The Unreasonable Effectiveness of the the join operator.

Antonio Penta,PhD

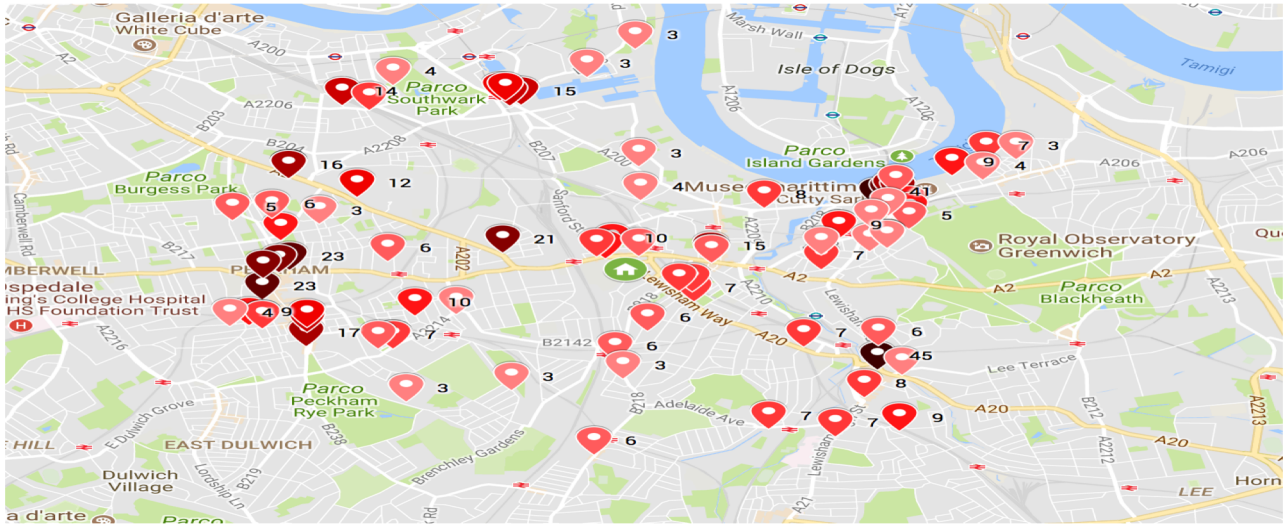
(*) E. Wigner, “The Unreasonable Effectiveness of Mathematics in the Natural Sciences,” Comm. Pure and Applied Mathematics, vol. 13, no. 1, 1960, pp. 1–14.)

TABLE OF CONTENTS

TEACHING OBJECTIVES

Compute the number of crimes reported in the same area of the pubs around University of Goldsmiths.

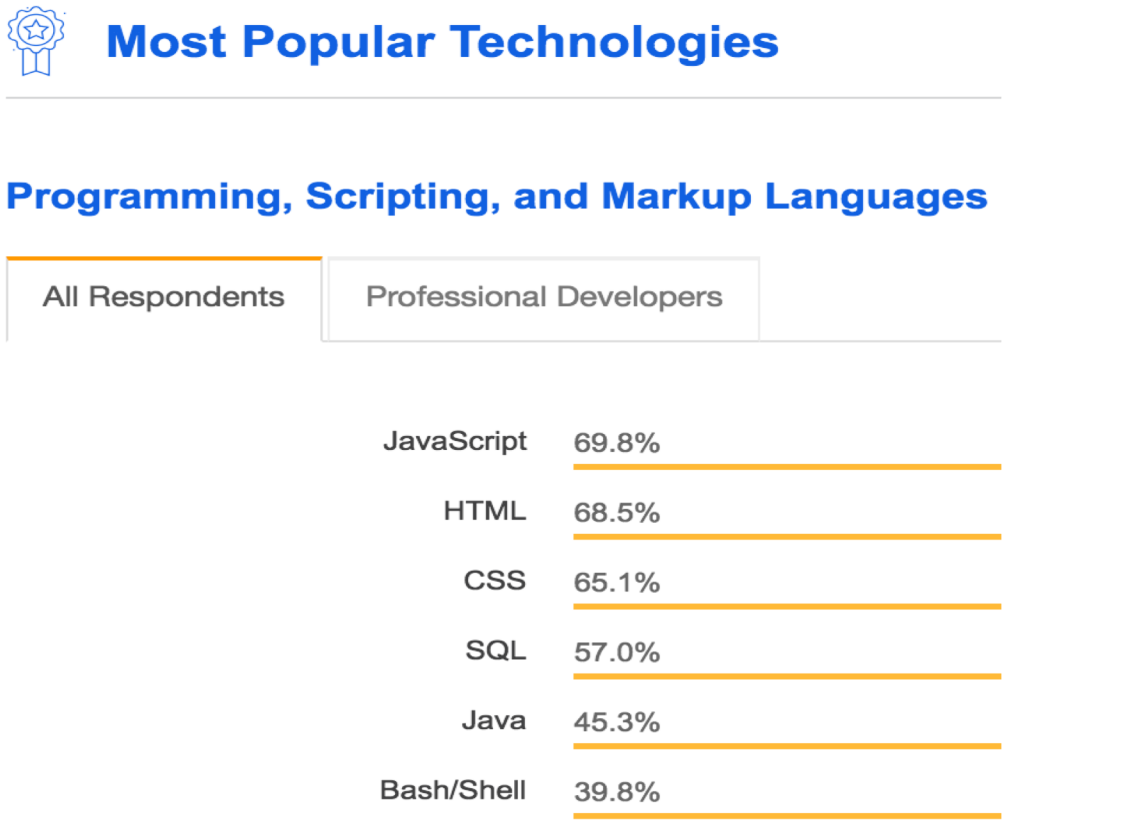
```
select p.name_pub,p.latitude,p.longitude, count(*) as total_crime
from crimes_around_goldsmiths as c, pubs_around_goldsmiths as p
where distance(p.latitude,p.longitude,c.latitude,c.longitude)<0.1
group by (p.name_pub,p.latitude,p.longitude)
having (count(*)>2) order by total_crime desc;
```

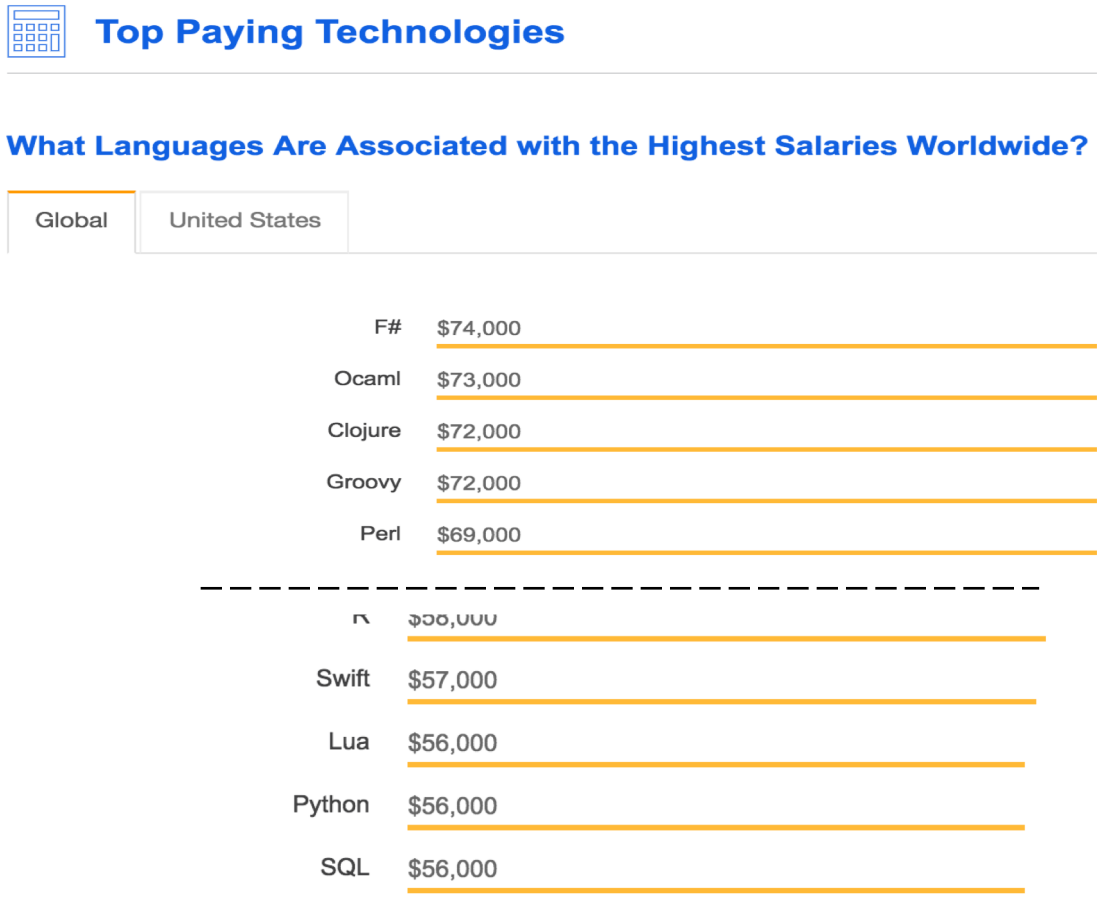


Crime Rate (April 2018) around pubs within the area (<3km) of Goldsmiths University

WHY LEARNING THE JOIN

- Most of the applications are dealing with multiple and very large tables
- LENEL Security
  - Data Of Paris Shops > more than 100 tables
- Accenture Customer Analytics:
  - Anthem Health Insurance -> more than 50 tables





JOIN
(THE PROBLEM)

Find names and prices of all the products under $200 manufactured in Japan;

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Let us suppose that “Manufacturer” and “CName” are the “same”

JOIN
(WHAT WE WOULD LIKE TO DO)

Find names and prices of all the products under $200 manufactured in Japan;

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Product

PName	Price	Category	Manuf
Gizmo	$19	Gadgets	GWorks
Powergizmo	$29	Gadgets	GWorks
SingleTouch	$149	Photography	Canon
MultiTouch	$203	Household	Hitachi

Company

Cname	Stock	Country
GWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

JOIN (SYNTAX)

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Ex: Find names and prices of all the products under $200 manufactured in Japan;

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName  AND Country='Japan'
       AND Price <= 200
```

Join Condition

JOIN – CARTESIAN PRODUCT (NO JOIN CONDITION)

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

```
SELECT PName, Price
FROM   Product, Company
```

Product

PName	Price	Category	Manuf
Gizmo	$19	Gadgets	GWorks
Powergizmo	$29	Gadgets	GWorks
SingleTouch	$149	Photography	Canon
MultiTouch	$203	Household	Hitachi

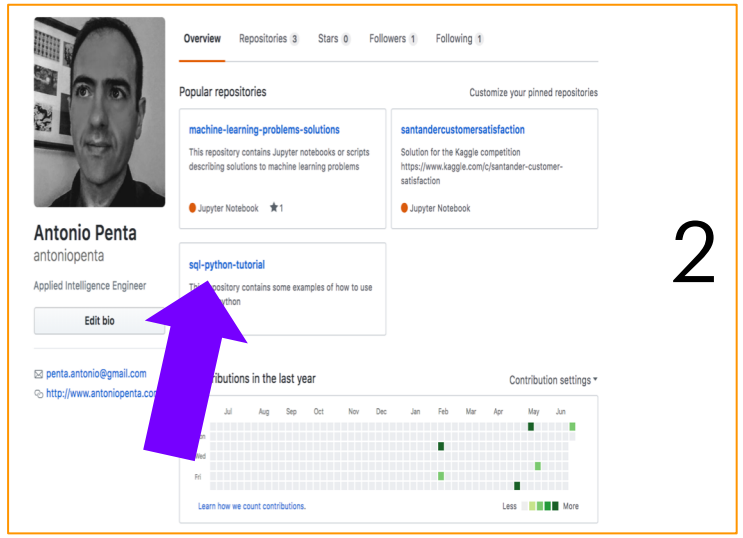
Company

Cname	Stock	Country
GWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

EXERCISE (SETTINGS)

Hypothesis : python 3.6 + WIFI

https://github.com/antoniopenta 1



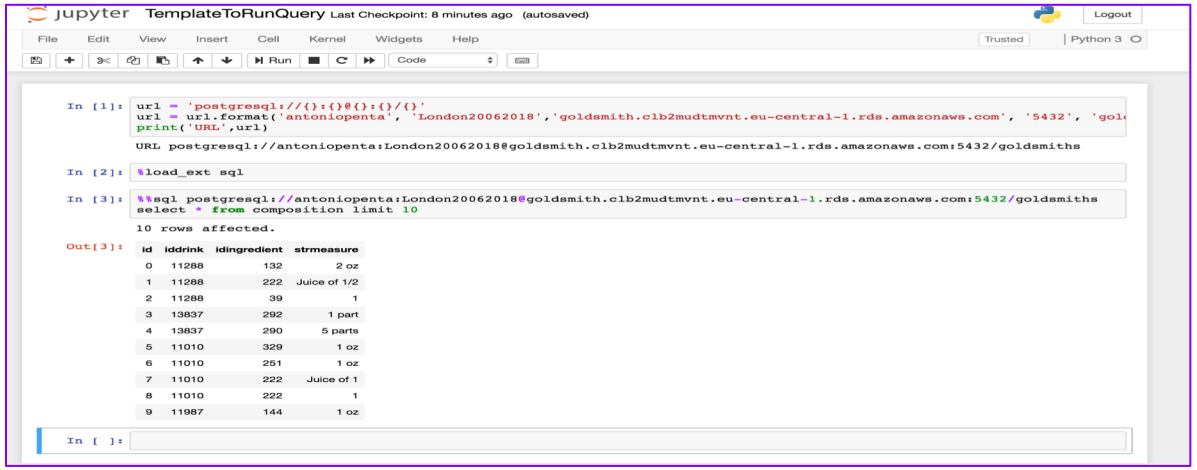
2

3

```
$cd sql-4-fun
$git clone https://github.com/antoniopenta/sql-python-tutorial.git
$cd sql-python-tutorial
$virtualenv sql
$ source sql/bin/activate
(sql)$ pip install -r requirements.txt
(sql)$ jupyter notebook
```

TemplateToRunQuery.ipynb

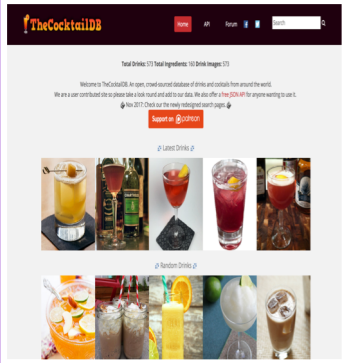
4



“COCKTAIL DATABASE”

Extraction Pipeline

https://www.thecocktaildb.com/api.php



JSON → jupyter (ScrapingCocktailDatasets.ipynb) → CSV → PostgreSQL pgAdmin3 → amazon RDS

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE

Find the name of the cocktails with more ingredients.

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

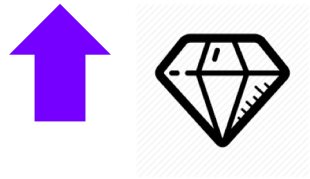
column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE-SOLUTION

Find the name of the cocktails with more ingredients.

- *select ck.strdrink,count(*) as total_ingredients from cocktail as ck, composition as co where ck.iddrink=co.iddrink group by ck.strdrink order by total_ingredients desc.*

strdrink character varying (100)	total_ingredients bigint
Angelica Liqueur	12
Egg Nog #4	11
Amaretto Liqueur	11
Arizona Twister	9
3-Mile Long Island Iced Tea	9
Masala Chai	9
Apple Cider Punch #1	8
Adam Bomb	8
Radioactive Long Island Ice...	8



Tips

Rename the table to make The query more readable.

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE

Find the most used ingredients in cocktails:

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

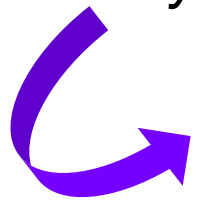
composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE-SOLUTION

Find the most used ingredients in cocktails:

- *select i.stringridient,count(*) as total_cocktails from ingredients as i, composition as co where co.idingredient=i.idingredient group by i.stringridient order by total_cocktails desc;*



stringridient character varying (100)	total_cocktails bigint
Gin	94
Vodka	89
Sugar	71
Orange juice	60
Lemon juice	51
Lemon	50
Ice	46
Light rum	44
Water	40
Amaretto	40
Grenadine	40

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

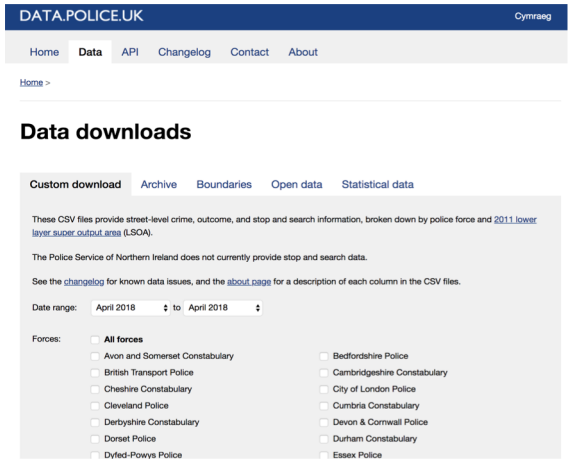
column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

PROBLEM AND DATA

Compute the number of crimes reported in the same area of the pubs around University of Goldsmiths.



column_name character varying	data_type character varying
id	bigint
crime_id	text
month	timestamp without ti...
reported_by	text
falls_within	text
longitude	double precision
latitude	double precision
location	text
lsoa_code	text
lsoa_name	text
crime_type	text
last_outcome_categ...	text
context	double precision

column_name character varying	data_type character varying
name_pub	character varying
street_name	character varying
postcode	character varying
latitude	double precision
longitude	double precision
borough	character varying

INITIAL EXPLORATION OF THE DATA

```
select crime_type,
latitude,longitude,
lsoa_name
from crime limit 5
```

crime_type text	latitude double precision	longitude double precision	lsoa_name text
Violence and sexual offe...	51.114954	0.955306	ashford
Anti-social behaviour	51.583427	0.140634	barking_and...
Anti-social behaviour	51.588063	0.134947	barking_and...
Anti-social behaviour	51.588063	0.134947	barking_and...
Anti-social behaviour	51.588063	0.134947	barking_and...

```
select name_pub,
latitude,longitude,
borough
from pubs limit 5
```

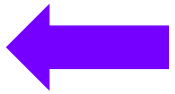
name_pub character varying (100)	latitude double precision	longitude double precision	borough character varying (100)
Aria Bar	51.54024	0.081962	Barking and Dagenham
Barking Arms Ltd	51.539531	0.080819	Barking and Dagenham
Barking football Club	51.545377	0.11296	Barking and Dagenham
Barking Indoor Bowls Club ...	51.542225	0.085096	Barking and Dagenham
Barking Rugby Union Footb...	51.53492	0.127357	Barking and Dagenham

SUB-PROBLEMS

Compute the number of crimes reported in the same area of the pubs around University of Goldsmiths.

- University of Goldsmiths in lat,lng -> PostCode (SE14 6NW) -> lat:51.47414, lng:-0.03540
- Around University of Goldsmiths -> <3km
- reported in the same area -> <200m
- connect crimes and pubs -> a distance between two latitude and longitude points
- Number of crimes: Group by crimes

- STEP 1 = find the crimes around Goldsmiths
- STEP 2= find the pubs around Goldsmiths
- STEP 3= combine crimes and pubs



Algorithm

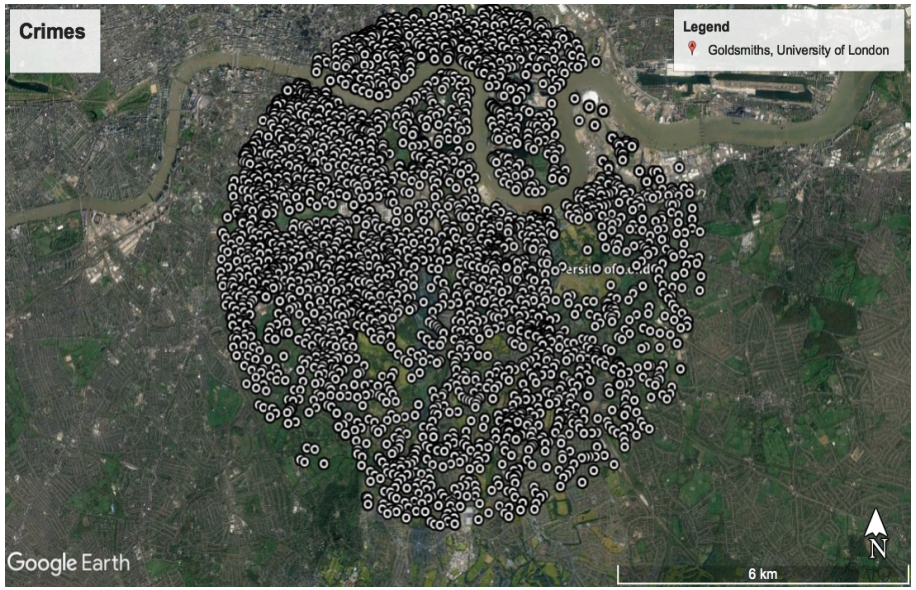
Distance function (in km) :

distance (latitude1, latitude2, longitude1, longitude2) =
*6371 * 2 * asin(sqrt(POWER(SIN((latitude1 – latitude2) * pi()/180 / 2),*
*2) + cos(latitude1 * pi()/180) *cos(latitude2 **
*pi()/180) * power(sin((longitude1 –longitude2) **
pi()/180 / 2), 2)))

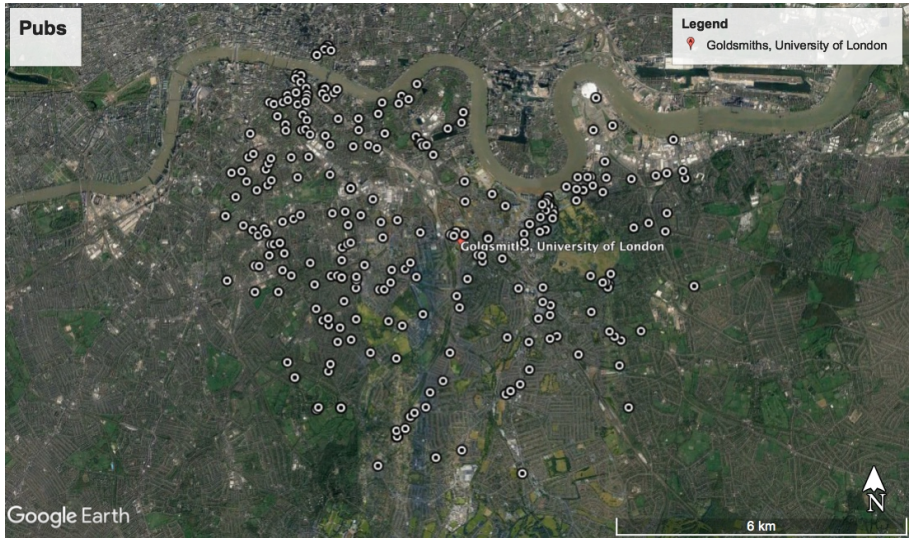
Python Code : CrimesRatesForPubsAroundGoldsmiths.ipynb

SOLUTION – STEPS 1-2

```
create view crimes_around_goldsmiths as (select *
from crimes as c where 6371 * 2 * asin(sqrt(
POWER(SIN((c.latitude - 51.47414) * pi()/180 / 2), 2) +
cos(c.latitude * pi()/180 ) * COS(51.47414 * pi()/180) *
power(sin((c.longitude +0.03540) * pi()/180 / 2), 2)
))<=3);
```




```
create view pubs_around_goldsmiths as (select *
from pubs as p where 6371 * 2 * asin(sqrt(
POWER(SIN((p.latitude - 51.47414) * pi()/180 / 2), 2) +
cos(p.latitude * pi()/180 ) * COS(51.47414 * pi()/180) *
power(sin((p.longitude +0.03540) * pi()/180 / 2), 2)
))<=3);
```

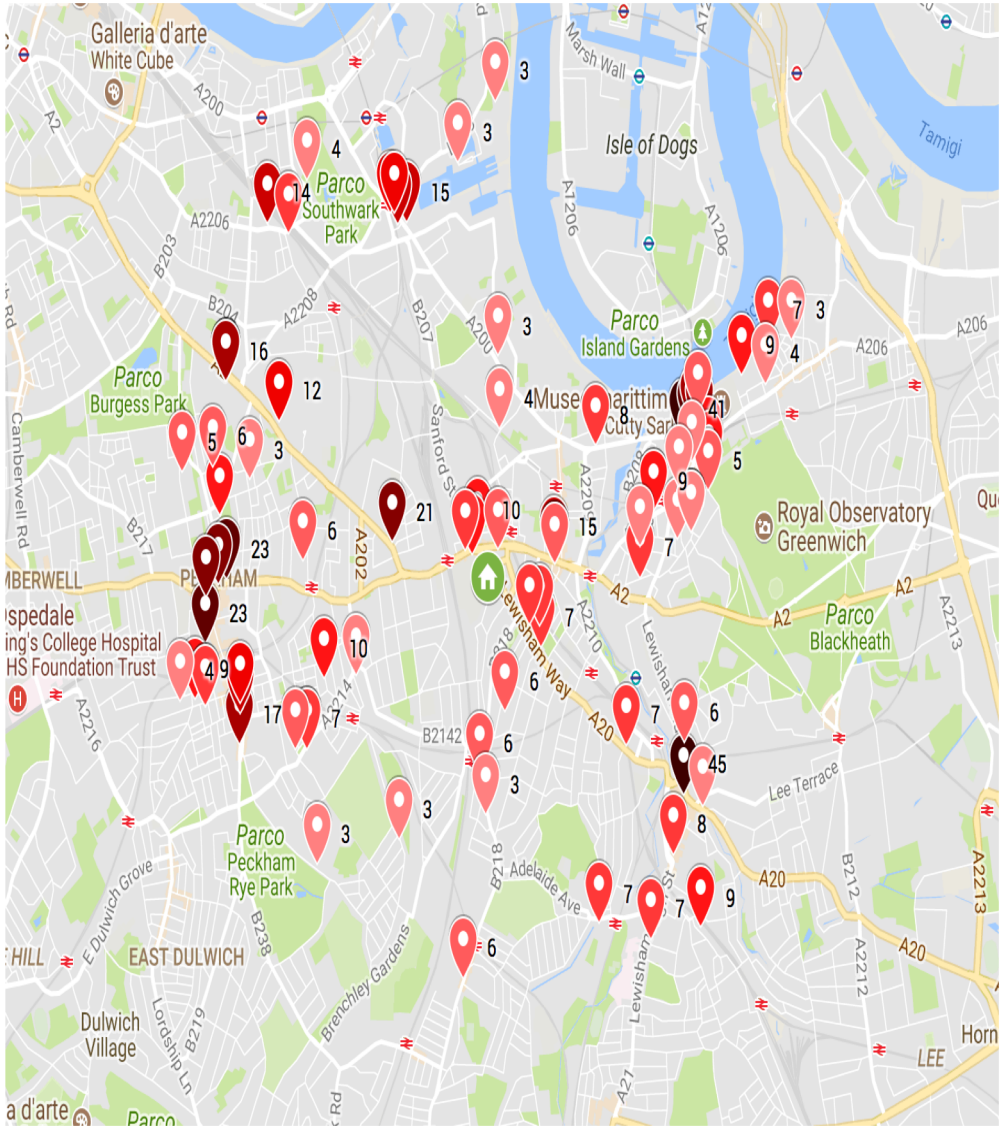


SOLUTION – STEP 3

Compute the number of crimes reported in the same area of the pubs around University of Goldsmiths

```
select p.name_pub,p.latitude,p.longitude, count(*) as total_crime
from crimes_around_goldsmiths as c,
pubs_around_goldsmiths as p
where 6371 * 2 * asin(sqrt( POWER(SIN((p.latitude - c.latitude) * pi()/180 / 2), 2) + cos(p.latitude * pi()/180 ) * COS(c.latitude * pi()/180) * power(sin((p.longitude - c.longitude) * pi()/180 / 2), 2) ))<0.1
group by (p.name_pub,p.latitude,p.longitude)
having (count(*)>2)
order by total_crime desc;
```

name_pub character varying (100)	latitude double precision	longitude double precision	total_crime bigint
Joiners Arms	51.462956	-0.010787	45
Up The Creek	51.481339	-0.010866	41
The Gipsy Moth	51.481708	-0.009856	27
The greyhound	51.474117	-0.068362	23
Peckham liberal club	51.470788	-0.071008	23



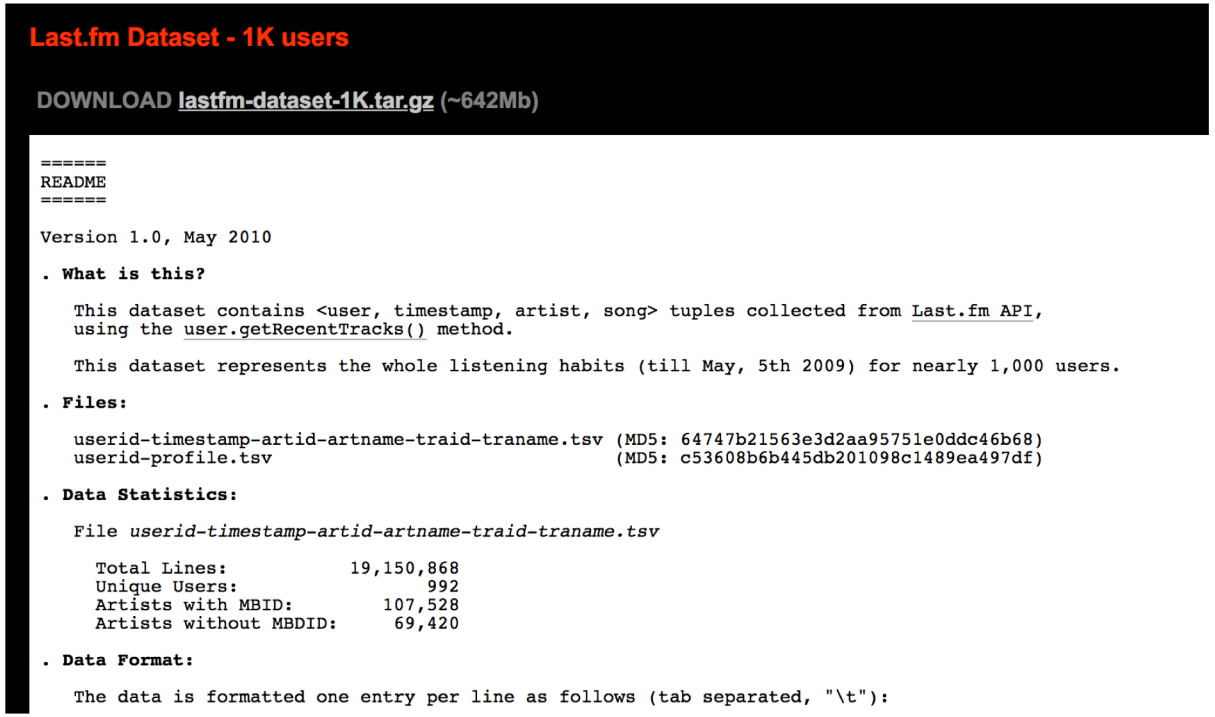
https://goo.gl/m1ZLNF

HOMEWORK

"Vanilla" Recommendation with Last.fm dataset.
https://goo.gl/RFVDLc

Problem
Find "new" artists by exploring the similarity between your profile and the ones stored in the database.

Tips multiple queries,
Similarity by country/age



Last.fm Dataset - 1K users

DOWNLOAD lastfm-dataset-1K.tar.gz (~642Mb)

```
======
README
======

Version 1.0, May 2010

. What is this?

   This dataset contains <user, timestamp, artist, song> tuples collected from Last.fm API,
   using the user.getRecentTracks() method.

   This dataset represents the whole listening habits (till May, 5th 2009) for nearly 1,000 users.

. Files:

   userid-timestamp-artid-artname-traid-traname.tsv (MD5: 64747b21563e3d2aa95751e0ddc46b68)
   userid-profile.tsv                               (MD5: c53608b6b445db201098c1489ea497df)

. Data Statistics:

   File userid-timestamp-artid-artname-traid-traname.tsv

     Total Lines:           19,150,868
     Unique Users:              992
     Artists with MBID:         107,528
     Artists without MBDID:     69,420

. Data Format:

   The data is formatted one entry per line as follows (tab separated, "\t"):
```

File: userid-timestamp-artid-artname-traid-traname.tsv
- userid \t timestamp \t musicbrainz-artist-id \t artist-name \t musicbrainz-track-id \t track-name

File: userid-profile.tsv
- userid \t gender ('m'|'f'|empty) \t age (int|empty) \t country (str|empty) \t signup (date|empty)

BK

FIRST QUERY

Table or Relations

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

Record or Tuple

Attribute

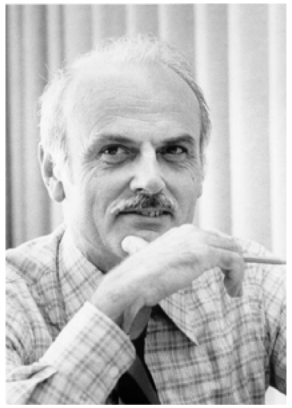
```
SELECT *
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks

```
SELECT <attributes|*>
FROM   <one or more relations>
WHERE  <conditions>
```

BASIC SYNTAX

```
SELECT <attributes|*>
FROM   <one or more
relations>
WHERE  <conditions>
```



Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of

Codd, E. F. (1970). "A relational model of data for large shared data banks" (PDF). *Communications of the ACM*. **13** (6): 377

```
SELECT *
FROM   Product
WHERE  Category = 'Gadgets'
```

SQL commands are case insensitive:

- Same: SELECT, Select, select
- Same: Product, product

Values are not:

- Different: 'London', 'london'

Use single quotes for constants:

- 'antonio' - yes
- "antonio" - no

WHAT CAN WE USE IN THE SELECT?

Multiple Attributes

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets'
```

Remove Duplicates

```
SELECT distinct (Manufacturer)
FROM   Product
WHERE  Category = 'Gadgets'
```

Aggregators:

- *COUNT* - return the number of rows
- *SUM* - cumulate the values
- *AVG* - return the average for the group
- *MIN / MAX* - smallest / largest value

```
SELECT count(*)
FROM   Product
WHERE  Category = 'Gadgets'
```

```
SELECT count(distinct Manufacturer)
FROM   Product
WHERE  Category = 'Gadgets'
```

```
SELECT avg(Price)
FROM   Product
WHERE  Category = 'Gadgets'
```

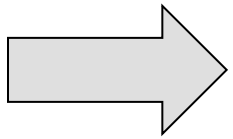
OPERATIONS IN A SELECT STATEMENT

How much money have you spent so far in cakes ??

Purchase

Product	Date	Price	Quantity
cakes	10/21	1	20
banana	10/3	0.5	10
banana	10/10	1	10
cakes	10/25	1.50	20

```
SELECT SUM(price * quantity)
FROM   Purchase
WHERE  product = 'cakes'
```



50 (= 1*20 + 1.50*20)

WHAT CAN WE DO WITH WHERE?

Express Boolean Conditions

```
SELECT Pname
FROM   Product
WHERE  Category = 'Gadgets' and
Price> 100
```

Compare the results of operation

```
SELECT Manufacturer
FROM   Product
WHERE Price/1000> 0.5
```

String Pattern Matching

```
SELECT *
FROM   Products
WHERE  PName LIKE '%gizmo%'
```

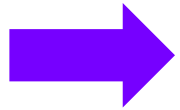
s **LIKE** p: pattern matching on strings
p may contain two special symbols:
% = any sequence of characters
_ = any single character

GROUP BY

Find total sales per product?

Purchase

Product	Date	Price	Quantity
cakes	10/21	1	20
banana	10/3	0.5	10
banana	10/10	1	10
cakes	10/25	1.50	20



Product	Total Sales
cakes	50
banana	15

Tips :
Rename the column

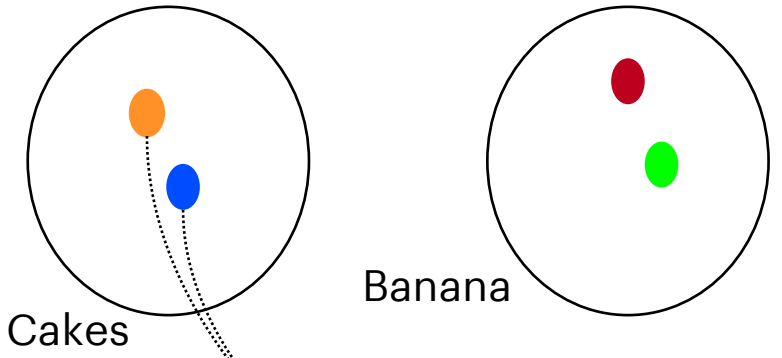
```
SELECT    product,
          SUM(price * quantity) AS TotalSales
FROM      Purchase
GROUP BY  product
```

UNDERSTANDING GROUP BY (THE SEMANTIC OF THE QUERY)

1. Compute the FROM (WHERE clauses)

Product	Date	Price	Quantity
cakes	10/21	1	20
banana	10/3	0.5	10
banana	10/10	1	10
cakes	10/25	1.50	20

2. Group by the attributes in the GROUP BY



SUM(price * quantity)

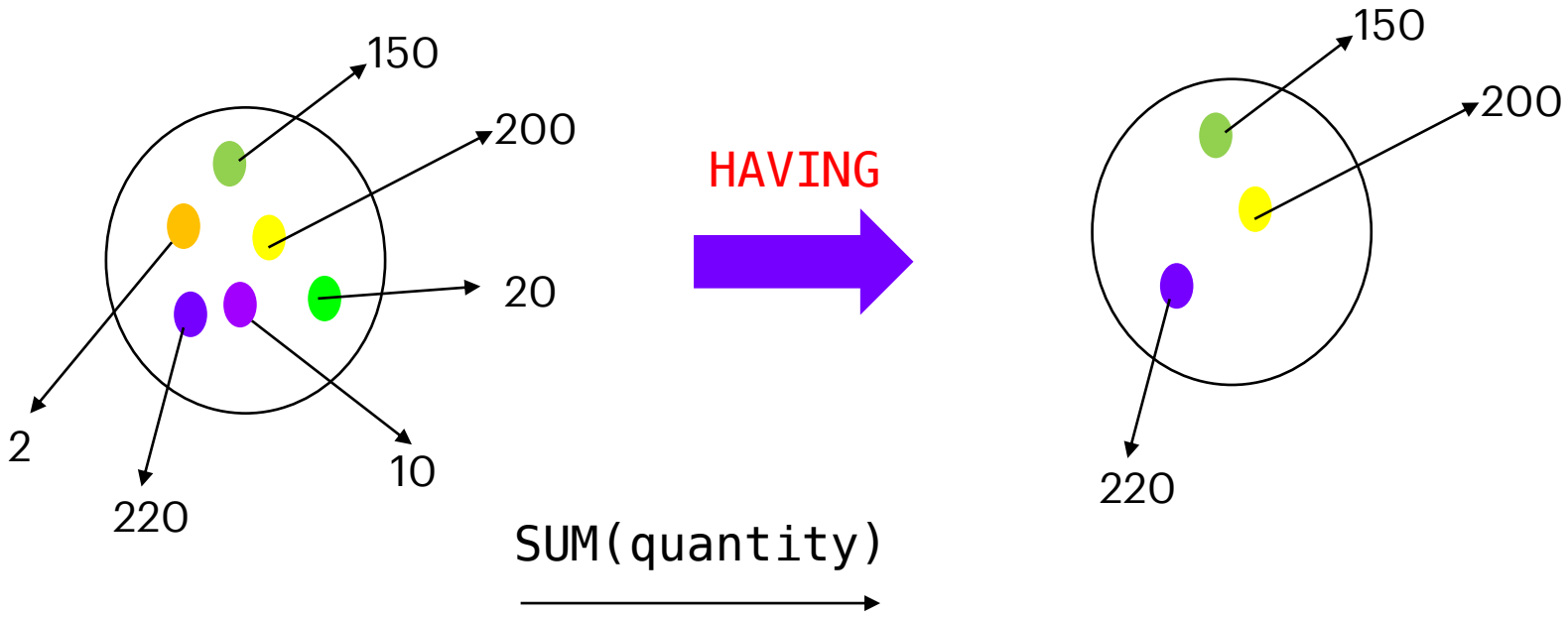
3. Compute the SELECT clause: grouped attributes and aggregates

Product	Total Sales
cakes	?
banana	?

LET US FILTER THE GROUPS WITH "HAVING"

Same query as before, except that we consider only products that have more than 100 buyers.

```
SELECT   product, SUM(price*quantity)
FROM     Purchase
GROUP BY product
HAVING   SUM(quantity) > 100
```



HAVING clauses contains conditions on aggregates

Whereas WHERE clauses condition on individual tuples.

EXAMPLE

Find the glasses that are used at least from 20 alcoholic drinks:

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

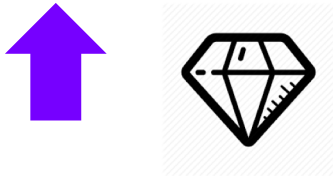
column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE-SOLUTION

Find the glasses that are used at least from 20 alcoholic drinks:

- *select strglass,count(*) as number_cocktails from cocktail where stralcoholic='Alcoholic' group by strglass having(count(*)>20) order by number_cocktails desc;*

strglass character varying (100)	number_cocktails bigint
Cocktail glass	127
Highball glass	72
Collins Glass	52
Old-fashioned glass	51
Collins glass	35
Shot glass	35



Tips

Oder By let us to order the results based on a column name in descending (desc) or ascending (asc) order

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE

Find the names of non alcoholic drinks:

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE-SOLUTION

Find the names of non alcoholic drinks:

- *select distinct stralcoholic from cocktail;*
- *select strdrink from cocktail where stralcoholic like '%Non%' limit 5;*

Be careful '%non% will not work'

Tips limit the result to 10

strdrink character varying (100)
Lassi - Mango
Cranberry Punch
Orange Scented Hot Choco...
Banana Strawberry Shake
Melya

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE

Find how many alcoholic cocktails are using a mug as glass :

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying

EXAMPLE-SOLUTION

Find how many alcoholic cocktails are using a mug as glass :

- *select count(*) from cocktail where strglass like '%mug%' and stralcoholic='Alcoholic'*
- *Answer: 11*

TIPS to check the conditions run first:
*select * from cocktail where strglass like '%mug%' and stralcoholic='Alcoholic'*

cocktail

column_name character varying	data_type character varying
iddrink	character varying
strdrink	character varying
stralcoholic	character varying
strcategory	character varying
strglass	character varying
strinstructions	character varying

ingredients

column_name character varying	data_type character varying
idingredient	character varying
stringridient	character varying

composition

column_name character varying	data_type character varying
id	character varying
iddrink	character varying
idingredient	character varying
strmeasure	character varying