Announcements

- HW2 is out
  - due next Tuesday 11pm
  - same format as HW1
  - uses joins, aggregation, grouping
- WQ2 due Sunday 11pm

Outline

- Last time:
  - outer joins
  - how to aggregate over all rows
- Grouping & aggregations (6.4.3 – 6.4.6)

Aggregation

Purchase(product, price, quantity)

Find number of bagels sold for more than $1

```
SELECT Sum(quantity) as TotalSold
FROM Purchase
WHERE price > 1 and product = 'bagel'
```

Grouping and Aggregation

Purchase(product, price, quantity)

Find number sold for more than $1 for each product

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

Let's see what this means...

Grouping and Aggregation

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

FWGS
## 1&2. FROM-WHERE-GROUPBY

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

WHERE price > 1

## 3. SELECT

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

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

### Other Examples

<table>
<thead>
<tr>
<th>Purchase</th>
<th>pid, product, price, quantity, month</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>product, count(*) FROM Purchase GROUP BY product</td>
</tr>
<tr>
<td>SELECT</td>
<td>month, count(*) FROM Purchase GROUP BY month</td>
</tr>
<tr>
<td>SELECT</td>
<td>product, sum(quantity) AS SumQuantity, max(price) AS MaxPrice FROM Purchase GROUP BY product</td>
</tr>
</tbody>
</table>

How about this one?

sqlite is WRONG on this query.

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

### Need to be Careful...

<table>
<thead>
<tr>
<th>Purchase</th>
<th>pid, product, price, quantity, month</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>product, max(quantity) FROM Purchase GROUP BY product</td>
</tr>
<tr>
<td>SELECT</td>
<td>product, quantity FROM Purchase GROUP BY month</td>
</tr>
</tbody>
</table>

### Ordering Results

<table>
<thead>
<tr>
<th>Purchase</th>
<th>pid, product, price, quantity, month</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>product, sum(price<em>quantity) FROM Purchase GROUP BY product ORDER BY sum(price</em>quantity) DESC</td>
</tr>
</tbody>
</table>

FWGOS

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

Same query as earlier, 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 groups.

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

Lesson:
DISTINCT is a special case of GROUP BY

Aggregates and Joins

```
create table Product(
    pid int primary key,
    pname varchar(15),
    manufacturer varchar(15));

insert into product values(1,'bagel','Sunshine Co.');
insert into product values(2,'banana','BusyHands');
insert into product values(3,'gizmo','GizmoWorks');
insert into product values(4,'gadget','BusyHands');
insert into product values(5,'powerGizmo','PowerWorks');
```

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)
```

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
```

Let’s figure out what these mean...

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

```
SELECT manufacturer, month, count(*) 
FROM Product, Purchase 
WHERE pname = product 
GROUP BY manufacturer, month
```
Nested Loop Semantics for SFW

SELECT x1.a1, x2.a2, ..., xm.am
FROM R1 as x1, R2 as x2, ..., Rm as xm
WHERE Cond

for x1 in R1:
  for x2 in R2:
    ...
  for xm in Rm:
    if Cond(x1, x2, ...):
      output(x1.a1, x2.a2, ..., xm.am)

Semantics for SFWGH

SELECT S
FROM R1, ..., Rn
WHERE C1
GROUP BY a1, ..., ak
HAVING C2

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes a1, ..., ak
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Aggregate + Join Example

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

Empty Groups

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

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

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes a1, ..., ak
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Execution order:
FWGHOS
Empty Group Solution: Outer Join

```
SELECT manufacturer, count(quantity)
FROM Product LEFT OUTER JOIN Purchase
ON pname = product
GROUP BY manufacturer
```

Why not count(*)?

Exercise:

```
SELECT manufacturer, sum(quantity)
FROM Product, Purchase
WHERE pname = product
GROUP BY manufacturer
HAVING sum(quantity) > 10
```

Find all manufacturers with more than 10 items sold. Return manufacturer name and number of items sold.

Exercise:

```
SELECT manufacturer, count(distinct product)
FROM Product, Purchase
WHERE pname = product
GROUP BY manufacturer
HAVING count(distinct product) > 1
```

Find all manufacturers with more than 1 distinct product sold. Return the name of the manufacturer and number of distinct products sold.

Exercise:

```
SELECT pname, max(price)
FROM Product, Purchase
WHERE pname = product
GROUP BY pname
HAVING COUNT(*) > 2
```

Find all products with more than 2 purchases. Return the name of the product and max price it was sold.

Exercise:

```
SELECT manufacturer, month, sum(quantity)
FROM Product, Purchase
GROUP BY manufacturer, month
HAVING count(*) > 2
```

Find all manufacturers with at least 5 purchases in one month. Return manufacturer name, month, and number of items sold.