Introduction to Data Management
CSE 344

Lectures 5: More SQL aggregates

Announcements

- Web quiz 2 is open: due Sunday 11pm
- Homework 2 is released: Tuesday 11p

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
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

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

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
Phone	Foo

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL
NULL	Foo

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.

$\boxed{\text{FWGS}}$

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

Grouping and Aggregation

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

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

Grouping and Aggregation

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

```
SELECT product, price*quantity as rev
FROM    Purchase
```

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

Ordering Results

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

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

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

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 aggregates.

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"

```
SELECT      month, sum(price*quantity),
            sum(quantity) 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”

```
SELECT      month, sum(price*quantity),
            sum(quantity) 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”

```
SELECT      month, sum(price*quantity),
            sum(quantity) 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”

```
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(quanity), max(price)
FROM        Purchase
GROUP BY    month
```

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

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

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

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');
```

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?

```
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

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

Why ?

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
```

FWGHOS

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

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

What if there are no purchases for a manufacturer

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