CSE 344

APRIL 2\textsuperscript{ND} – GROUPING/AGGREGATION
ADMINISTRIVIA

• HW1 Due Wednesday (11:30)
  • Don’t forget to git add and tag your assignment
  • Check on gitlab after submitting

• OQ1 Due Friday (11:00)
  • A few of you still need to enroll
As the information we want gets more complex, we need to utilize more elements of the RDBMS:

- Multi-table queries -> join
- Data statistics -> grouping
QUERY COMPLEXITY

- As the information we want gets more complex, we need to utilize more elements of the RDBMS
  - Multi-table queries -> join
  - Data statistics -> grouping
- Whatever you can do in SQL, you should
  - Optimization
  - Basic analysis tools
    - Sum, min, average, max, count
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
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
NEED TO BE CAREFUL...

```
SELECT product, max(quantity)
FROM Purchase
GROUP BY product
```
NEED TO BE CAREFUL...

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

-- what does this mean?

<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>
NEED TO BE CAREFUL...

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

```
SELECT product, quantity
FROM Purchase
GROUP BY product
-- what does this mean?
```

<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>

<table>
<thead>
<tr>
<th>Product</th>
<th>Max(quantity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>50</td>
</tr>
</tbody>
</table>
NEED TO BE CAREFUL...

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

-- what does this mean?

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Max(quantity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>50</td>
</tr>
</tbody>
</table>

<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>

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>??</td>
</tr>
</tbody>
</table>
Everything in SELECT must be either a GROUP-BY attribute, or an aggregate

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

-- what does this mean?

<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>

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>??</td>
</tr>
</tbody>
</table>
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

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

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

Do these queries return the same number of rows? Why?

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
GROUP BY product
```
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
```

Do these queries return the same number of rows? Why?

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
GROUP BY product
```

Empty groups are removed, hence first query may return fewer groups
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.
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
3,4. GROUPING, SELECT

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

<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>1.5</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>

<table>
<thead>
<tr>
<th>Product</th>
<th>TotalSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>40</td>
</tr>
<tr>
<td>Banana</td>
<td>20</td>
</tr>
</tbody>
</table>
SELECT product, sum(price*quantity) as rev
FROM Purchase
GROUP BY product
ORDER BY rev desc

Note: some SQL engines want you to say ORDER BY sum(price*quantity) desc
Having clause contains conditions on aggregates.

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
```
GENERAL FORM OF GROUPING AND AGGREGATION

SELECT S 
FROM $R_1, \ldots, R_n$
WHERE C1 
GROUP BY $a_1, \ldots, a_k$
HAVING 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 C_1
GROUP BY a_1, ..., a_k
HAVING C_2
```

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantic
2. Group by the attributes $a_1, ..., a_k$
3. Apply condition $C_2$ to each group (may have aggregates)
4. Compute aggregates in $S$ and return the result
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”

```
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”

```
FROM Purchase
GROUP BY month
```
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
```
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”

```sql
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
- Occasionally, some groups become empty and are removed

HAVING condition is applied to the entire group

- Entire group is returned, or removed
- May use aggregate functions on the group
What do they compute?

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

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

```sql
SELECT month
FROM Purchase
GROUP BY month
```
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
AGGREGATE + JOIN

For each manufacturer, compute how many products with price > $100 they sold
AGGREGATE + JOIN

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Purchase, price is in Product...
AGGREGATE + JOIN

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Purchase, price is in Product...

```
-- step 1: think about their join
SELECT ...
FROM Product x, Purchase y
WHERE x.pid = y.product_id
    and y.price > 100
```
AGGREGATE + JOIN

For each manufacturer, compute how many products with price > $100 they sold

Problem: manufacturer is in Purchase, price is in Product...

<table>
<thead>
<tr>
<th>manufacture</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi</td>
<td>150</td>
</tr>
<tr>
<td>Canon</td>
<td>300</td>
</tr>
<tr>
<td>Hitachi</td>
<td>180</td>
</tr>
</tbody>
</table>

-- step 1: think about their join
SELECT ...
FROM Product x, Purchase y
WHERE x.pid = y.product_id
    and y.price > 100

-- step 2: do the group-by on the join
SELECT x.manufacturer, count(*)
FROM Product x, Purchase y
WHERE x.pid = y.product_id
    and y.price > 100
GROUP BY x.manufacturer
AGGREGATE + JOIN

Variant:
For each manufacturer, compute how many products with price > $100 they sold in each month

```
SELECT x.manufacturer, y.month, count(*)
FROM Product x, Purchase y
WHERE x.pid = y.product_id
    and y.price > 100
GROUP BY x.manufacturer, y.month
```
INCLUDING EMPTY GROUPS

In the result of a group by query, there is one row per group in the result

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

Count(pid) is 0 when all pid’s in the group are NULL
A subquery is a SQL query nested inside a larger query

Such inner-outer queries are called nested queries

A subquery may occur in:

• A SELECT clause
• A FROM clause
• A WHERE clause

Rule of thumb: avoid nested queries when possible

• But sometimes it’s impossible to avoid, as we will see
SUBQUERIES...

• Can return a single value to be included in a SELECT clause
• Can return a relation to be included in the FROM clause, aliased using a tuple variable
• Can return a single value to be compared with another value in a WHERE clause
• Can return a relation to be used in the WHERE or HAVING clause under an existential quantifier