Database Management Systems

Lecture 5: Aggregates in SQL (continued)
Logistics

- HW1 was out last Thursday
  - Due on 9/27
- Extra credit on paper review (for CS457)
  - No
  - May add something else later
    - More features in your programming assignment
      - Nested queries…
Logistics

• More clarifications on HW1
  – No Sqlite3
  – Where to store table’s attributes? (Demo?)
  – If you are writing huge programs, e.g., 1000+ lines of code, let me know…

• More questions on HW1?
Ordering Results

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

```sql
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”

```sql
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 sum(quantity) < 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      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
Aggregate + Join Example

What do these query mean?

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

```
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 R₁,…,Rₙ | WHERE C₁ | GROUP BY a₁,…,aₖ | HAVING C₂ |

S = may contain attributes a₁,…,aₖ and/or any aggregates but NO OTHER ATTRIBUTES

C₁ = is any condition on the attributes in R₁,…,Rₙ

C₂ = is any condition on aggregate expressions and on attributes a₁,…,aₖ

**Why?**
Semantics of SQL With Group-By

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

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes $a_1, \ldots, a_k$
3. Apply condition $C_2$ 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

```sql
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
```