CSE 344

APRIL 2ND – 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

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**
  - Optimization
  - Basic analysis tools
    - Sum, min, average, max, count

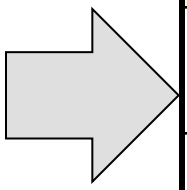
GROUPING AND AGGREGATION

```
Purchase(product, price, quantity)
```

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

GROUPING AND AGGREGATION

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10



Product	TotalSales
Bagel	40
Banana	20

```
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
```

What does it return?

NEED TO BE CAREFUL…

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

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

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?
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

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?
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product	Max(quantity)
Bagel	20
Banana	50

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?
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product	Max(quantity)
Bagel	20
Banana	50

Product	Quantity
Bagel	20
Banana	??

Everything in SELECT must be either a GROUP-BY attribute, or an aggregate

CAREFUL...

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

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

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product	Max(quantity)
Bagel	20
Banana	50

Product	Quantity
Bagel	20
Banana	??

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

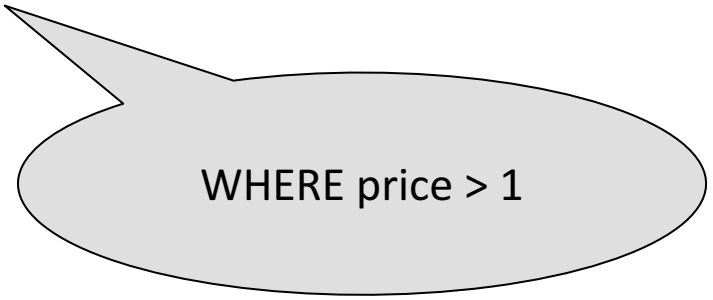
1. Compute the FROM and WHERE clauses.
2. Group by the attributes in the GROUPBY
3. Compute the SELECT clause:
   grouped attributes and aggregates.

FWGS™

1,2: FROM, WHERE

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	~~0.5~~	~~50~~
Banana	2	10
Banana	4	10

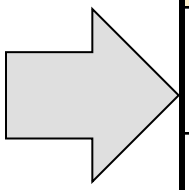



```
SELECT   product, Sum(quantity) AS TotalSales
FROM     Purchase
WHERE    price > 1
GROUP BY product
```

3,4. GROUPING, SELECT

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	~~0.5~~	~~50~~
Banana	2	10
Banana	4	10



Product	TotalSales
Bagel	40
Banana	20

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

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) desc`

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

HAVING CLAUSE

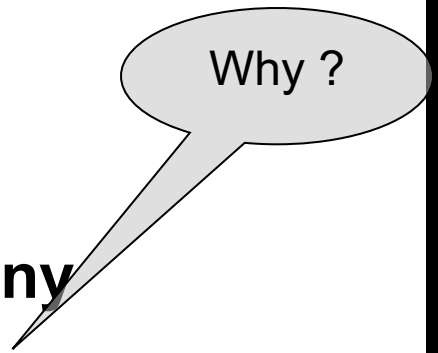
Same query as before, except that we consider only products that had at least 30 sales.

```
SELECT   product, sum(price*quantity)
FROM     Purchase
WHERE    price > 1
GROUP BY product
HAVING   sum(quantity) > 30
```

HAVING clause contains conditions on aggregates.

GENERAL FORM OF GROUPING AND AGGREGATION

```
SELECT    S
FROM      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
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      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
HAVING    C2
```

FWGHOS

Evaluation steps:

1. **Evaluate FROM-WHERE using Nested Loop Semantic**
2. **Group by the attributes $a_1,\ldots,a_k$**
3. **Apply condition C2 to each group (may have aggrega**
4. **Compute aggregates in S and return the result**

```
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”

```
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”

```
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”

```
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
```

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
- 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

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

MYSTERY QUERY

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
```

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

MYSTERY QUERY

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

```
Product(pid,pname,manufacturer)
Purchase(id,product_id,price,month)
```

AGGREGATE + JOIN

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

```
Product(pid,pname,manufacturer)
Purchase(id,product_id,price,month)
```

AGGREGATE + JOIN

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

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

Product(pid,pname,manufacturer)
Purchase(id,product_id,price,month)

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
```

manu facturer	...	price	...
Hitachi		150	
Canon		300	
Hitachi		180	

Product(pid,pname,manufacturer)
Purchase(id,product_id,price,month)

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
```

manu facturer	...	price	...
Hitachi		150	
Canon		300	
Hitachi		180	

```
-- 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
```

manu facturer	count(*)
Hitachi	2
Canon	1
...	

Product(pid,pname,manufacturer)
Purchase(id,product_id,price,month)

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
```

manu facturer	month	count(*)
Hitachi	Jan	2
Hitachi	Feb	1
Canon	Jan	3
...		

INCLUDING EMPTY GROUPS

FWGHOS

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

Count(*) is never 0

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

INCLUDING EMPTY GROUPS

```
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

SUBQUERIES

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