

# Introduction to Database Systems CSE 414

## Lecture 5: SQL Aggregates and Grouping

CSE 414 - Autumn 2018

1

# Announcements

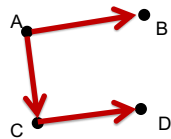
- Web quiz 1 due tonight
- HW 2 due Tuesday at midnight

CSE 414 - Autumn 2018

2

Edge(start, end)

## Self Join Example



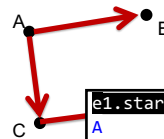
start	end
A	B
A	C
C	D

CSE 414 - Autumn 2018

3

Edge(start, end)

## Self Join Example

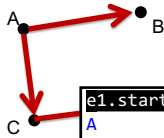


```
SELECT *
FROM Edge e1, Edge e2
```

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C

Edge(start, end)

## Self Join Example

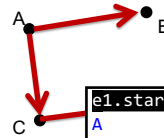


```
SELECT *
FROM Edge e1, Edge e2
WHERE e1.end = e2.start
```

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C

Edge(start, end)

## Self Join Example



```
SELECT *
FROM Edge e1, Edge e2
WHERE e1.end = e2.start
```

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C

Edge(start, end)

### Self Join Example

```
SELECT e1.start, e2.end
FROM Edge e1, Edge e2
WHERE e1.end = e2.start
```

e1.start	e2.end
A	D

start	end
A	B
A	C
C	D

CSE 414 - Autumn 2018 7

### Simple Aggregations

Five basic aggregate operations in SQL

```
select COUNT(*) from Purchase
select SUM(quantity) from Purchase
select AVG(price) from Purchase
select MAX(quantity) from Purchase
select MIN(quantity) from Purchase
```

Except count, all aggregations apply to a single attribute

CSE 414 - Autumn 2018 8

### Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

```
select sum(quantity) from Purchase
```

sum(quantity)
50

9

### Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

```
select avg(price) from Purchase
```

avg(price)
1.7675

10

### Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

```
select count(*) from Purchase
```

count(*)
4

11

### Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

```
select count(quantity) from Purchase
```

count(quantity)
4

12

## Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

`select count(DISTINCT quantity) from Purchase`



count(DISTINCT quantity)
3

13

## Simple Aggregations

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	NULL	february

`select count(quantity) from Purchase`



count(DISTINCT quantity)
3

14

## Counting Duplicates

COUNT applies to duplicates, unless otherwise stated:

`SELECT count(product)` same as `count(*)` if no nulls  
`FROM Purchase`  
`WHERE price > 4.99`

We probably want:

`SELECT count(DISTINCT product)`  
`FROM Purchase`  
`WHERE price > 4.99`

CSE 414 - Autumn 2018

15

## More Examples

What do they mean ?

`SELECT SUM(P.price * P.quantity)`  
`FROM Purchase AS P`

`SELECT SUM(P.price * P.quantity)`  
`FROM Purchase AS P`  
`WHERE P.product = 'bagel'`

CSE 414 - Autumn 2018

16

## Grouping and Aggregation

`Purchase(product, price, quantity)`

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

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	70

CSE 414 - Autumn 2018

17

## 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	70

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

18

## Grouping and Aggregation

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

Product
Bagel
Banana

```
SELECT product
FROM Purchase
GROUP BY product
```

19

## Grouping and Aggregation

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

Product
Bagel
Banana

```
SELECT product
FROM Purchase
GROUP BY product
```

20

## Grouping and Aggregation

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

Product
Bagel
Banana

Price	Quantity
3	20
1.50	20

Price	Quantity
0.5	50
2	10
4	10

Intermediate  
collections

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

21

## Grouping and Aggregation

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

Product
Bagel
Banana

Price	Quantity
3	20
1.50	20

Price	Quantity
0.5	50
2	10
4	10

Intermediate  
collections

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

22

## Grouping and Aggregation

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

Product
Bagel
Banana

Price	Quantity
3	20
1.50	20

Price	Quantity
0.5	50
2	10
4	10

Intermediate  
collections

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

23

## Remember: Simple Aggregate

pid	product	price	quantity	month
1	bagel	1.99	20	september
2	bagel	2.5	12	december
3	banana	0.99	9	september
4	banana	1.59	9	february

```
select sum(quantity) from Purchase
```



sum(quantity)
50

24

### Grouping and Aggregation

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

Intermediate collections

Product	SUM(quantity)
Bagel	40
Banana	70

```

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

25

### Grouping and Aggregation

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

Intermediate collections

Product	SUM(quantity)
Bagel	40
Banana	70

```

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

26

### 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	70

```

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

27

### 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?

CSE 414 - Autumn 2018 28

### 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

CSE 414 - Autumn 2018 29

### 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

```

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

CSE 414 - Autumn 2018 30

### 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

CSE 414 - Autumn 2018 31

### Need to be Careful...

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

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

**NOT FIRST NORMAL FORM!**

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	??

CSE 414 - Autumn 2018 32

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

### Need to be Careful...

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

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

**NOT FIRST NORMAL FORM!**

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	??

CSE 414 - Autumn 2018 33

### 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?

CSE 414 - Autumn 2018 34

### Grouping and Aggregation

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
```

### 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
```

## 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
```

Rows where price > 1 are removed, so 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™

CSE 414 - Autumn 2018

39

## 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

WHERE price > 1

```
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
```

## 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

CSE 414 - Autumn 2018

42

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

## Ordering and SQLite LIMIT

Useful keyword:

LIMIT N

constrains output to N tuples

```
SELECT product, sum(price*quantity) as rev
FROM Purchase
GROUP BY product
ORDER BY rev desc
LIMIT 5
```

Often use for "top 5" type queries

CSE 414 - Autumn 2018

43

## Filtering Groups

FWGOS

If the **WHERE** filter comes before **GROUP BY**,  
Need some way to filter after forming groups

CSE 414 - Autumn 2018

44

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.

CSE 414 - Autumn 2018

45

## General form of Grouping and Aggregation

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

Why ?

S = may contain attributes  $a_1, \dots, a_k$  and/or any  
aggregates but **NO OTHER ATTRIBUTES**

C1 = is any condition on the attributes in  $R_1, \dots, R_n$

C2 = is any condition on aggregate expressions  
and on attributes  $a_1, \dots, a_k$

CSE 414 - Autumn 2018

46

## 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 Semantics
2. Group by the attributes  $a_1, \dots, a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

CSE 414 - Autumn 2018

47

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"

CSE 414 - Autumn 2018

48



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
```

CSE 414 - Autumn 2018

49

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
```

CSE 414 - Autumn 2018

50

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
```

CSE 414 - Autumn 2018

51

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
```

CSE 414 - Autumn 2018

52

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)
```

CSE 414 - Autumn 2018

53

## 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

CSE 414 - Autumn 2018

54

Product(pid,pname,manufacturer)  
Purchase(id,product\_id,price,month)

## Aggregate + Join

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

CSE 414 - Autumn 2018 57

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 Product, price is in Purchase...

CSE 414 - Autumn 2018 58

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 Product, price is in Purchase...

```
-- 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

CSE 414 - Autumn 2018 59

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 Product, price is in Purchase...

```
-- 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
...	60

CSE 414 - Autumn 2018 60

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
...		61

CSE 414 - Autumn 2018 61

FWGHOS

## 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
```

Count(\*) is never 0

CSE 414 - Autumn 2018 62

## Including Empty Groups

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

Product			Purchase		
pname	manufacturer	...	product	price	...
Gizmo	GizmoWorks		Camera	150	
Camera	Canon		Camera	300	
OneClick	Hitachi		OneClick	180	

Join(Product, Purchase)					
pname	manu	facturer	...	manu	facturer
Camera	Canon	Canon	Canon	150	
Camera	Canon	Canon	Canon	300	
OneClick	Hitachi	Hitachi	Hitachi	180	

Final results	
manufacturer	Count(*)
Canon	2
Hitachi	1

No GizmoWorks!

63

## 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

CSE 414 - Autumn 2018

64

## 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
```

Product			Purchase		
pname	manufacturer	...	product	price	...
Gizmo	GizmoWorks		Camera	150	
Camera	Canon		Camera	300	
OneClick	Hitachi		OneClick	180	

Left Outer Join(Product, Purchase)					
pname	manufacturer	...	product	price	...
Camera	Canon		Camera	150	
Camera	Canon		Camera	300	
OneClick	Hitachi		OneClick	180	
Gizmo	GizmoWorks		NULL	NULL	NULL

Why 0 for GizmoWorks?

Final results	
manufacturer	Count(y.pid)
Canon	2
Hitachi	1
GizmoWorks	0

GizmoWorks is paired with NULLs

65

## Including Empty Groups

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

Product			Purchase		
pname	manufacturer	...	product	price	...
Gizmo	GizmoWorks		Camera	150	
Camera	Canon		Camera	300	
OneClick	Hitachi		OneClick	180	

Left Outer Join(Product, Purchase)					
pname	manufacturer	...	product	price	...
Camera	Canon		Camera	150	
Camera	Canon		Camera	300	
OneClick	Hitachi		OneClick	180	
Gizmo	GizmoWorks		NULL	NULL	NULL

Final results	
manufacturer	Count(*)
Canon	2
Hitachi	1
GizmoWorks	1

Probably not what we want!

66