Tools & Models for Data Science

SQL Aggregations and Grouping

Risa Myers

Rice University

Aggregations

- Can compute simple statistics using built-in SQL functions
  - `SUM`
  - `AVG`
  - `COUNT`
  - `MAX`
  - `MIN`
  - etc.

? What do all of these aggregates have in common?

Our First Aggregation

RATES (DRINKER, COFFEE, SCORE)

? What is the average coffee rating given by Risa?

Our First Aggregation

RATES (DRINKER, COFFEE, SCORE)

- What is the average coffee rating given by Risa?

```
SELECT AVG (r.SCORE)
FROM RATES r
WHERE r.DRINKER = 'Risa'
```

RATES (DRINKER, COFFEE, SCORE)

? How many coffees has Risa rated?

- Note: RATES does not have a primary key

? What are the reperecussions?

COUNT DISTINCT

RATES (DRINKER, COFFEE, SCORE)

- How many coffees has Risa rated?
- ? Does this work?

```
SELECT COUNT (*)
FROM RATES r
WHERE r.DRINKER = 'Risa'
```

RATES (DRINKER, COFFEE, SCORE)

- How many coffees has Risa rated?
- Does this work?

```
SELECT COUNT (*)
FROM RATES r
WHERE r.DRINKER = 'Risa'
```

- Counts the number of ratings due to Risa.
- ? Count the number of different types of coffee drinks that Risa has rated

RATES (DRINKER, COFFEE, SCORE)

- How many coffees has Risa rated?
- This gives us the actual number rated:

```
SELECT COUNT (DISTINCT r.COFFEE)
FROM RATES r
WHERE r.DRINKER = 'Risa'
```

RATES (DRINKER, COFFEE, SCORE)

- It is often desirable to compute an aggregate at a finer level of granularity.
- ? What is the average rating for each coffee?

RATES (DRINKER, COFFEE, SCORE)

- It is often desirable to compute an aggregate at a finer level of granularity.
- What is the average rating for each coffee?

```
SELECT r.COFFEE, AVG (r.SCORE)
FROM RATES r
GROUP BY r.COFFEE
```

- This first groups the relation into subgroups
- Every tuple in the subgroup has the same value for r.COFFEE
- Then the aggregate runs over each subgroup independently

```
SELECT r.COFFEE, AVG (r.SCORE)
FROM RATES r
GROUP BY r.COFFEE
```

- Example input:

```
('Chris', 'Cold Brew', 1)
('Chris', 'Turkish Coffee', 5)
('Jorge', 'Cold Brew', 1)
('Jorge', 'Chai Latte', 3)
('Risa', 'Cold Brew', 4)
('Risa', 'Cold Brew', 5)
('Risa', 'Espresso', 2)
```

? What is the output?

```
SELECT r.COFFEE, AVG (r.SCORE)
FROM RATES r
GROUP BY r.COFFEE
```

- Example input:

```
('Chris', 'Cold Brew', 1)
('Chris', 'Turkish Coffee', 5)
('Jorge', 'Cold Brew', 1)
('Jorge', 'Chai Latte', 3)
('Risa', 'Cold Brew', 4)
('Risa', 'Cold Brew', 5)
('Risa', 'Espresso', 2)
```

- What is the output?

```
('Turkish Coffee', 5)
('Chai Latte', 3)
('Cold Brew', 2.75)
('Espresso', 2)
```

- Take care with integer arithmetic!

```
SELECT r.COFFEE, AVG (R.SCORE)
FROM RATES r
GROUP BY r.COFFEE
```

- Note: If you have an attribute outside of an aggregate function in an aggregate query
- Example: r.COFFEE here
- Then you must have grouped by that attribute
- Or the query will not compile

? Why?

GROUP BY Conceptually

- Given the following data

DRINKER	COFFEE	SCORE
Risa	Espresso	2
Chris	Cold Brew	1
Chris	Turkish Coffee	5
Risa	Cold Brew	4
Risa	Cold Brew	5

? What is each drinker's average coffee rating?

GROUP BY Conceptually

- Given the following data

DRINKER	COFFEE	SCORE
Risa	Espresso	2
Chris	Cold Brew	1
Chris	Turkish Coffee	5
Risa	Cold Brew	4
Risa	Cold Brew	5

? What is each drinker's average coffee rating?

1 GROUP BY DRINKER

DRINKER	COFFEE	SCORE
Chris	Cold Brew	1
Chris	Turkish Coffee	5
Risa	Espresso	2
Risa	Cold Brew	4
Risa	Cold Brew	5

2 Aggregate

DRINKER	AVGSCORE
Chris	3
Risa	3.67

RATES (DRINKER, COFFEE, SCORE)

? What is the highest rated type of coffee, on average, considering only coffees that have at least 3 ratings?

- From last class:

```
CREATE VIEW COFFEE_AVG_RATING AS
   SELECT r.COFFEE, AVG (r.SCORE) AS AVG_RATING
   FROM RATES r
   GROUP BY r.COFFEE

SELECT a.COFFEE
FROM COFFEE_AVG_RATING a
WHERE a.AVG_RATING = (SELECT MAX(a.AVG_RATING)
                      FROM COFFEE_AVG_RATING a)
```

? How do we check for at least 3 ratings?

RATES (DRINKER, COFFEE, SCORE)

- What is the highest rated type of coffee, on average, considering only coffees that have at least 3 ratings?
- Change COFFEE_AVG_RATING to:

```
CREATE VIEW COFFEE_AVG_RATING AS
    SELECT r.COFFEE, AVG(r.SCORE) AS AVG_RATING
    FROM RATES r
    GROUP BY COFFEE
    HAVING COUNT(*) >= 3
```

HAVING Conceptually

- Given the following data

DRINKER	COFFEE	SCORE
Risa	Espresso	2
Chris	Cold Brew	1
Chris	Turkish Coffee	5
Risa	Cold Brew	4
Risa	Cold Brew	5

? What is the highest rated type of coffee, on average, considering only coffees that have at least 3 ratings?

1 `GROUP BY` COFFEE

DRINKER	COFFEE	SCORE
Chris	Cold Brew	1
Risa	Cold Brew	4
Risa	Cold Brew	5
Chris	Turkish Coffee	5
Risa	Espresso	2

2 Aggregate

COFFEE	AVGSCORE
Cold Brew	3.33
Turkish Coffee	5
Espresso	2

3 `HAVING` COUNT(*) >= 3

DRINKER	AVGSCORE
Cold Brew	3.33

Aggregate Functions & NULL

What about NULL?

- `COUNT`(1) or `COUNT`(*) will count every row
- `COUNT`(<attribute>) will count NON-NULL values
- `AVG, MIN, MAX`, etc. ignore `NULL` values
- `GROUP BY` includes a row for `NULL`

Subquery in `FROM` Revisited

RATES (DRINKER, COFFEE, SCORE)

- Can have a subquery in `FROM` clause, treated as a temporary table
- ? What is the highest rated coffee, on average?

Subquery in FROM Revisited

RATES (DRINKER, COFFEE, SCORE)

- Can have a subquery in FROM clause, treat as a temporary table
- What is the highest rated coffee, on average?

```
SELECT a.COFFEE
FROM (SELECT r.COFFEE, AVG (r.SCORE) AS AVG_RATING
      FROM RATES r
      GROUP BY r.COFFEE) a
WHERE a.AVG_RATING = (SELECT MAX(a.AVG_RATING)
                      FROM (SELECT r.COFFEE, AVG (r.SCORE)
                                AS AVG_RATING
                            FROM RATES r
                            GROUP BY r.COFFEE) a)
```

Subquery in FROM Revisited

RATES (DRINKER, COFFEE, SCORE)

- Note: The code is a lot cleaner with a view!

```
CREATE VIEW COFFEE_AVG_RATING AS
SELECT r.COFFEE, AVG (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.COFFEE

SELECT a.COFFEE
FROM COFFEE_AVG_RATING a
WHERE a.AVG_RATING = (SELECT MAX(a.AVG_RATING)
                      FROM COFFEE_AVG_RATING a)
```

RATES (DRINKER, COFFEE, SCORE)

- What is the highest rated coffee, on average?
- Actually, can be a lot easier with `LIMIT` k.

```
CREATE VIEW COFFEE_AVG_RATING AS
SELECT r.COFFEE, AVG (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.COFFEE

SELECT  a.COFFEE
FROM COFFEE_AVG_RATING a
ORDER BY a.AVG_RATING DESC LIMIT 1;
```

TOP k / LIMIT k

- What is the highest rated coffee, on average?
- Actually, can be a lot easier with `LIMIT` k.

- Can choose `ASC` or `DESC`
- Finally: note that `ORDER BY` can be used without `LIMIT`

? Will this approach always work?

More True/False Questions

1. ORDER BY only sorts by a single attribute
2. All attributes in the ORDER BY clause are sorted by the same ASC or DESC rule
3. GROUP BY ignores NULL values
4. Aggregate functions ignore NULL values
5. Aggregate functions are a pain to use and are slow. You are better off implementing your own version of them

Questions?