Updated 1/23/2021

Topic 1: Data models and query languages
Unit 1: SQL (a refresher)
Lecture 1

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp21)

https://northeastern-datalab.github.io/cs7240/sp21/

1/19/2021

Topic 1: Data models and query languages

PRELIMINARY

- **Lecture 1 (Tue 1/19):** Course introduction, SQL refresher, PostgreSQL setup, SQL files
- **Lecture 2 (Fri 1/22):** SQL continued, Logic & relational calculus
- **Lecture 3 (Tue 1/26):** Relational calculus & relational algebra
- **Lecture 4 (Fri 1/29):** Relational algebra & Codd's theorem, Datalog
- **Lecture 5 (Tue 2/2):** Datalog and more expressive variations
- **Lecture 6 (Fri 2/5): (A1 due)** Datalog vs. stable model semantics
- **Lecture 7 (Tue 2/9):** Datalog evaluation strategies, NoSQL

Pointers to relevant concepts & supplementary material:

- **1. SQL**: SQL refresher [SAMS'12], [Cow'03] Ch3 & Ch5, [Complete'08] Ch6
- **2. Logic**: First-order logic, relational calculus: [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'10] Ch6.2 & Ch6.3 [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4
- **3. Algebra**: Relational algebra, Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'10] Ch6.1, [Alice'95] Ch4.4 & Ch5.4
- **4. Datalog**: Datalog, stable model semantics: [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [Gatterbauer, Suciu'10]
- **5. Data models**: Alternative data models, NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Source: https://northeastern-datalab.github.io/cs7240/sp21/

question @6

stop following 3 views

Actions

Class illustration sign-up sheet

Please sign-up to be the illustrator for 3-5 lectures (depending on the number of students in class) by first copying this template below and create one single "student answer" in Piazza that all students can then edit (https://trunkuserguide.screenstepslive.com/s/5891/m/18197/l/195294-how-do-students-respond-to-other-student-s-questions-in-piazza)

Notice that we may have lectures that continue the topic from a previous lecture. In those cases it is best for the scribes to work together on a single document for two lectures (we will discuss those situations in class).

L1: 1/19
L2: 1/22
L3: 1/26
L4: 1/29
L5: 2/2
L6: 2/5
L7: 2/9
L8: 2/12
L9: 2/16
L10: 2/19

L11: 2/23
L12: 2/26
L13: 3/2
L14: 3/5
L15: 3/9
L16: 3/12
L17: 3/16
L18: 3/19
L19: 3/23
L20: 3/26

L21: 3/30
L22: 4/2
L23: 4/6
L24: 4/9
L25: 4/13
L26: 4/16
L27: 4/14: Project presentations (no lecture)
L28: 4/17: Project presentations (no lecture)

#pin

logistics

edit · good question | 0

Updated 1 day ago by Wolfgang Gatterbauer

the instructors' answer, *where instructors collectively construct a single answer*

Click to start off the wiki answer

followup discussions *for lingering questions and comments*

Start a new followup discussion

Compose a new followup discussion

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins
  - Top-k

Structured Query Language: SQL

- Influenced by relational calculus (= First Order Logic)
- SQL is a declarative query language
  - We say what we want to get
  - We don’t say how we should get it ("separation of concerns")

Simple SQL Query

Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

302

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```

?

Simple SQL Query

Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

302

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```

PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Selection
& Projection

Selection vs. Projection

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```

Where does the selection happen?

?

PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Selection & Projection

Selection vs. Projection

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```

PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

One **selects** certain entires=tuples (rows)
-> happens in the **WHERE** clause
-> acts like a **filter**

Selection vs. Projection

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

One **projects** onto some attributes (columns)
-> happens in the **SELECT** clause

One **selects** certain entires=tuples (rows)
-> happens in the **WHERE** clause
-> acts like a **filter**

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```

PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
PowerGizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT category
FROM   Product
```

⇨ ?

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
PowerGizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Set vs. Bag semantics

```
SELECT category
FROM   Product
```

⇨

Category
Gadgets
Gadgets
Photography
Household

X : 0
G : 2
P : 1
H : 1

?

⇨

Category
Gadgets
Photography
Household

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
PowerGizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Set vs. Bag semantics

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

```
SELECT DISTINCT category
FROM   Product
```

Category
Gadgets
Photography
Household

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins
  - Top-k

Keys and Foreign Keys

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is here
a key vs.
a foreign key?

?

Keys and Foreign Keys

Key

Foreign key

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Key

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

Foreign key: attribute in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

?

Foreign key: attribute in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

?

Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');

SuperTouch	$249.99	Computer	NewCom

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

tuple violates foreign key constraint

SuperTouch	$249.99	Computer	NewCom

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');

Gizmo	$14.99	Gadgets	Hitachi

tuple violates foreign key constraint

Foreign key: attribute in a relational table that matches a candidate key of another table

Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');

SuperTouch	$249.99	Computer	NewCom

Delete from Company where CName = 'Canon';

?

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

tuple violates foreign key constraint

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

SuperTouch	$249.99	Computer	NewCom

```
Delete from Company
where CName = 'Canon';
```

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins
  - Top-k

Joins

302

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

?

Joins

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

```
SELECT  pName, price
FROM    Product, Company
WHERE   manufacturer=cName
  and   country='Japan'
  and   price <= 200
```

Join b/w Product and Company

PName	Price
SingleTouch	$149.99

Meaning (Semantics) of conjunctive SQL Queries

```
SELECT  a1, a2, …, ak
FROM    R1 as x1, R2 as x2, …, Rn as xn
WHERE   Conditions
```

Conceptual evaluation strategy (nested for loops):

```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        …..
            for xn in Rn do
                if Conditions
                    then Answer = Answer ∪ {(a1,…,ak)}
return Answer
```

Meaning (Semantics) of conjunctive SQL Queries

R1

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

R2 C

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

R3 P

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
1		30	

```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        .....
            for xn in Rn do
                if Conditions
                    then Answer = Answer ∪ {(a1,…,ak)}
return Answer
```

Meaning (Semantics) of conjunctive SQL Queries

R1

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

R2

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

R3

att_1	att_2	...	att_k
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        …..
            for xn in Rn do
                if Conditions
                    then Answer = Answer ∪ {(a1,…,ak)}
return Answer
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - FROM: Compute the cross-product of relation-list.
  - WHERE: Discard resulting tuples if they fail qualifications ("select" the rest)
  - SELECT: Delete attributes that are not in target-list.
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute the same answers.

- Quiz: we say “semantics” not “execution order”. Why?

?

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - FROM: Compute the cross-product of relation-list.
  - WHERE: Discard resulting tuples if they fail qualifications ("select" the rest)
  - SELECT: Delete attributes that are not in target-list.
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute the same answers.

- Quiz: we say “semantics” not “execution order”. Why?
  - The preceding slides show what a join means (semantics = meaning)
  - Not actually how the DBMS executes it under the covers (separation of concerns)

Using the Formal Semantics

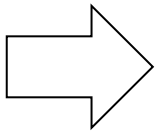
305

R(a), S(a), T(a)

R	S	T
a	a	a
1	1	2
2		

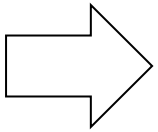
What do these queries compute?

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```



?

```
SELECT  R.a
FROM    R, S, T
WHERE   R.a=S.a
    or  R.a=T.a
```



?

Using the Formal Semantics

305

R(a), S(a), T(a)

R	S	T
a	a	a
1	1	2
2		

What do these queries compute?

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```

a
1

Returns R ∩ S (intersection)

```
SELECT  R.a
FROM    R, S, T
WHERE   R.a=S.a
     or R.a=T.a
```

?

Using the Formal Semantics

305

R(a), S(a), T(a)

What do these queries compute?

R	S	T
a	a	a
1	1	2
2		

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```

a
1

Returns $R \cap S$ (intersection)

```
SELECT  R.a
FROM    R, S, T
WHERE   R.a=S.a
     or R.a=T.a
```

a
1
2

Returns $R \cap (S \cup T)$
if $S \neq \emptyset$ and $T \neq \emptyset$

Using the Formal Semantics

R(a), S(a), T2(a)

R	S	T2
a	a	a
1	1	~~2~~
2		

What do these queries compute?

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```

a
1

Returns R ∩ S (intersection)

?

Next, we are removing the input tuple "(2)"

```
SELECT  R.a
FROM    R, S, T2 as T
WHERE   R.a=S.a
     or R.a=T.a
```

a
1
2

Returns R ∩ (S ∪ T) if S ≠ ∅ and T ≠ ∅

?

Using the Formal Semantics

R(a), S(a), T2(a)

R	S	T2
a	a	a
1	1	~~2~~
2		

What do these queries compute?

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```

a
1

Returns R ∩ S (intersection)

Next, we are removing the input tuple "(2)"

```
SELECT  R.a
FROM    R, S, T2 as T
WHERE   R.a=S.a
     or R.a=T.a
```

a
1
2

Returns R ∩ (S ∪ T) if S ≠ ∅ and T ≠ ∅

?

Using the Formal Semantics

R(a), S(a), T2(a)

R	S	T2
a	a	a
1	1	~~2~~
2		

What do these queries compute?

```
SELECT  R.a
FROM    R, S
WHERE   R.a=S.a
```

a
1

Returns R ∩ S (intersection)

Next, we are removing the input tuple "(2)"

```
SELECT  R.a
FROM    R, S, T2 as T
WHERE   R.a=S.a
   or   R.a=T.a
```

a

Returns Ø if S = Ø or T = Ø

Can seem counterintuitive! But remember conceptual evaluation strategy: Nested loops. If one table is empty -> no looping

Illustration with Python

305

```python
'''
Created on 3/23/2015
Illustrates nested Loop Join in SQL
__author__ = 'gatt'
'''

print "--- 1st nested loop ---"
for i in xrange(2):
    for j in xrange(3):
        for k in xrange(2):
            print "i=%d, j=%d, k=%d: " % (i, j, k),
            if i == j or i == k:
                print "TRUE",
            print

print "\n--- 2nd nested loop ---"
for i in xrange(2):
    for j in xrange(3):
        for k in xrange(1):
            print "i=%d, j=%d, k=%d: " % (i, j, k),
            if i == j or i == k:
                print "TRUE",
            print

print "\n--- 3rd nested loop ---"
for i in xrange(2):
    for j in xrange(3):
        for k in xrange(0):
            print "i=%d, j=%d, k=%d: " % (i, j, k),
            if i == j or i == k:
                print "TRUE",
            print
```

```
/Library/Frameworks/Python.framework/Versio
--- 1st nested loop ---
i=0, j=0, k=0:  TRUE
i=0, j=0, k=1:  TRUE
i=0, j=1, k=0:  TRUE
i=0, j=1, k=1:
i=0, j=2, k=0:  TRUE
i=0, j=2, k=1:
i=1, j=0, k=0:
i=1, j=0, k=1:  TRUE
i=1, j=1, k=0:  TRUE
i=1, j=1, k=1:  TRUE
i=1, j=2, k=0:
i=1, j=2, k=1:  TRUE

--- 2nd nested loop ---
i=0, j=0, k=0:  TRUE
i=0, j=1, k=0:  TRUE
i=0, j=2, k=0:  TRUE
i=1, j=0, k=0:
i=1, j=1, k=0:  TRUE
i=1, j=2, k=0:

--- 3rd nested loop ---

Process finished with exit code 0
```

The comparison gets never evaluated

"Premature optimization is the root of all evil."
Donald Knuth (1974)

"When you are diagnosing problems, don't think about how you will solve them—just diagnose them. Blurring the steps leads to suboptimal outcomes because it interferes with uncovering the true problems."
Ray Dalio (Principled, 2017)

Quiz


302

Our colorful hands represent "team exercises"

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM
WHERE
```

Quiz


302

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and P.price < 20
    and P.price > 25
    and P.manufacturer = cName
```

What about this query?

?

Quiz


302

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and P.price < 20
    and P.price > 25
    and P.manufacturer = cName
```

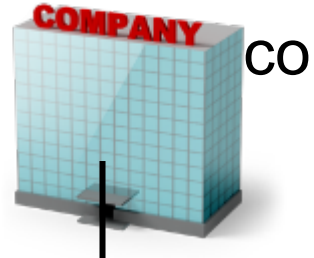
Wrong! Gives empty result: There is no product with price <20 and >25

Quiz

Product (<u>pName</u>, price, category, manufacturer)
Company (<u>cName</u>, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

C



country='USA'

P

P.price < 20 and
P.price > 25

What do we actually want?

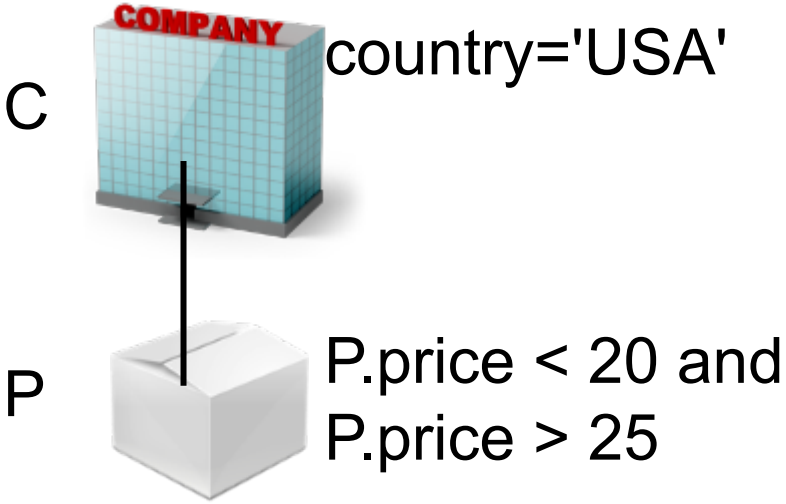
?

not possible!
→ Empty result

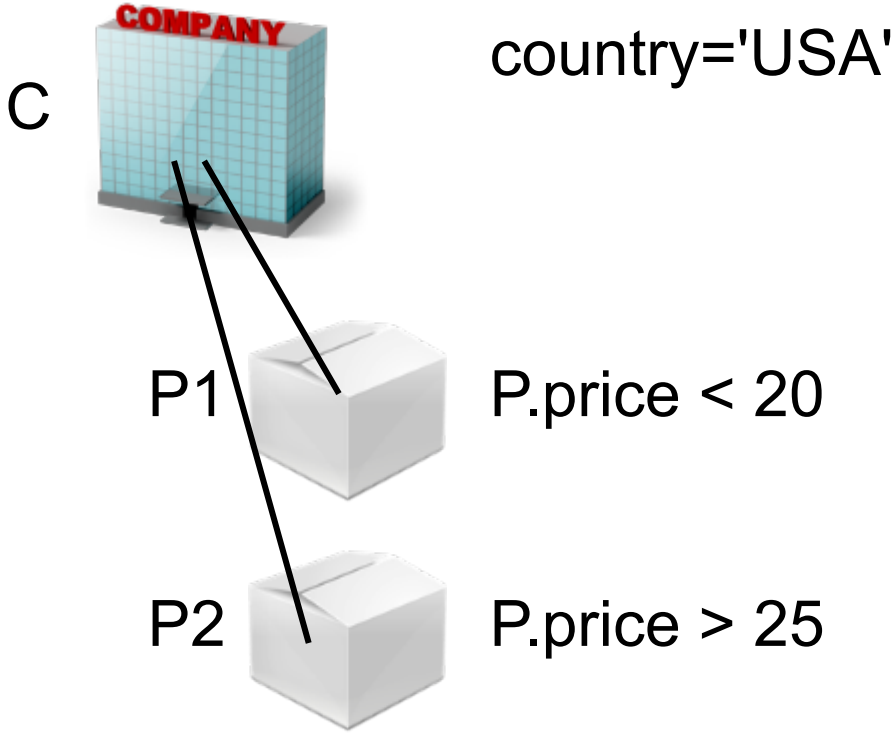
Quiz

Product (<u>pName</u>, price, category, manufacturer)
Company (<u>cName</u>, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.



not possible!
→ Empty result



Quiz


302

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and (P.price < 20
    or  P.price > 25)
    and P.manufacturer = cName
```

What about this query?

?

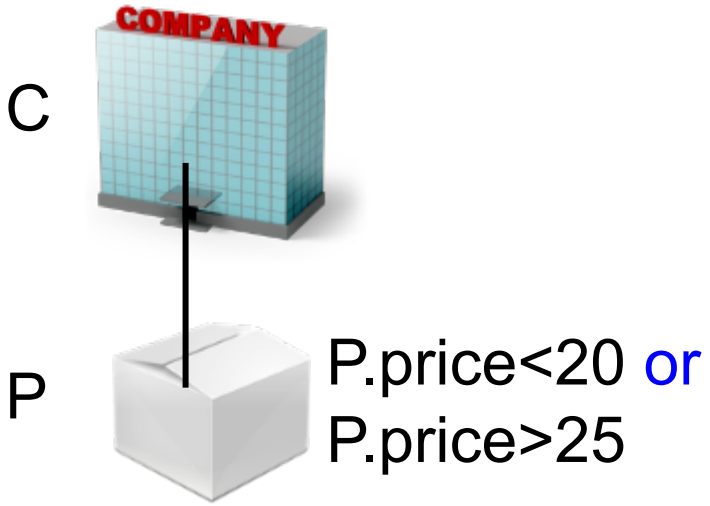
Quiz

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and (P.price < 20
    or  P.price > 25)
    and P.manufacturer = cName
```

Returns companies with single product w/price (<20 or >25)

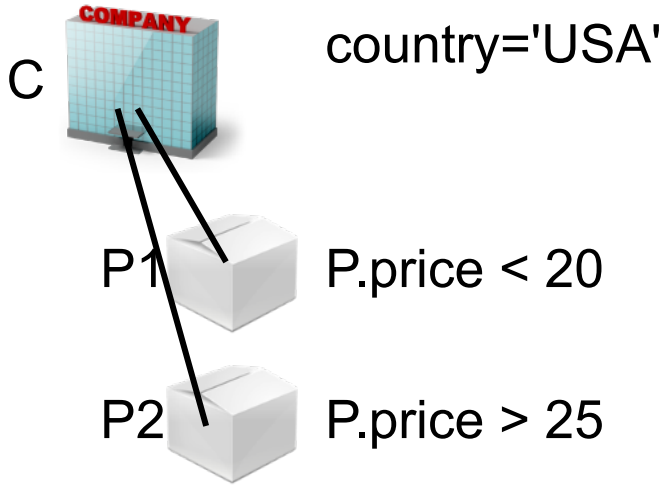


Quiz response: Self-join


Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P1, Product as P2, Company
WHERE   country = 'USA'
    and P1.price < 20
    and P2.price > 25
    and P1.manufacturer = cName
    and P2.manufacturer = cName
```



Quiz response: Self-join

P1

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
...	...	...	...

P2

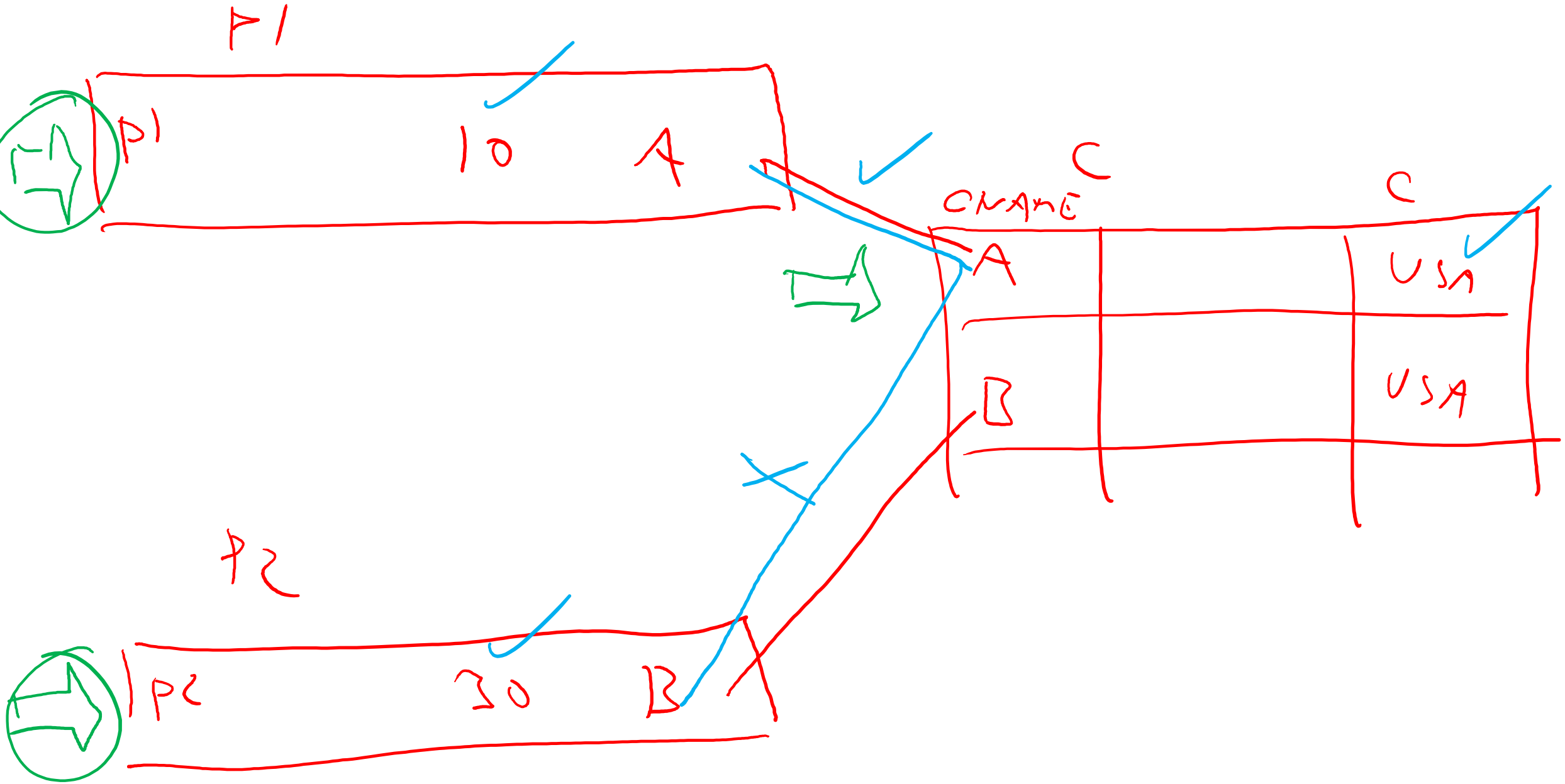
PName	Price	Category	Manufacturer
...	...	...	...
Powergizmo	$29.99	Gadgets	GizmoWorks

Company

CName	StockPrice	Country
GizmoWorks	25	USA
...	...	...

```
SELECT  DISTINCT cName
FROM    Product as P1, Product as P2, Company
WHERE   country = 'USA'
    and P1.price < 20
    and P2.price > 25
    and P1.manufacturer = cName
    and P2.manufacturer = cName
```

Cname
GizmoWorks


P1
P1
10
A
P2
P2
30
B
C
CNAME
C
A
USA
B
USA

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins
  - Top-k

Grouping and Aggregation

308

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10

?

For each product, find total quantities (sum of quantities) for all purchases with price over $1.

Grouping and Aggregation

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10

Product	TotalQuantities
Bagel	?
Banana	?

For each product, find total quantities (sum of quantities) for all purchases with price over $1.

Grouping and Aggregation

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
~~Banana~~	~~1~~	~~50~~
Banana	2	10
Banana	4	10

Product	TotalQuantities
Bagel	40
Banana	20

For each product, find total quantities (sum of quantities) for all purchases with price over $1.

From → Where → Group By → Select

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
~~Banana~~	~~1~~	~~50~~
Banana	2	10
Banana	4	10

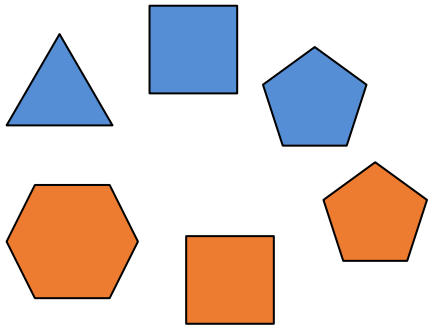
Product	TotalQuantities
Bagel	40
Banana	20

Select contains
- grouped attributes
- and aggregates

```
4  SELECT     product, sum(quantity) as TotalQuantities
1  FROM       Purchase  P
2  WHERE      price > 1
3  GROUP BY   product
```

Groupings illustrated with colored shapes

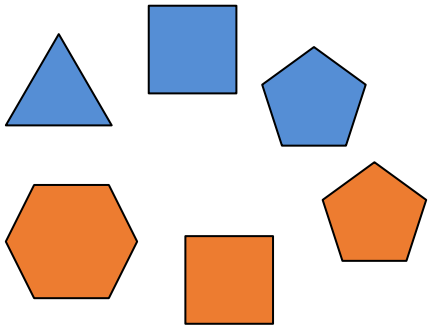
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

⇨ ?

group by numc (# of corners)



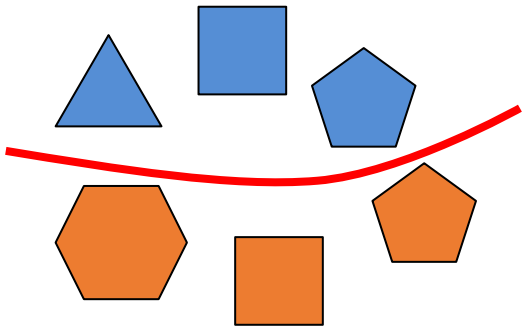
```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

⇨ ?

Groupings illustrated with colored shapes

color	anc
blue	3
blue	4
blue	5
orange	4
orange	5
orange	6

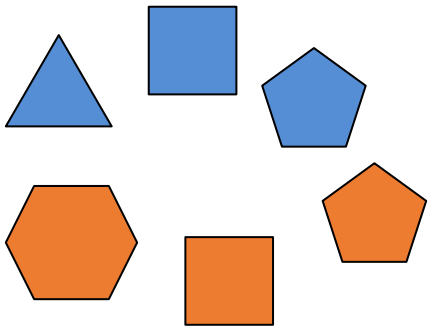
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

⇨ ?

group by numc (# of corners)



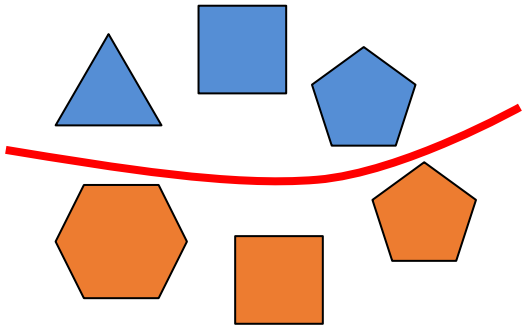
```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

⇨ ?

Groupings illustrated with colored shapes

color	anc
blue	3
blue	4
blue	5
orange	4
orange	5
orange	6

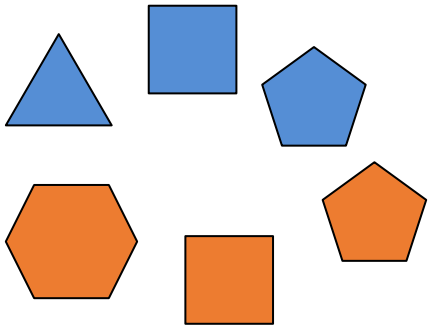
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

color	anc
blue	4
orange	5

group by numc (# of corners)



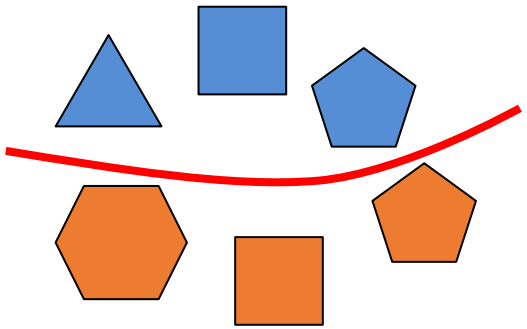
```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

?

Groupings illustrated with colored shapes

color	anc
blue	3
blue	4
blue	5
orange	4
orange	5
orange	6

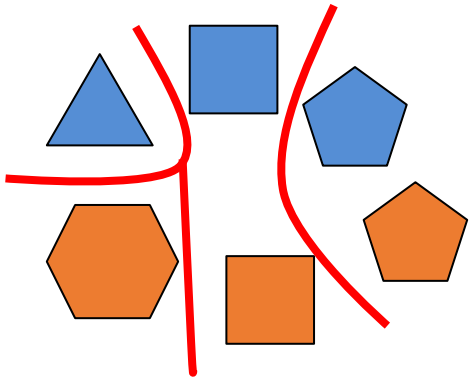
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

color	anc
blue	4
orange	5

group by numc (# of corners)



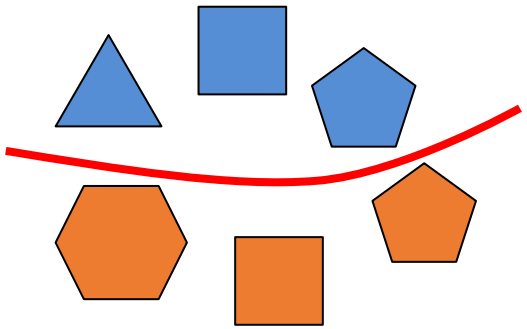
```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

numc
3
4
5
6

Groupings illustrated with colored shapes

color	anc
blue	3
blue	4
blue	5
orange	4
orange	5
orange	6

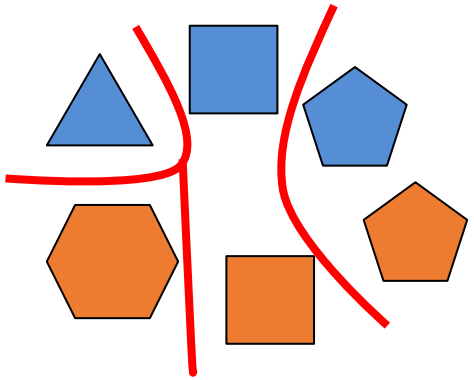
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

color	anc
blue	4
orange	5

group by numc (# of corners)



```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

numc
3
4
5
6

Same as:

```
SELECT  DISTINCT numc
FROM    Shapes
```