Introduction to Data Management
CSE 344

Lectures 5: More SQL aggregates
Announcements

• Homework 1 is due on Thursday, 11 pm
  – Max 2 late days can be used

• Office hours start this week
  – Check webpage for details
Outline

• Outer joins (6.3.8, review)
• More aggregations (6.4.3 – 6.4.6)
```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase
ON Product.name = Purchase.prodName
```
```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON Product.name = Purchase.prodName
```

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Category</td>
<td></td>
</tr>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td></td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Purchase</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ProdName</td>
<td>Store</td>
<td></td>
</tr>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
<td></td>
</tr>
</tbody>
</table>
### Product

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

### Purchase

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

---

**SQL Query**

```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase
ON Product.name = Purchase.prodName
```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON
Product.name = Purchase.prodName
```
SELECT Product.name, Purchase.store 
FROM Product 
JOIN Purchase ON Product.name = Purchase.prodName
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
</tbody>
</table>
```
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON Product.name = Purchase.prodName
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
</tbody>
</table>
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON Product.name = Purchase.prodName

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON Product.name = Purchase.prodName
SELECT Product.name, Purchase.store
FROM Product
LEFT OUTER JOIN Purchase
ON Product.name = Purchase.prodName
SELECT Product.name, Purchase.store
FROM Product FULL OUTER JOIN Purchase
ON Product.name = Purchase.prodName

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
<tr>
<td>OneClick</td>
<td>NULL</td>
</tr>
<tr>
<td>NULL</td>
<td>Foo</td>
</tr>
</tbody>
</table>
Outer Joins

- **Left outer join:**
  - Include the left tuple even if there’s no match
- **Right outer join:**
  - Include the right tuple even if there’s no match
- **Full outer join:**
  - Include both left and right tuples even if there’s no match
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.

FWGS
Grouping and Aggregation

```
SELECT product, sum(price*quantity) as rev
FROM purchase
GROUP BY product
```
Grouping and Aggregation

\[
\text{SELECT} \quad \text{product, sum(price*quantity) as rev} \\
\text{FROM} \quad \text{purchase} \\
\text{GROUP BY product}
\]

\[
\text{SELECT} \quad \text{product, price*quantity as rev} \\
\text{FROM} \quad \text{purchase}
\]
Ordering Results

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

```
SELECT product, sum(price*quantity) as rev
FROM purchase
GROUP BY product
ORDER BY rev desc
```
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 aggregates.
Exercise

Compute the total income per month
Show only months with less than 10 items sold
Order by quantity sold and display as “TotalSold”
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
```
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
```
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
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 TotalSold < 10
```
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 TotalSold < 10
```
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 TotalSold < 10
ORDER BY TotalSold
```
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
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');
Aggregate + Join Example

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

What do these query mean?

SELECT x.manufacturer, y.month, count(*)
FROM Product x, Purchase y
WHERE x.pname = y.product
GROUP BY x.manufacturer, y.month
General form of Grouping and Aggregation

\[
\text{SELECT} \quad S \\
\text{FROM} \quad R_1, \ldots, R_n \\
\text{WHERE} \quad C1 \\
\text{GROUP BY} \quad a_1, \ldots, a_k \\
\text{HAVING} \quad C2
\]

S = may contain attributes \( a_1, \ldots, a_k \) and/or any aggregates but NO OTHER ATTRIBUTES

C1 = is any condition on the attributes in \( R_1, \ldots, R_n \)

C2 = is any condition on aggregate expressions and on attributes \( a_1, \ldots, a_k \)
Semantics of SQL With Group-By

```
SELECT S
FROM R_1,...,R_n
WHERE C1
GROUP BY a_1,...,a_k
HAVING C2
```

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes \( a_1,\ldots,a_k \)
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result
Semantics of SQL With Group-By

```
SELECT  S
FROM    R_1,...,R_n
WHERE   C1
GROUP BY a_1,...,a_k
HAVING  C2
```

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