

# Introduction to Data Management

## CSE 344

### Lecture 5: Grouping and Query Evaluation

# Announcements

- Web quiz 2 is open: due next Tuesday 11pm
- Homework 2 is released: due next Wednesday 11pm

# Review

- Selection
- Projection
- Join
  - Inner and outer
- Aggregates

# Today

- Aggregations and grouping (6.4.3 – 6.4.6)
- Order of query evaluation

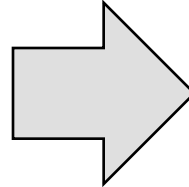
# Grouping and Aggregation

`Purchase(product, price, quantity)`

Find total quantities for all sales over \$1, by product.

# Grouping and Aggregation

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10



Product	TotalSales
Bagel	40
Banana	20

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

# Other Examples

Compare these  
two queries:

```
SELECT product, count(*)  
FROM Purchase  
GROUP BY product
```

```
SELECT month, count(*)  
FROM Purchase  
GROUP BY month
```

```
SELECT product,  
       sum(quantity) AS SumQuantity,  
       max(price) AS MaxPrice  
FROM Purchase  
GROUP BY product
```

What does  
it mean ?

# Need to be Careful...

```
SELECT product,  
       max(quantity)  
FROM   Purchase  
GROUP BY product
```

```
SELECT product, quantity  
FROM   Purchase  
GROUP BY product
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10



```
SELECT product, quantity
FROM Purchase
GROUP BY product
```

# Need to be Careful...

```
SELECT product
FROM Purchase
GROUP BY product
```

```
SELECT quantity
FROM Purchase
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product
Bagel
Banana

+

???

Quantity
20
20
50
10
10

```
SELECT product, quantity
FROM Purchase
GROUP BY product
```



Can't project a non-grouped / non-aggregated column!

# Need to be Careful...

```
SELECT product,  
       max(quantity)  
FROM   Purchase  
GROUP BY product
```

```
SELECT product, quantity  
FROM   Purchase  
GROUP BY product
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

sqlite is WRONG on  
this query.

Advanced DBMS (e.g. SQL  
Server) gives an error

# Grouping and Aggregation

Purchase(product, price, quantity)

Find total quantities for all sales over \$1, by product.

```
SELECT    product, Sum(quantity) AS TotalSales
FROM      Purchase
WHERE     price > 1
GROUP BY  product
```

How is this query processed?

# Grouping and Aggregation

1. Compute the `FROM` and `WHERE` clauses.
2. Group by the attributes in the `GROUPBY`
3. Compute the `SELECT` clause:  
grouped attributes and aggregates.



# 1,2: From, Where

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	<del>0.5</del>	<del>50</del>
Banana	2	10
Banana	4	10

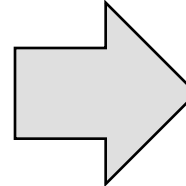
WHERE price > 1

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

## 3,4. Grouping, Select

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	<del>0.5</del>	<del>50</del>
Banana	2	10
Banana	4	10



Product	TotalSales
Bagel	40
Banana	20

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

Purchase(pid, product, price, quantity, month)

## Ordering Results

```
SELECT product, sum(price*quantity) as rev
FROM Purchase
GROUP BY product
ORDER BY rev desc
```

FWGOS

TM

Note: some SQL engines  
want you to say ORDER BY sum(price\*quantity) desc

Purchase(pid, product, price, quantity, month)

## HAVING Clause

Same query as before, except that we consider only products that had at least 30 sales.

```
SELECT    product, sum(price*quantity)
FROM      Purchase
WHERE     price > 1
GROUP BY  product
HAVING    sum(quantity) > 30
```

HAVING clause contains conditions on aggregates.



# General form of Grouping and Aggregation

SELECT	S
FROM	$R_1, \dots, R_n$
WHERE	C1
GROUP BY	$a_1, \dots, a_k$
HAVING	C2

Why ?

S = may contain attributes  $a_1, \dots, a_k$  and/or any aggregates but **NO OTHER ATTRIBUTES**

C1 = is any condition on the attributes in  $R_1, \dots, R_n$

C2 = is any condition on aggregate expressions and on attributes  $a_1, \dots, a_k$

# Semantics of SQL With Group-By

SELECT	S
FROM	$R_1, \dots, R_n$
WHERE	C1
GROUP BY	$a_1, \dots, a_k$
HAVING	C2

FWGHOS

Evaluation steps:

1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes  $a_1, \dots, a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as "TotalSold"

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as "TotalSold"

```
FROM Purchase
```

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as “TotalSold”

```
FROM Purchase
GROUP BY month
```

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as “TotalSold”

```
FROM      Purchase
GROUP BY  month
HAVING    sum(quantity) < 10
```

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as “TotalSold”

```
SELECT    month, sum(price*quantity),  
          sum(quantity) as TotalSold  
FROM      Purchase  
GROUP BY  month  
HAVING    sum(quantity) < 10
```

Purchase(pid, product, price, quantity, month)

## Exercise

Compute the total income per month

Show only months with less than 10 items sold

Order by quantity sold and display as “TotalSold”

```
SELECT      month, sum(price*quantity),  
            sum(quantity) as TotalSold  
FROM        Purchase  
GROUP BY   month  
HAVING     sum(quantity) < 10  
ORDER BY   sum(quantity)
```



# WHERE vs HAVING

- WHERE condition is applied to individual rows
  - The rows may or may not contribute to the aggregate
  - No aggregates allowed here
- HAVING condition is applied to the entire group
  - Entire group is returned, or not at all
  - May use aggregate functions in the group

Purchase(pid, product, price, quantity, month)

# Mystery Query

What do they compute?

```
SELECT    month, sum(quantity), max(price)
FROM      Purchase
GROUP BY  month
```

```
SELECT    month, sum(quantity)
FROM      Purchase
GROUP BY  month
```

```
SELECT    month
FROM      Purchase
GROUP BY  month
```

Purchase(pid, product, price, quantity, month)

# Mystery Query

What do they compute?

```
SELECT    month, sum(quantity), max(price)
FROM      Purchase
GROUP BY  month
```

```
SELECT    month, sum(quantity)
FROM      Purchase
GROUP BY  month
```

```
SELECT    month
FROM      Purchase
GROUP BY  month
```

Lesson:  
DISTINCT is  
a special case  
of GROUP BY

Purchase(pid,product,price,quantity,month)

Product(pid,pname,manufacturer)

## Aggregate + Join Example

```
SELECT x.manufacturer, count(*)  
FROM Product x, Purchase y  
WHERE x.pname = y.product  
GROUP BY x.manufacturer
```

What do these queries mean?

manufa cturer	month	count(*)
canon	1	10
canon	2	20
sony	4	50

```
SELECT x.manufacturer, y.month, count(*)  
FROM Product x, Purchase y  
WHERE x.pname = y.product  
GROUP BY x.manufacturer, y.month
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10

## Empty Groups

FWGHOS

- In the result of a group by query, there is one row per group in the result
- No group can be empty!
  - i.e., `count(*)` is never 0

```
SELECT x.manufacturer, count(*)  
FROM Product x, Purchase y  
WHERE x.pname = y.product  
GROUP BY x.manufacturer
```

What if there  
are no  
purchases for a  
manufacturer

# Empty Group Solution: Outer Join

```
SELECT x.manufacturer, count(y.pid)
FROM Product x LEFT OUTER JOIN Purchase y
ON x.pname = y.product
GROUP BY x.manufacturer
```