AGGREGATION, GROUPING & OUTER JOINS

7

Aggregation

8

- SUM, AVG, COUNT, MIN, and MAX can be applied to a column in a SELECT clause to produce that aggregation on the column.
- COUNT(*) counts the number of tuples.

8

Example: Aggregation

9

- From Sells(bar, beer, price), find the average price of Bud:

```
SELECT AVG(price)
FROM Sells
WHERE beer = 'Bud';
```

9

Eliminating Duplicates in an Aggregation

10

- Use DISTINCT inside an aggregation.
- Example: find the number of *different* prices charged for Bud:

```
SELECT COUNT(DISTINCT price)
 FROM Sells
 WHERE beer = 'Bud';
```

10

NULL's Ignored in Aggregation

11

- NULL never contributes to a sum, average, or count, and can never be the minimum or maximum of a column.
- But if all the values in a column are NULL, then the result of the aggregation is NULL.
  - Exception: COUNT of an empty set is 0.

11

Example: Effect of NULL's

12

Sells(bar, beer, price)

```
SELECT count(*)
FROM Sells
WHERE beer = 'Bud';
```

The number of bars that sell Bud.

```
SELECT count(price)
FROM Sells
WHERE beer = 'Bud';
```

The number of bars that sell Bud at a known price (i.e., where price is not NULL)

12

Example Query

13

- Find the age of the youngest employee at each rating level

```
SELECT MIN (age)
FROM Employees
WHERE  rating = i
```

13

Grouping

14

- We may follow a SELECT-FROM-WHERE expression by GROUP BY and a list of attributes.
- The relation that results from the SELECT-FROM-WHERE is grouped according to the values of all those attributes, and any aggregation is applied only within each group.

```
SELECT   rating, MIN(age)
FROM       Employees
GROUP BY  rating
```

14

Example: Grouping

- From Sells(bar, beer, price), find the average price for each beer:

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer;
```

beer	AVG(price)
Bud	2.33
Miller	4.55
…	…

15

Example: Grouping

- From Sells(bar, beer, price) and Frequents(drinker, bar), find for each drinker the average price of Bud at the bars they frequent:

```
SELECT drinker, AVG(price)
FROM Frequents, Sells
WHERE beer = 'Bud' AND
      Frequents.bar = Sells.bar
GROUP BY drinker;
```

Compute all drinker-bar-price triples for Bud.

Then group them by drinker.

16

Restriction on SELECT Lists With Aggregation

- If any aggregation is used, then each element of the SELECT list must be either:
  1. Aggregated, or
  2. An attribute on the GROUP BY list.

17

Illegal Query Example

```
SELECT bar, beer, MIN(price)
FROM Sells
GROUP BY bar
```

- But this query is illegal in SQL.
- Only one tuple output for each bar, no unique way to select which beer to output

18

A Closer Look

```
SELECT bar, beer, MIN(price) AS minP
FROM Sells
GROUP BY bar  beer
```

Sells

Bar	Beer	Price
Joe	Bud	3.00
Joe	Miller	4.00
Tom	Bud	3.50
Tom	Miller	4.25
Jane	Bud	3.25
Jane	Miller	4.75
Jane	Coors	4.00

Result

bar	beer	minP
Joe	?	3.00
Tom	?	3.50
Jane	?	3.25

{Bud, Miller, Coors}?

Only one tuple output for each bar, no unique way to select which beer to output

19

HAVING Clauses

- HAVING <condition> may follow a GROUP BY clause.
- If so, the condition applies to each group, and groups not satisfying the condition are eliminated.

20

Example: HAVING

- From Sells(bar, beer, price) and Beers(name, manf), find the average price of those beers that are either served in at least three bars or are manufactured by Pete's.

21

Solution

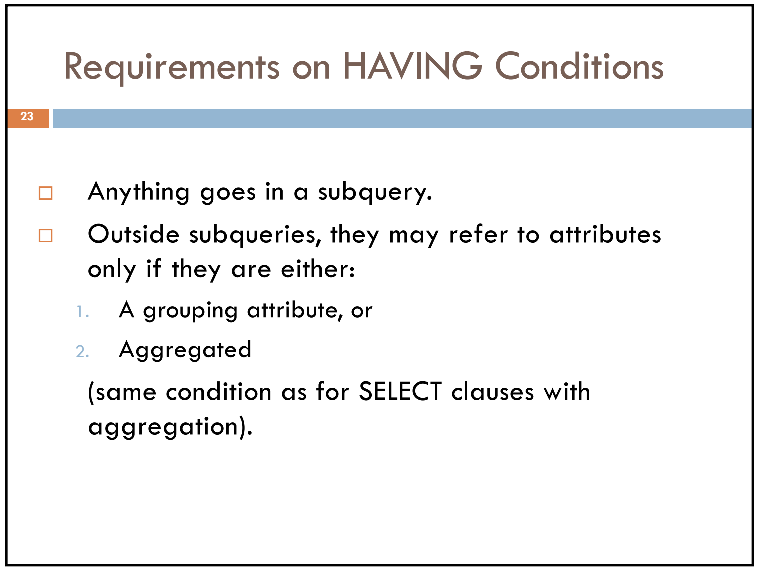
Sells(bar, beer, price) and Beers(name, manf),

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer
HAVING COUNT(bar) >= 3 OR
    beer IN (SELECT name
             FROM Beers
             WHERE manf = 'Pete''s');
```

Beer groups with at least 3 non-NULL bars

Beers manu-factured by Pete's.

22



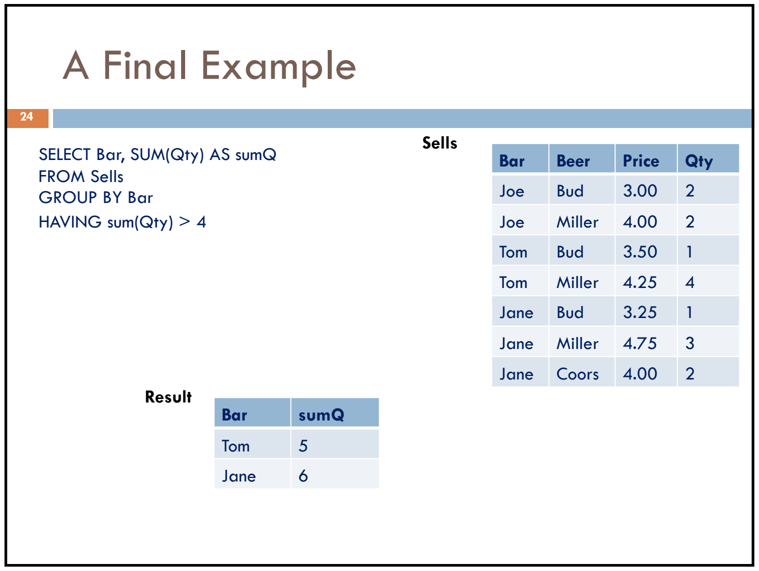
Requirements on HAVING Conditions

23

- Anything goes in a subquery.
- Outside subqueries, they may refer to attributes only if they are either:
  1. A grouping attribute, or
  2. Aggregated

  (same condition as for SELECT clauses with aggregation).

23



A Final Example

24

```
SELECT Bar, SUM(Qty) AS sumQ
FROM Sells
GROUP BY Bar
HAVING sum(Qty) > 4
```

Sells

Bar	Beer	Price	Qty
Joe	Bud	3.00	2
Joe	Miller	4.00	2
Tom	Bud	3.50	1
Tom	Miller	4.25	4
Jane	Bud	3.25	1
Jane	Miller	4.75	3
Jane	Coors	4.00	2

Result

Bar	sumQ
Tom	5
Jane	6

24