COMP 330/543: SQL 2

Luis Guzman

Sinan Kockara

Chris Jermaine

Rice University

Aggregations

Can compute simple statistics using SQL

- SUM
- AVERAGE (AVG)
- COUNT
- MAX
- MIN
- etc.

Question: What do all of these aggregates have in common?

Our First Aggregation

RATES (DRINKER, BEER, SCORE)

What is the average beer rating given by Luis?

Our First Aggregation

RATES (DRINKER, BEER, SCORE)

What is the average beer rating given by Luis?

```
SELECT AVERAGE (r.SCORE)
FROM RATES r
WHERE r.DRINKER = 'Luis'
```

COUNT DISTINCT

RATES (DRINKER, BEER, SCORE)

How many beers has Luis rated?

COUNT

RATES (DRINKER, BEER, SCORE)

How many beers has Luis rated?

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

Does it work?

- Counts the number of ratings due to Luis.
- What is the primary key for RATES?

COUNT DISTINCT

RATES (DRINKER, BEER, SCORE)

How many beers has Luis rated?

This gives us the actual number rated:

```
SELECT COUNT (DISTINCT r.BEER)
FROM RATES r
WHERE r.DRINKER = 'Luis'
```

GROUP BY

RATES (DRINKER, BEER, SCORE)

It is often desirable to compute an aggregate at a finer level of granularity.

Q: What is the average rating for each beer?

GROUP BY

RATES (DRINKER, BEER, SCORE)

It is often desirable to compute an aggregate at a finer level of granularity.

Q: what is the average rating for each beer?

```
SELECT r.BEER, AVERAGE (r.RATING)
FROM RATES r
GROUP BY r.BEER
```

- This first groups the relation into subsets
- Every tuple in the subset has the same value for r.BEER
- Then aggregate run over each subset independently

GROUP BY

```
SELECT r.BEER, AVERAGE (r.RATING)
FROM RATES r
GROUP BY r.BEER
```

- Example input:

```
('Sinan', 'SSTP', 8)
('Chris', 'PBR', 1)
('Chris', 'SSTP', 10)
('Luis', 'PBR', 5)
('Luis', 'Modelo', 7)
('Sinan', 'Modelo', 6)
```

GROUP BY

```
SELECT r.BEER, AVERAGE (r.RATING)
FROM RATES r
GROUP BY r.BEER
```

- Example input:

```
('Sinan', 'SSTP', 8)
('Chris', 'PBR', 1)
('Chris', 'SSTP', 10)
('Luis', 'PBR', 5)
('Luis', 'Modelo', 7)
('Sinan', 'Modelo', 6)
```

- Output:

```
('SSTP', 9)
('PBR', 3)
('Modelo', 6.5)
```

GROUP BY

```
SELECT r.BEER, AVERAGE (r.RATING)
FROM RATES r
GROUP BY r.BEER
```

- Also note: If you have an attribute outside of an agg function in an agg query
- Example: r.BEER here
- Then you must have grouped by that attribute
- Or query will not compile
- Why?

GROUP BY

Q: what is the average score per drinker?

```
SELECT r.DRINKER, AVERAGE (r.RATING)
FROM RATES r
GROUP BY r.DRINKER
```

```
('Sinan', 'SSTP', 8)
('Chris', 'PBR', 1)
('Chris', 'SSTP', 10)
('Luis', 'PBR', 5)
('Luis', 'Modelo', 7)
('Sinan', 'Modelo', 6)
```

GROUP BY

Q: what is the average score per drinker?

1. Group by DRINKER

```
('Sinan', 'SSTP', 8)
('Sinan', 'Modelo', 6)

('Chris', 'PBR', 1)
('Chris', 'SSTP', 10)

('Luis', 'PBR', 5)
('Luis', 'Modelo', 7)
```

GROUP BY

Q: what is the average score per drinker?

1. Group by DRINKER
2. Agreggate

```
('Sinan', 7)
('Chris', 5.5)
('Luis', 6)
```

Aggregates & NULL values

What about NULL?

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

Subquery in FROM

- Can have a subquery in FROM clause
- Treated as a temporary table
- MUST be assigned an alias

FREQUENTS (DRINKER, BAR)

SERVES (BAR, BEER)

Q: Who goes to a bar that serves 'Modelo'?

Subquery in FROM

FREQUENTS (DRINKER, BAR)

SERVES (BAR, BEER)

Q: Who goes to a bar that serves 'Modelo'?

1. Standard way

```
SELECT DISTINCT f.DRINKER
FROM FREQUENTS f, SERVES s
WHERE f.BAR = s.BAR
        AND s.BEER = 'Modelo'
```

Subquery in FROM

Q: Who goes to a bar that serves 'Modelo'?

2. Subquery in FROM

```
SELECT DISTINCT f.DRINKER
FROM FREQUENTS f,
    (SELECT s.BAR FROM SERVES s
     WHERE s.BEER = 'Modelo') s2
WHERE f.BAR = s2.BAR
```

Subquery in FROM

Q: Who goes to a bar that serves 'Modelo'?

- Method comparison

```
SELECT DISTINCT f.DRINKER
FROM FREQUENTS f, SERVES s
WHERE f.BAR = s.BAR
        AND s.BEER = 'Modelo'

SELECT DISTINCT f.DRINKER
FROM FREQUENTS f,
    (SELECT s.BAR FROM SERVES s
     WHERE s.BEER = 'Modelo') s2
WHERE f.BAR = s2.BAR
```

Subquery in FROM

Q: Who goes to a bar that serves 'Modelo'?

- Using views to clean it up a little

```
CREATE VIEW MODELO_BARS AS
SELECT s.BAR FROM SERVES s
    WHERE s.BEER = 'Modelo'

SELECT DISTINCT f.DRINKER
FROM FREQUENTS f, MODELO_BARS m
WHERE f.BAR = m.BAR
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

1. Find the average ratings for all beers

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

1. Find the average ratings for all beers

```
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
     GROUP BY r.BEER
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

1. Find the average ratings for all beers
2. Select the one with the highest rating
   2.1 Need to find the highest (MAX) rating first

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

1. Find the average ratings for all beers
2. Select the one with the highest rating
   2.1 Need to find the highest (MAX) rating first

```
SELECT MAX b.AVG_RATING
FROM (SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
      FROM RATES r
      GROUP BY r.BEER) b
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

3. Putting it all together

```
SELECT a.BEER
FROM (SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
      FROM RATES r
      GROUP BY r.BEER) a
WHERE a.AVG_RATING = (SELECT MAX (b.AVG_RATING)
                      FROM (SELECT r.BEER, AVERAGE (r.SCORE)
                                AS AVG_RATING
                            FROM RATES r
                            GROUP BY r.BEER) b)
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Note: having a view really helps readability here

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT a.BEER
FROM AVG_RATES a
WHERE a.AVG_RATING = (SELECT MAX (b.AVG_RATING)
                      FROM AVG_RATES b)
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Will this work?

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT a.BEER, MAX (a.AVG_RATING)
FROM AVG_RATES a
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

Will this work?

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT a.BEER, MAX (a.AVG_RATING)
FROM AVG_RATES a
```

- No, won't even compile as we are selecting an attribute (BEER) in an agg query when we have not grouped on that attribute

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

What about this?

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT a.BEER, MAX (a.AVG_RATING)
FROM AVG_RATES a GROUP BY a.BEER
```

Subquery in FROM

RATES (DRINKER, BEER, SCORE)

What about this?

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT a.BEER, MAX (a.AVG_RATING)
FROM AVG_RATES a GROUP BY a.BEER
```

- Again no, though this will compile, the end result is not what we want

Subquery in FROM

Example contents for AVG_RATES:

```
('SSTP', 9)
('PBR', 3)
('Modelo', 6)
```

Grouping by beer:

```
('SSTP', 9)

('PBR', 3)

('Modelo', 6)
```

Applying the aggregate function (MAX) per subgroup:

```
('SSTP', 9)
('PBR', 3)
('Modelo', 6)
```

Top k

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

Actually, it can be a lot easier with top k.

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER
```

Top k

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

Actually, it can be a lot easier with top k.

```
CREATE VIEW AVG_RATES AS
SELECT r.BEER, AVERAGE (r.SCORE) AS AVG_RATING
FROM RATES r
GROUP BY r.BEER

SELECT TOP (1) a.BEER
FROM AVG_RATES a
ORDER BY a.AVG_RATING DESC;
```

Will this work every time?

Top k

RATES (DRINKER, BEER, SCORE)

Q: What is the highest-rated beer, on average?

Actually, it can be a lot easier with top k.

- Can optionally use the PERCENT keyword
- Can add WITH TIES
- Can choose ASC or DESC
- Finally: note that ORDER BY can be used without TOP

Questions?