SQL Workshop Week 2

UC Davis DSI

2018

1 Review

1. Which tickers have a beta of 3 or more?

Solution:

```
SELECT *
FROM financial_ratios
WHERE beta >= 3
;
```

2. How many days in 2014-2015 did Amazon (AMZN) shares close above 350?

Solution:

```
SELECT COUNT(*)
FROM daily_share_prices
WHERE
        ticker = 'AMZN'
        AND date BETWEEN '2014-01-01' AND '2015-01-01'
        AND close > 350
;
```

3. Are any values missing from the state populations table?

Solution:

```
SELECT *
FROM state_populations
WHERE
        popnEstimate2017 IS NULL
        OR popn2010 IS NULL
;
```

2 Aggregation

Aggregation Functions

1. What's the total assets of all companies in the company info table?

Solution:

```
SELECT SUM(asset)
FROM company_info
;
```

2. How many companies are in the "Energy" sector?

Solution:

```
SELECT COUNT(*)
FROM company_info
WHERE sector = 'Energy'
;
```

3. Which ticker has the most assets? Which has the least? Answer this question without using `ORDER BY`.

Solution:

```
-- Min assets
SELECT ticker, min(asset)
FROM company_info
;

-- Max assets
SELECT ticker, max(asset)
FROM company_info
;
```

4. What happens if you aggregate two different columns in the same query? Give an example and explain the result. What happens if you also include columns that aren't aggregated?

Solution: Consider the query

```
SELECT ticker, min(asset), max(asset)
FROM company_info
;
```

The result of this query is

```
ticker  min(asset)  max(asset)
FNMA    74100.0     3287968000000.0
```

The company FNMA has the most assets and FTI has the least. The query returns the correct min and max values for the asset column, but only the ticker FNMA.

By trying several different queries, it looks like rows for columns that aren't aggregated are selected based on the last aggregation in the query. In the example query, that's `max(asset)`, so FNMA is selected rather than FTI.

Grouping

5. How many Fortune 500 companies are in each state?

Solution:

```
SELECT state, COUNT(*) AS num_companies
FROM company_locations
GROUP BY state
ORDER BY num_companies DESC
;
```

6. According to the company locations table, how many companies went down in Fortune 500 rankings from 2014 to 2015? How many went up? Try to get all of this information in one query.

Solution:

```
SELECT trend, COUNT(*)
FROM company_locations
GROUP BY trend
;
```

7. What is the average opening price for each stock in the daily share prices table? Sort the result from lowest to highest average price.

Solution:

```
SELECT ticker, AVG(open) AS avg_price
FROM daily_share_prices
GROUP BY ticker
ORDER BY avg_price
;
```

8. Which companies in the daily share prices table had the 3 highest average closing prices for 2014? Hint: use `STRFTIME('%Y', date)` to extract the year.

Solution:

```
SELECT ticker, AVG(close) AS avg_close
FROM daily_share_prices
WHERE STRFTIME('%Y', date) = '2014'
GROUP BY ticker
ORDER BY avg_close DESC
LIMIT 3
;
```

Filter Rows After Aggregation

9. Which states have between 5 and 10 companies?

Solution:

```
SELECT state, COUNT(*) AS freq
FROM company_locations
GROUP BY state
HAVING freq BETWEEN 5 AND 10
ORDER BY freq
;
```

10. Which stocks have negative average daily increase (close − open)) over the 5-year period?

Solution:

```
SELECT ticker, AVG(close - open) AS avg_increase
FROM daily_share_prices
GROUP BY ticker
HAVING avg_increase < 0
;
```

11. Which year and month combinations (e.g., April 2017, June 2016) have less than 8000 share prices? Hint: use `STRFTIME('%Y-%m', date)` to extract the year and month.

Solution:

```
SELECT
        STRFTIME('%Y-%m', date) AS year_month,
        COUNT(*) AS freq
FROM daily_share_prices
GROUP BY year_month
HAVING freq < 8000
;
```

12. Which industries have a negative average earning per share?

Solution:

```
SELECT sector, AVG(earning_per_share) AS avg_earning
FROM company_info
GROUP BY industry
HAVING avg_earning <= 0
ORDER BY avg_earning
;
```

3 Combining Data

From Wikipedia:

> The Standard Industrial Classification (SIC) is a system for classifying industries by a four-digit code. Established in the United States in 1937, it is used by government agencies to classify industry areas.

1. How many different SIC codes appear in the `company_info` table?

Solution:

```
SELECT COUNT(DISTINCT(sic_code))
FROM company_info
;
```

2. Which SIC codes appear in `company_name` but have no description in the `sic` table?

Solution:

```
SELECT DISTINCT(sic_code)
FROM company_info
EXCEPT
SELECT SIC
FROM sic
;
```

We can also use a subquery

```
SELECT DISTINCT(sic_code)
FROM company_info
WHERE NOT sic_code IN (SELECT SIC FROM sic);
```

Single Joins

3. Select the ticker, company name, and SIC code with corresponding description for companies that have SIC codes.

Solution:

```
SELECT ticker, company_name, sic_code, Description
FROM company_info
INNER JOIN sic
ON company_info.sic_code = sic.SIC
;
```

4. Select the ticker, company name, and SIC code with corresponding description (possibly NULL) for all companies in the `company_info` table.

Solution:

```
SELECT ticker, company_name, sic_code, Description
FROM company_info
LEFT JOIN sic
ON company_info.sic_code = sic.SIC
;
```

5. Which 4 digit SIC codes have the most companies?

Solution:

```
SELECT sic_code, Description, count(*) as cnt
FROM company_info
LEFT JOIN sic
ON company_info . sic_code = sic . SIC
GROUP BY sic_code
ORDER BY cnt DESC
;
```

Sub-Queries

6. Focusing only on 2014 and the `daily_share_prices` table, find the names of the companies which had any closing price that exceed the average closing price of AAPL in 2014.

Solution: This is a two-step operation. We need to first compute the average closing price for Apple stock for 2014. We do this code we have already practiced:

```
SELECT AVG(close) FROM daily_share_prices
WHERE ticker = 'AAPL' AND date BETWEEN '2014-01-01' AND '2014-12-31';
```

We could read the average price we get and use it to filter rows in a new query by typing it directly. But that is not ideal. Instead, we'll compute this result and use it directly in our larger query.

```
SELECT ticker FROM daily_share_prices
WHERE
   date BETWEEN '2014-01-01' AND '2014-12-31'
 AND
   close > (SELECT AVG(close) FROM daily_share_prices
             WHERE ticker = 'AAPL' AND date BETWEEN '2014-01-01' AND '2014-12-31');
```

This gives back many rows, the first of which are AAPL. What we have is a row for each day that a close price exceeded the average. We want just the names of the companies, so not the same name repeated as many times as that ticker exceeded the AAPL average. We do this by adding DISTINCT to our query to make the rows unique:

```
SELECT DISTINCT ticker FROM daily_share_prices
WHERE
   date BETWEEN '2014-01-01' AND '2014-12-31'
 AND
   close > (SELECT AVG(close) FROM daily_share_prices
             WHERE ticker = 'AAPL' AND date BETWEEN '2014-01-01' AND '2014-12-31');
```

Bonus Question Count the number of days these stocks had a close price above this average.

Solution: For this, we use a GROUP BY and a call to COUNT() to get the result we want, discarding the DISTINCT we added above for a different purpose:

```
SELECT ticker, COUNT(ticker) FROM daily_share_prices
WHERE
   date BETWEEN '2014-01-01' AND '2014-12-31'
 AND
   close > (SELECT AVG(close) FROM daily_share_prices
              WHERE ticker = 'AAPL' AND date BETWEEN '2014-01-01' AND '2014-12-31')
GROUP BY ticker;
```

Multiple Joins

7. Write a query that produces a table with columns for state, SIC code, SIC description, and count of companies located in that state with that SIC code.

Solution:

```
SELECT l.state,  i.sic_code, s.Description, count(*) as cnt
FROM company_info as i
LEFT JOIN sic as s
ON i.sic_code = s.SIC
INNER JOIN company_locations as l
ON l.ticker = i.ticker
GROUP BY i.sic_code, l.state
ORDER BY l.state ASC, cnt DESC
;
```

We'll break this into 2 steps. We want the state and ticker and SIC ode all in one table for each company. If a compay doesn't have a SIC code, we want to drop. So this an INNER JOIN.

```
SELECT l.ticker, l.state, r.sic_code
FROM company_locations AS l
INNER JOIN company_info AS r
ON l.ticker = r.ticker;
```

Having done this, we can do the next step. This merges the description of the SIC code

Now with this, we want to count by

```
SELECT state, sic_code, COUNT(ticker) AS cnt, B.description
FROM
   (SELECT l.ticker, l.state, r.sic_code
        FROM company_locations AS l
        INNER JOIN company_info AS r
        ON l.ticker = r.ticker )
      AS A
LEFT JOIN sic as B
ON A.sic_code = B.SIC
GROUP BY state, sic_code
ORDER BY state ASC, cnt DESC;
```

8. Modify the query above to only return rows with a count of three or more companies.

Solution:

```
SELECT l.state,  i.sic_code, s.Description, count(*) as cnt
FROM company_info as i
LEFT JOIN sic as s
ON i.sic_code = s.SIC
INNER JOIN company_locations as l
ON l.ticker = i.ticker
GROUP BY i.sic_code, l.state
HAVING cnt >= 3
ORDER BY l.state ASC, cnt DESC
;
```

9. Which ticker symbols appear in both `company_locations` and `daily_share_prices`?

Solution: We can use a subquery again.

```
SELECT ticker
FROM company_locations
WHERE ticker IN (SELECT ticker FROM daily_share_prices);
```

Alternatively, like EXCEPT, we can use INTERSECT

```
SELECT ticker
FROM company_locations
INTERSECT
SELECT ticker
FROM daily_share_prices
;
```