 'ERROR: UPDATE FAILED INVALID Ticker OR Date' AS 'Result';
        END IF;
    END$$
delimiter ;
```

d. Close Prices greater than $45.00

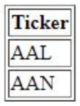


Ticker
A
AA
AAL
AAMC

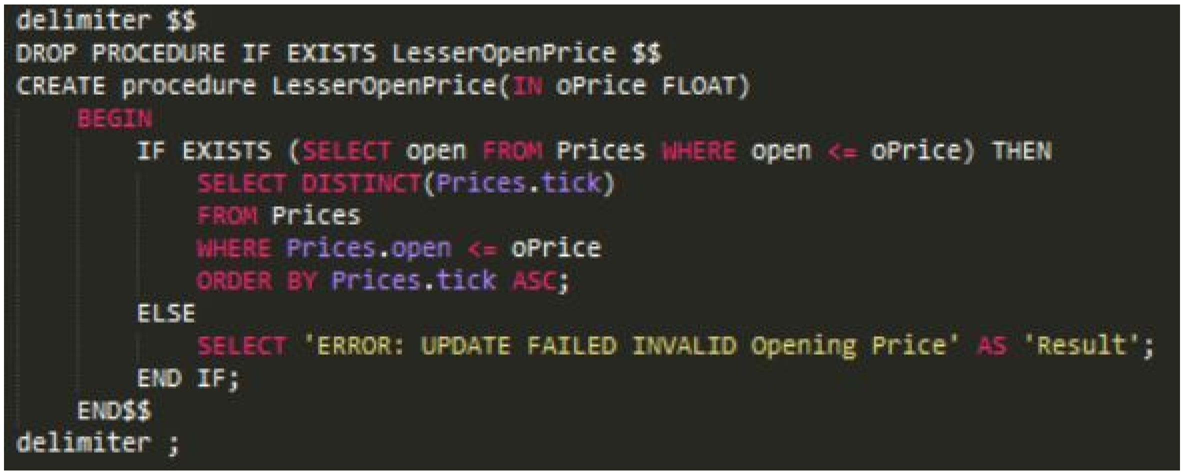


```
delimiter $$
DROP PROCEDURE IF EXISTS GreaterOpeningPrice $$
CREATE procedure GreaterOpeningPrice(IN oPrice FLOAT)
    BEGIN
        IF EXISTS (SELECT open FROM Prices WHERE open >= oPrice) THEN
            SELECT DISTINCT(Prices.tick)
            FROM Prices
            WHERE Prices.open >= oPrice
            ORDER BY Prices.tick ASC;
        ELSE
            SELECT 'ERROR: UPDATE FAILED INVALID Opening Price' AS 'Result';
        END IF;
    END$$
delimiter ;
```

e. Close Prices lesser than $10.00

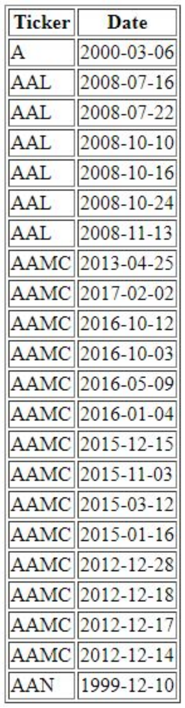


Ticker
AAL
AAN

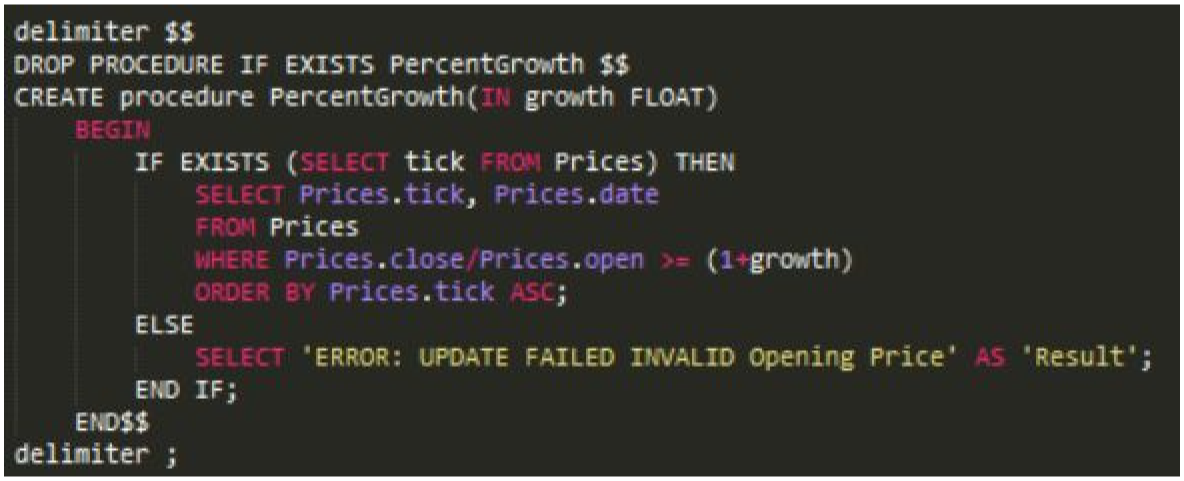


```
delimiter $$
DROP PROCEDURE IF EXISTS LesserOpenPrice $$
CREATE procedure LesserOpenPrice(IN oPrice FLOAT)
    BEGIN
        IF EXISTS (SELECT open FROM Prices WHERE open <= oPrice) THEN
            SELECT DISTINCT(Prices.tick)
            FROM Prices
            WHERE Prices.open <= oPrice
            ORDER BY Prices.tick ASC;
        ELSE
            SELECT 'ERROR: UPDATE FAILED INVALID Opening Price' AS 'Result';
        END IF;
    END$$
delimiter ;
```

f. Growth of .2 or 20%

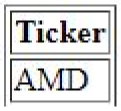


Ticker	Date
A	2000-03-06
AAL	2008-07-16
AAL	2008-07-22
AAL	2008-10-10
AAL	2008-10-16
AAL	2008-10-24
AAL	2008-11-13
AAMC	2013-04-25
AAMC	2017-02-02
AAMC	2016-10-12
AAMC	2016-10-03
AAMC	2016-05-09
AAMC	2016-01-04
AAMC	2015-12-15
AAMC	2015-11-03
AAMC	2015-03-12
AAMC	2015-01-16
AAMC	2012-12-28
AAMC	2012-12-18
AAMC	2012-12-17
AAMC	2012-12-14
AAN	1999-12-10

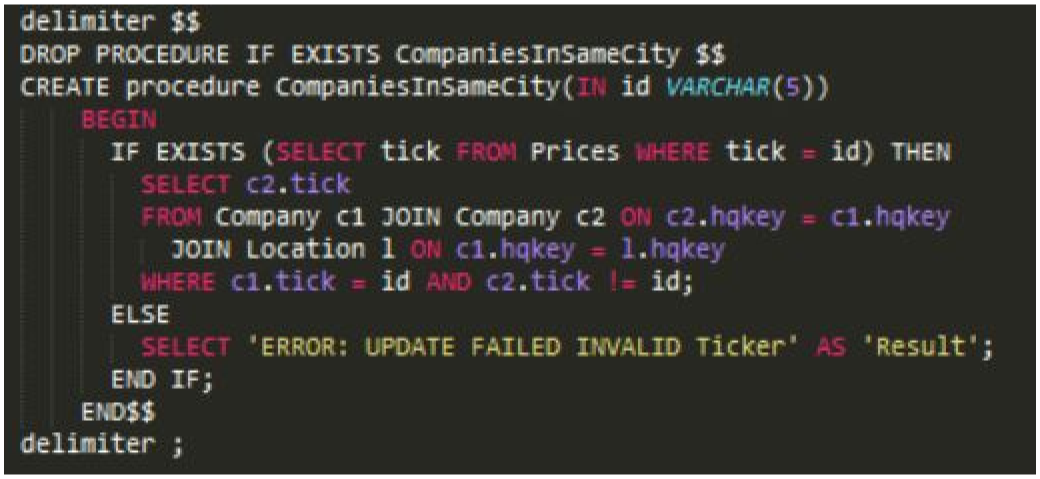


```
delimiter $$
DROP PROCEDURE IF EXISTS PercentGrowth $$
CREATE procedure PercentGrowth(IN growth FLOAT)
    BEGIN
        IF EXISTS (SELECT tick FROM Prices) THEN
            SELECT Prices.tick, Prices.date
            FROM Prices
            WHERE Prices.close/Prices.open >= (1+growth)
            ORDER BY Prices.tick ASC;
        ELSE
            SELECT 'ERROR: UPDATE FAILED INVALID Opening Price' AS 'Result';
        END IF;
    END$$
delimiter ;
```

g. Companies in the same city with input “A”

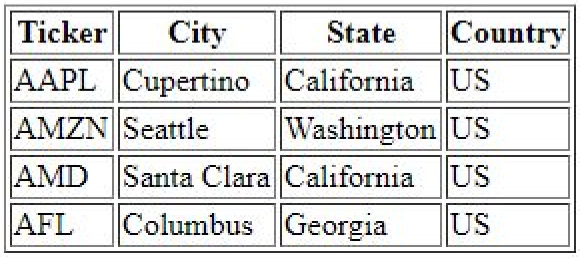


Ticker
AMD

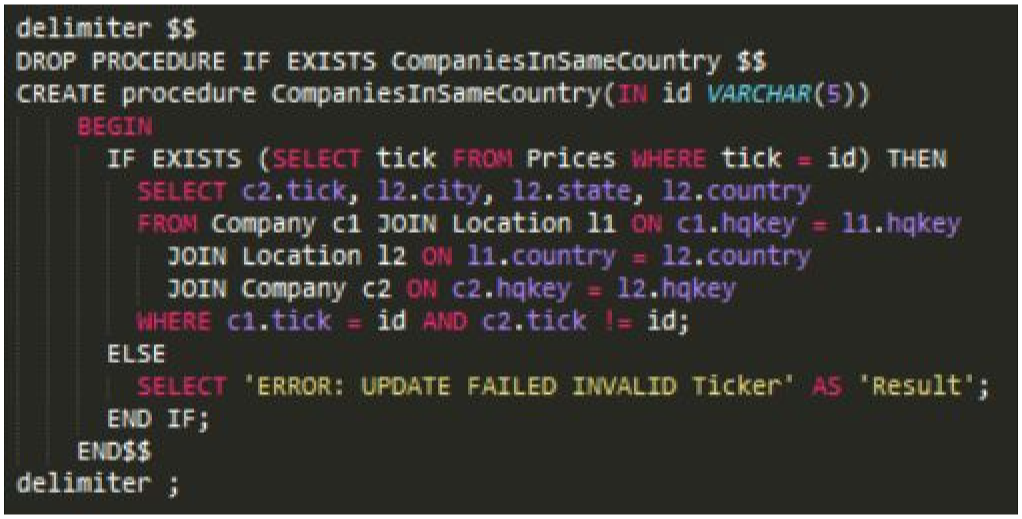


```sql
delimiter $$
DROP PROCEDURE IF EXISTS CompaniesInSameCity $$
CREATE procedure CompaniesInSameCity(IN id VARCHAR(5))
    BEGIN
      IF EXISTS (SELECT tick FROM Prices WHERE tick = id) THEN
        SELECT c2.tick
        FROM Company c1 JOIN Company c2 ON c2.hqkey = c1.hqkey
          JOIN Location l ON c1.hqkey = l.hqkey
        WHERE c1.tick = id AND c2.tick != id;
      ELSE
        SELECT 'ERROR: UPDATE FAILED INVALID Ticker' AS 'Result';
      END IF;
    END$$
delimiter ;
```

h. Companies in same country with input “A”

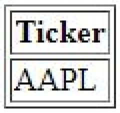


Ticker	City	State	Country
AAPL	Cupertino	California	US
AMZN	Seattle	Washington	US
AMD	Santa Clara	California	US
AFL	Columbus	Georgia	US

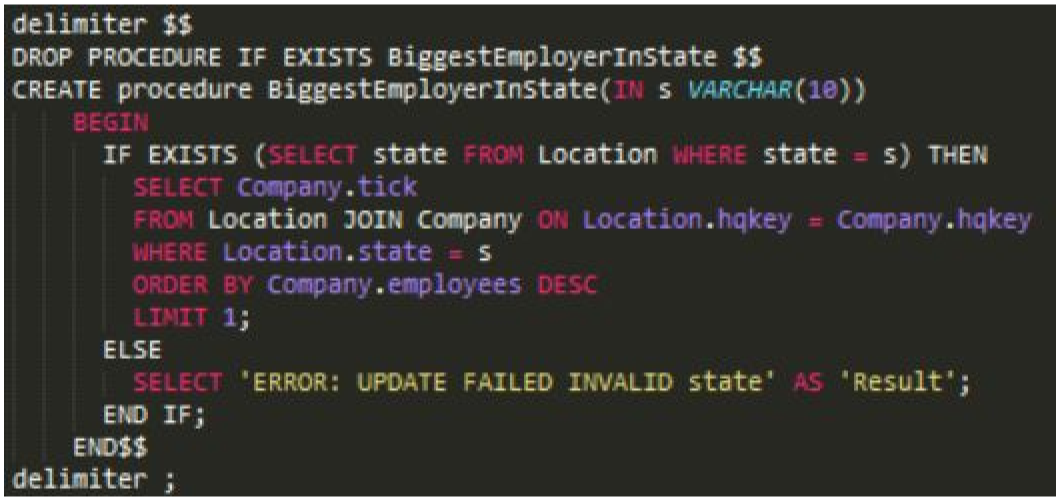


```sql
delimiter $$
DROP PROCEDURE IF EXISTS CompaniesInSameCountry $$
CREATE procedure CompaniesInSameCountry(IN id VARCHAR(5))
    BEGIN
      IF EXISTS (SELECT tick FROM Prices WHERE tick = id) THEN
        SELECT c2.tick, l2.city, l2.state, l2.country
        FROM Company c1 JOIN Location l1 ON c1.hqkey = l1.hqkey
          JOIN Location l2 ON l1.country = l2.country
          JOIN Company c2 ON c2.hqkey = l2.hqkey
        WHERE c1.tick = id AND c2.tick != id;
      ELSE
        SELECT 'ERROR: UPDATE FAILED INVALID Ticker' AS 'Result';
      END IF;
    END$$
delimiter ;
```

i. Biggest Employer in the State of California



Ticker
AAPL



```
delimiter $$
DROP PROCEDURE IF EXISTS BiggestEmployerInState $$
CREATE procedure BiggestEmployerInState(IN s VARCHAR(10))
    BEGIN
      IF EXISTS (SELECT state FROM Location WHERE state = s) THEN
        SELECT Company.tick
        FROM Location JOIN Company ON Location.hqkey = Company.hqkey
        WHERE Location.state = s
        ORDER BY Company.employees DESC
        LIMIT 1;
      ELSE
        SELECT 'ERROR: UPDATE FAILED INVALID state' AS 'Result';
      END IF;
    END$$
delimiter ;
```