SQL Workshop : Day 1

2018年5月4日

2pm - 4:30pm start
30 setup
2:20 30 SELECT / LIMIT / DISTINCT
ORDER BY / functions
3:20 10 break
30 work
30 WHERE / AND / OR / LIKE
NOT / BETWEEN / IN
IS NULL
30 work
end

Get browser from sqlitebrowser.org
Get database from $\boxed{\text{anson.ucdavis.edu/~clarkf/sql}}$

Filtering Data

- Only select rows that satisfy a condition

  Coca-Cola

- $\boxed{\text{WHERE}}$ + condition

```
SELECT * FROM daily_share_prices WHERE ticker = 'KO';
```

one or two =

single or double quotes

- This works even if we do not select columns in condition

```
SELECT open FROM daily_share_prices WHERE ticker = 'KO';
```

- Use $\boxed{\text{AND}}$ and $\boxed{\text{OR}}$ to combine

```
SELECT * FROM daily_share_prices WHERE ticker = 'KO'
AND open >= 40;

SELECT * FROM daily_share_prices WHERE ticker = 'KO'
OR ticker = 'PEP';
```

- $\boxed{\text{IN}}$ for a list of values

```
SELECT * FROM daily_share_prices WHERE ticker IN ('KO', 'PEP');
```

- Order of operations matters

```
SELECT * FROM daily_share_prices
WHERE (ticker = 'KO' OR ticker = 'PEP') AND open > 40
ORDER BY open;
```

  Use parens to fix.

- Also a shortcut for ranges, $\boxed{\text{BETWEEN}}$

```
SELECT * FROM daily_share_prices
WHERE ticker = 'KO' AND open BETWEEN 40 AND 42;
```

Dates

- SQLite stores $\boxed{\text{dates}}$ as text or numbers

```
SELECT * FROM daily_share_prices
WHERE ticker = 'KO' AND date >= '2018-01-01';
```

  Also work with BETWEEN and ORDER BY

Text

- $\boxed{\text{LIKE}}$ to match patterns

```
SELECT * FROM daily_share_prices
WHERE ticker LIKE 'K%';
```

%	0 or more
_	1

```
SELECT * FROM daily_share_prices
WHERE ticker LIKE 'K_';
```

Missing Values

- Use $\boxed{\text{IS NULL}}$ to find missing values
- NULL is not 0, false, etc...
  It means "we don't know"

```
SELECT * FROM company_info
WHERE web_page = NULL;          ← no results
```

```
SELECT * FROM company_info
WHERE web_page IS NULL;
```

Functions

- Use $\boxed{\text{operators}}$ to combine columns

```
SELECT share_price * dividend_yield FROM financial_ratios;
```

- Use $\boxed{\text{AS}}$ to rename columns computed:

```
SELECT share_price * dividend_yield AS dividend
FROM financial_ratio
ORDER BY dividend DESC;
```

- Many more functions on cheatsheet

SQL Workshop : Day 2

2018年 5月 11日

2pm - 4:30pm

	start	
2:10	15	review
2:25	45	Aggregation
3:10	10	break
3:20	45	Joins
	end	

URL : anson.ucdavis.edu/~nulle/sql Updated!

Quick Review

- Last time we saw

  SELECT, LIMIT, ORDER BY, DISTINCT,
  COUNT, WHERE, AND, OR, NOT,
  BETWEEN, IN, IS NULL, LIKE

- For example, we can build up the query

```
SELECT * FROM daily_share_prices
WHERE ticker IN ('KO', 'PEP')
AND date BETWEEN '2018-01-01' AND '2018-01-07';
```

Aggregation

- We saw the COUNT() function last time.
  It collapses or aggregates many rows into one :

```
SELECT COUNT(*) FROM state_populations;
```

- There are other aggregation functions

  MIN() MAX() SUM() AVG()

```
SELECT AVG(open) FROM daily_share_prices;
```

Use fang_prices!

```
SELECT MAX ~~[illegible]~~ (open) ~~[illegible]~~ FROM daily_share_prices
WHERE ticker = 'KO';
```

Compare ORDER BY

- You can include unaggregated columns:

```
SELECT ticker, MAX(volume) FROM daily_share_prices;
```

  But sometimes this may not make sense

```
SELECT ticker, AVG(volume) FROM ~~daily_share_prices~~
fang_prices;
```

Grouping

- Aggregates can be computed for groups with GROUP BY

```
SELECT ticker, AVG(close) FROM fang_prices
GROUP BY ticker;
```

  - Again, it may not make sense to include unaggregated columns

Filtering After Aggregation

- The WHERE clause filters rows before aggregation

```
SELECT ticker, AVG(high) FROM fang-prices
WHERE ticker IN ('AMZN', 'NFLX')
GROUP BY ticker;
```

- The HAVING clause filters rows after aggregation

```
  SELECT ticker, AVG(high) AS avg-high FROM fang-prices
-- WHERE ticker IN ('AMZN', 'NFLX')
  GROUP BY ticker
  HAVING avg-high > 1000;
```