CSE 344

JUNE 25TH

GROUPING/AGGREGATION

CH. 6.3-6.4
ADMINISTRIVIA

• WQ1 due **tonight**

• HW1 due Wednesday
  • can leave out sqlite-specific (.e.g., .mode commands)
    • grader uses JDBC (covered later) to check results
    • alternatively: pull a new copy of Grader from upstream
  • run the script to tag and push your assignment
    • check on gitlab after submitting
  • run grader on attu to make sure it passes
REVIEW

• **Relational data model**
  - database is a collection of tables
  - table has a set of named & typed columns
  - table contains a list of (unordered) rows
  - query results can be ordered and/or include duplicates

• **Joins**
  - put related tables back together to answer questions
  - SELECT ... FROM A, B ...
  - SELECT ... FROM A JOIN B ...
  - SELECT ... FROM A JOIN B ON ...
  - with no join condition, result is the Cartesian product
  - outer join adds match with NULLs when no other matches
(INNER) JOINS

```sql
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)

This is called nested loop semantics since we are interpreting what a join means using a nested loop.
ANOTHER EXAMPLE

Product\texttt{(pname, price, category, manufacturer)}
Company\texttt{(cname, country)}
-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both ‘gadget’ and ‘photography’ categories
ANOTHER EXAMPLE

Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both ‘gadget’ and ‘photography’ categories

SELECT DISTINCT z.cname
FROM Product x, Company z
WHERE z.country = 'USA'
    AND x.manufacturer = z.cname
    AND x.category = 'gadget'
    AND x.category = 'photography';

Does this work?
ANOTHER EXAMPLE

Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both ‘gadget’ and ‘photography’ categories

```
SELECT DISTINCT z.cname
FROM Product x, Company z
WHERE z.country = 'USA'
    AND x.manufacturer = z.cname
    AND (x.category = 'gadget'
        OR x.category = 'photography');
```
ANOTHER EXAMPLE

Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both ‘gadget’ and ‘photography’ categories

SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
  AND x.manufacturer = z.cname
  AND y.manufacturer = z.cname
  AND x.category = 'gadget'
  AND y.category = 'photography;

Need to include Product twice!
SELF-JOINS AND TUPLE VARIABLES

Find USA companies that manufacture both products in the ‘gadgets’ and ‘photo’ category

Joining Product with Company is insufficient: need to join Product, with Product, and with Company

When a relation occurs twice in the FROM clause we call it a self-join; in that case we must use tuple variables (why?)
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>SingleTouch</td>
<td>photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>MultiTouch</td>
<td>Photo</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
### Self-Joins

```sql
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
  AND x.category = 'gadget'
  AND y.category = 'photo'
  AND x.manufacturer = z.cname
  AND y.manufacturer = z.cname;
```

---

**Product**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Gizmo</strong></td>
<td>gadget</td>
</tr>
<tr>
<td><strong>SingleTouch</strong></td>
<td>photo</td>
</tr>
<tr>
<td><strong>MultiTouch</strong></td>
<td>Photo</td>
</tr>
</tbody>
</table>

**Company**

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
```
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
```
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
    AND x.category = 'gadget'
    AND y.category = 'photo'
    AND x.manufacturer = z.cname
    AND y.manufacturer = z.cname;
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
OUTER JOINS

Product(name, category)
Purchase(prodName, store)

-- prodName is foreign key

```
SELECT Product.name, Purchase.store
FROM Product, Purchase
WHERE Product.name = Purchase.prodName
```

We want to include products that are never sold, but some are not listed! Why?
OUTER JOINS

Product(name, category)
Purchase(prodName, store)

-- prodName is foreign key

```sql
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON Product.name = Purchase.prodName
```
### SQL Query

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

### Product Table

<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

<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

```sql
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
```
### 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>

### Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
</tbody>
</table>
```sql
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>

### Output

<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

**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>

**Output**

<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
```

**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>

**Output**

<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>
</tbody>
</table>
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
```

**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>

**Output**

<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

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

### 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>

### Output

<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>
### SQL Query

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

### Tables

#### 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>

#### Output

<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>
<tr>
<td>OneClick</td>
<td>NULL</td>
</tr>
</tbody>
</table>
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>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td></td>
<td>Wiz</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>NULL</td>
</tr>
<tr>
<td>NULL</td>
<td></td>
<td>Foo</td>
</tr>
</tbody>
</table>

Output
OUTER JOINS

tableA (LEFT/RIGHT/FULL) OUTER JOIN tableB ON p

Left outer join:
  • Include tuples from tableA even if no match

Right outer join:
  • Include tuples from tableB even if no match

Full outer join:
  • Include tuples from both even if no match

In all cases:
  • Patch tuples without matches using NULL
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
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
  • (DBMSs are good at their job!)
  • Query 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:

```sql
SELECT product, count(*)
FROM Purchase
GROUP BY product
```

```sql
SELECT month, count(*)
FROM Purchase
GROUP BY month
```

```sql
SELECT product,
    sum(quantity) AS SumQuantity,
    max(price) AS MaxPrice
FROM Purchase
GROUP BY product
```

What does it return?
NEED TO BE CAREFUL...

```
SELECT product, max(quantity)
FROM Purchase
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>1.50</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...

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

```sql
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>
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>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
```

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

<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?

SELECT product, quantity
FROM Purchase
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>1.50</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>

<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

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
```
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.
### 1,2: FROM, WHERE

<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>

```sql
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```
### 3.4. GROUPING, SELECT

```sql
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>1.50</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>TotalSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>40</td>
</tr>
<tr>
<td>Banana</td>
<td>20</td>
</tr>
</tbody>
</table>
ORDERING RESULTS

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

Same query as before, 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.
GENERAL FORM OF GROUPING AND AGGREGATION

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

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

\( C_1 = \) is any condition on the attributes in \( R_1, \ldots, R_n \)

\( C_2 = \) is any condition on aggregate expressions and on attributes \( a_1, \ldots, a_k \)
SEMANTICS OF SQL WITH GROUP-BY

\[
\begin{align*}
\text{SELECT} & \quad S \\
\text{FROM} & \quad R_1, \ldots, R_n \\
\text{WHERE} & \quad C_1 \\
\text{GROUP BY} & \quad a_1, \ldots, a_k \\
\text{HAVING} & \quad C_2
\end{align*}
\]

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
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
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
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
```
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
- 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