EMORY UNIVERSITY

Managing Big Data

Homework #1 **Due: turned in by Wed 01/22/2020 before class**

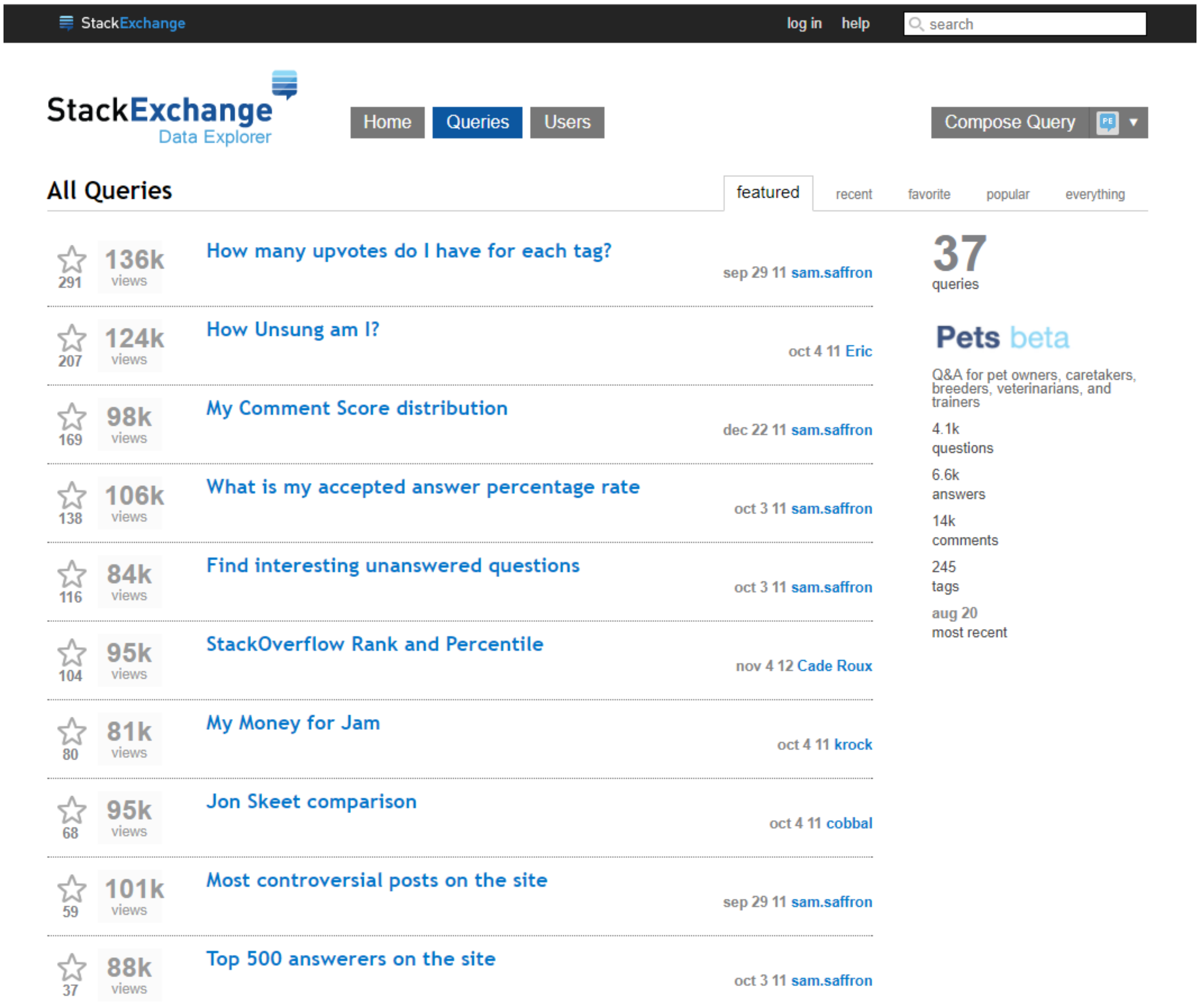
Adil Ashish Kumar

(put your name above)

Total grade: ________ out of ___100___ points

There are 20 numbered questions. Please answer them all and submit your assignment as a single PDF or Word file by uploading it to the HW1 drop-box on the course website. You should provide: SQL statements, results of the SQL statement (typically copy first 10 rows), and answers to questions, if any.

Pets Stackexchange (http://pets.stackexchange.com/) is a Q&A forum for pets. This assignment is based on data of this forum. As other Stackexchange-branded forums, the pet exchange lists questions by the number of votes, answers, and views. Questions can be tagged so that users can easily explore related questions. A screenshot of the forum is below. You can use the StackExchange data explorer (http://data.stackexchange.com/) to explore the data and answer the questions of this assignment. Apart from http://data.stackexchange.com/pets/query/new, you can instead download a data dump and import the database to your local installation of MySQL, if you prefer.



291	136k views	How many upvotes do I have for each tag?	sep 29 11 sam.saffron
207	124k views	How Unsung am I?	oct 4 11 Eric
169	98k views	My Comment Score distribution	dec 22 11 sam.saffron
138	106k views	What is my accepted answer percentage rate	oct 3 11 sam.saffron
116	84k views	Find interesting unanswered questions	oct 3 11 sam.saffron
104	95k views	StackOverflow Rank and Percentile	nov 4 12 Cade Roux
80	81k views	My Money for Jam	oct 4 11 krock
68	95k views	Jon Skeet comparison	oct 4 11 cobbal
59	101k views	Most controversial posts on the site	sep 29 11 sam.saffron
37	88k views	Top 500 answerers on the site	oct 3 11 sam.saffron

37 queries

Pets beta

Q&A for pet owners, caretakers, breeders, veterinarians, and trainers

4.1k questions

6.6k answers

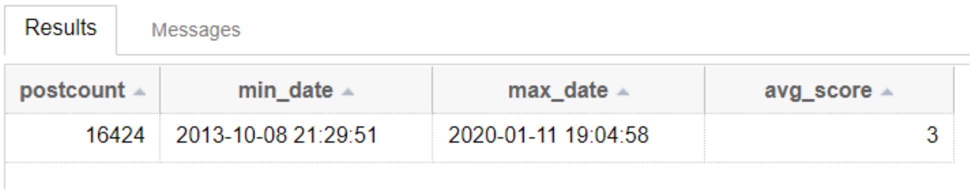
14k comments

245 tags

aug 20 most recent

1. Using the posts table, find out the number of posts, the minimum creation date (as min_date), the maximum creation date (as max_date), and average score (as avg_score).

```
SELECT count(Id) as postcount,min(CreationDate) as min_date,max(CreationDate) as max_date,
avg(Score) as avg_score

FROM Posts;
```

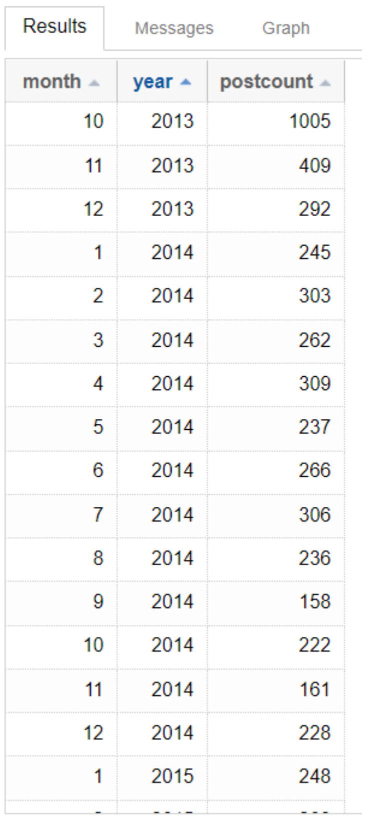


Results | Messages

postcount	min_date	max_date	avg_score
16424	2013-10-08 21:29:51	2020-01-11 19:04:58	3

2. We want to get some ideas of how many posts were written each month. Use SQL to count the number of posts by year-month. Note that by year-month, we mean that May 2013 and May 2014 should be considered as different year-months. The result table should show year-month and count and order results by year-month.

```
SELECT month(CreationDate) as month, year(CreationDate) as year,count(Id) as postcount

FROM Posts

GROUP BY year(CreationDate),month(CreationDate)

ORDER BY Year,month;
```



Results | Messages | Graph

month	year	postcount
10	2013	1005
11	2013	409
12	2013	292
1	2014	245
2	2014	303
3	2014	262
4	2014	309
5	2014	237
6	2014	266
7	2014	306
8	2014	236
9	2014	158
10	2014	222
11	2014	161
12	2014	228
1	2015	248

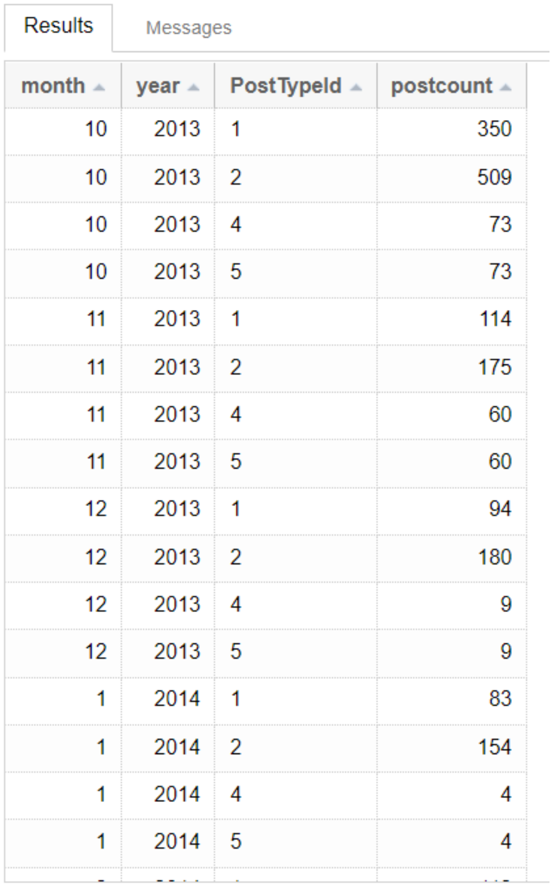
3. We know that there are different types of posts, as reflected by the PostTypeID: the original posts (i.e., 1), follow up posts (i.e., 2), survey questions (i.e., 4), and unknown (i.e., 5). Please use SQL to get the number of posts by year-month and by post type. The results table should show: year, month, posttypeid, and count (label: cnt). Use this result to answer the following question: which year-month has the most original posts?

SELECT month(CreationDate) as month, year(CreationDate) as year,PostTypeId,count(Id) as postcount

FROM Posts

GROUP BY month(CreationDate),year(CreationDate),PostTypeId

ORDER BY year,Month,PostTypeId;



month	year	PostTypeId	postcount
10	2013	1	350
10	2013	2	509
10	2013	4	73
10	2013	5	73
11	2013	1	114
11	2013	2	175
11	2013	4	60
11	2013	5	60
12	2013	1	94
12	2013	2	180
12	2013	4	9
12	2013	5	9
1	2014	1	83
1	2014	2	154
1	2014	4	4
1	2014	5	4

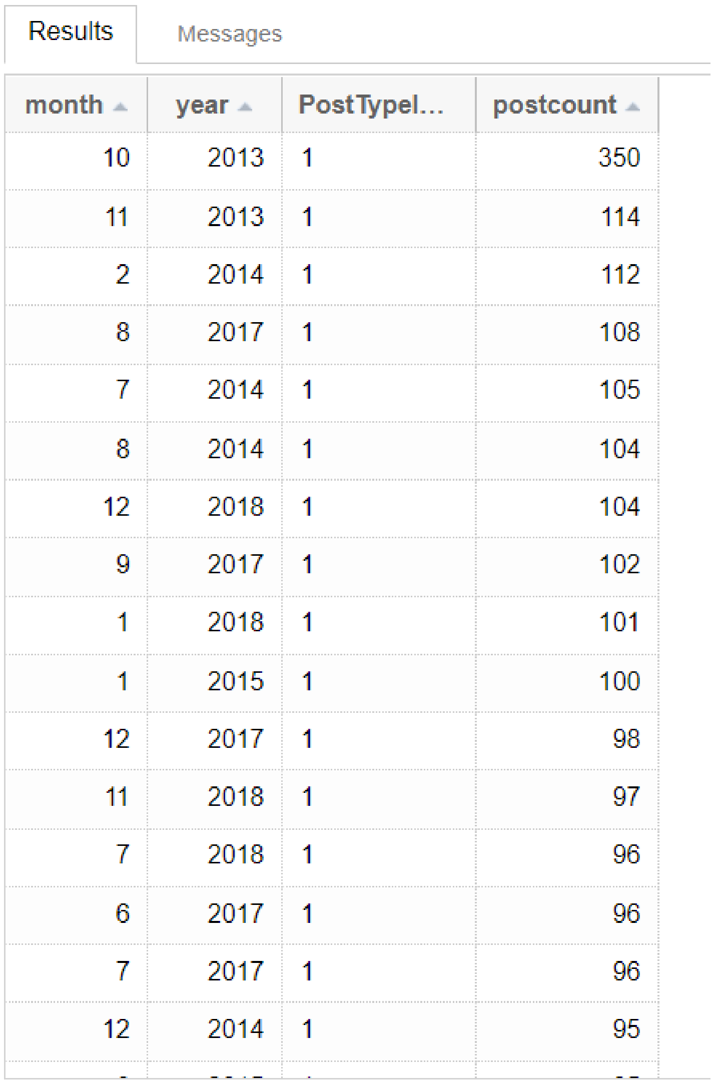
SELECT month(CreationDate) as month, year(CreationDate) as year,PostTypeId,count(Id) as postcount

FROM Posts

where PostTypeId = 1

GROUP BY month(CreationDate),year(CreationDate),PostTypeId

ORDER BY postcount DESC;

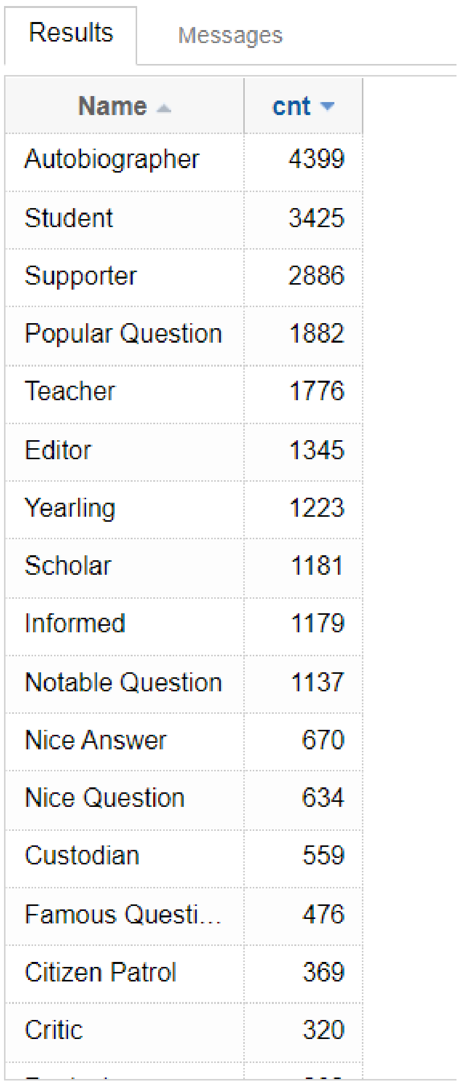

Results Messages

month	year	PostTypeI...	postcount
10	2013	1	350
11	2013	1	114
2	2014	1	112
8	2017	1	108
7	2014	1	105
8	2014	1	104
12	2018	1	104
9	2017	1	102
1	2018	1	101
1	2015	1	100
12	2017	1	98
11	2018	1	97
7	2018	1	96
6	2017	1	96
7	2017	1	96
12	2014	1	95

October 2013 has most original posts

4. Popular badges. Pets Exchange has an elaborate badging system. Use the badges table to find the most common type of badges. Specifically, show badge name (label name) and the number of people who won it (label cnt), limiting results to badges with at least 20 winners and sort the results by cnt in a descending order.

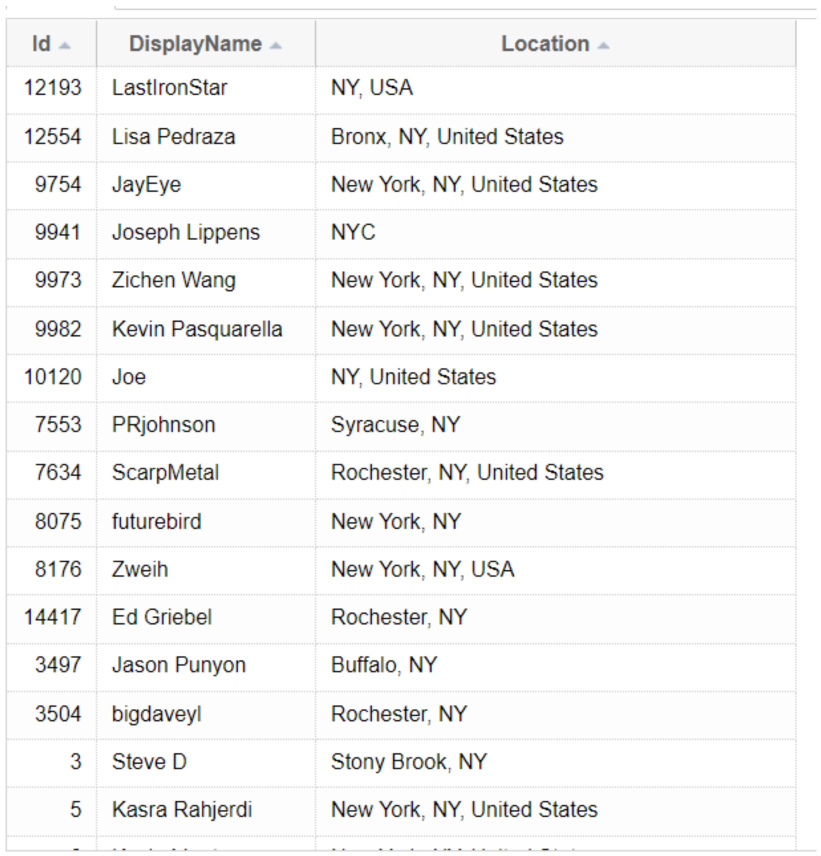
```
SELECT Name, count(Id) as cnt

FROM Badges

GROUP BY Name

HAVING count(Id)>19

ORDER BY cnt DESC;
```


Results Messages

Name	cnt
Autobiographer	4399
Student	3425
Supporter	2886
Popular Question	1882
Teacher	1776
Editor	1345
Yearling	1223
Scholar	1181
Informed	1179
Notable Question	1137
Nice Answer	670
Nice Question	634
Custodian	559
Famous Questi...	476
Citizen Patrol	369
Critic	320

5. (use the users table) Find the number of users who report being located in the New York state. These include people who report themselves in New York city or in the NY state. Be as accurate as possible as the self-reported location may take different forms. Report id, displayname, and location in your result set.

```
SELECT Id,DisplayName,Location

FROM Users

WHERE Location like'%NY%' OR

  Location like '%New york%' ;
```



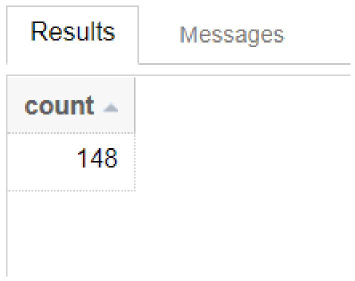
Id	DisplayName	Location
12193	LastIronStar	NY, USA
12554	Lisa Pedraza	Bronx, NY, United States
9754	JayEye	New York, NY, United States
9941	Joseph Lippens	NYC
9973	Zichen Wang	New York, NY, United States
9982	Kevin Pasquarella	New York, NY, United States
10120	Joe	NY, United States
7553	PRjohnson	Syracuse, NY
7634	ScarpMetal	Rochester, NY, United States
8075	futurebird	New York, NY
8176	Zweih	New York, NY, USA
14417	Ed Griebel	Rochester, NY
3497	Jason Punyon	Buffalo, NY
3504	bigdaveyl	Rochester, NY
3	Steve D	Stony Brook, NY
5	Kasra Rahjerdi	New York, NY, United States

SELECT count(distinct Id) as count

FROM Users

WHERE Location like'%NY%' OR

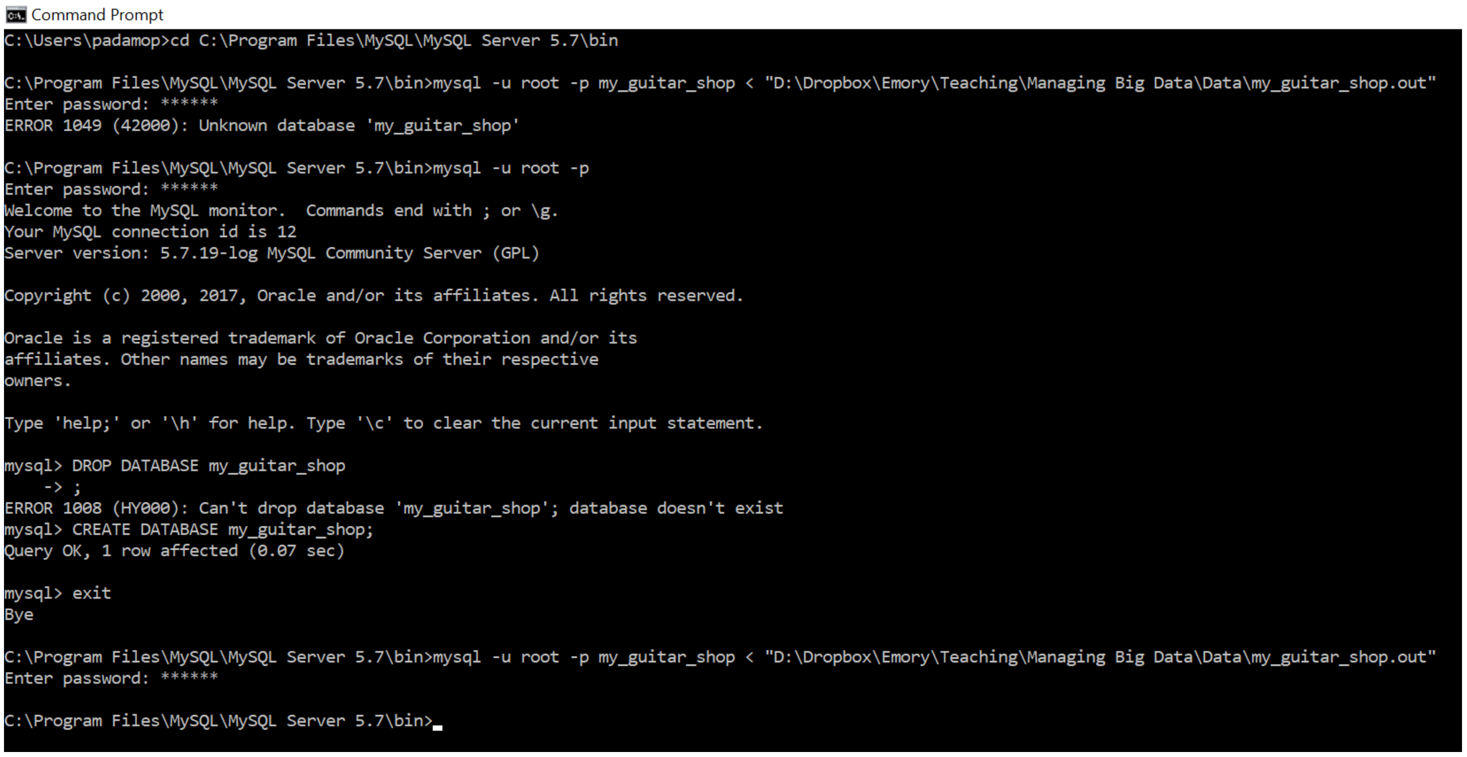
Location like '%New York%' ;



count
148

Please use the *my_guitar_shop* database provided by the instructor to explore the data in your local MySQL instance and answer the next questions.

After installing MySQL, you can import a database following the steps illustrated in the next print screen of the Windows command prompt; the image illustrates both the basic command ‘myqsl –u user –p database_name < database_file’ and how to deal with the common “bug” of ‘unknown database’.



Command Prompt

```
C:\Users\padamop>cd C:\Program Files\MySQL\MySQL Server 5.7\bin

C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p my_guitar_shop < "D:\Dropbox\Emory\Teaching\Managing Big Data\Data\my_guitar_shop.out"
Enter password: ******
ERROR 1049 (42000): Unknown database 'my_guitar_shop'

C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.19-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> DROP DATABASE my_guitar_shop
    -> ;
ERROR 1008 (HY000): Can't drop database 'my_guitar_shop'; database doesn't exist
mysql> CREATE DATABASE my_guitar_shop;
Query OK, 1 row affected (0.07 sec)

mysql> exit
Bye

C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p my_guitar_shop < "D:\Dropbox\Emory\Teaching\Managing Big Data\Data\my_guitar_shop.out"
Enter password: ******

C:\Program Files\MySQL\MySQL Server 5.7\bin>_
```

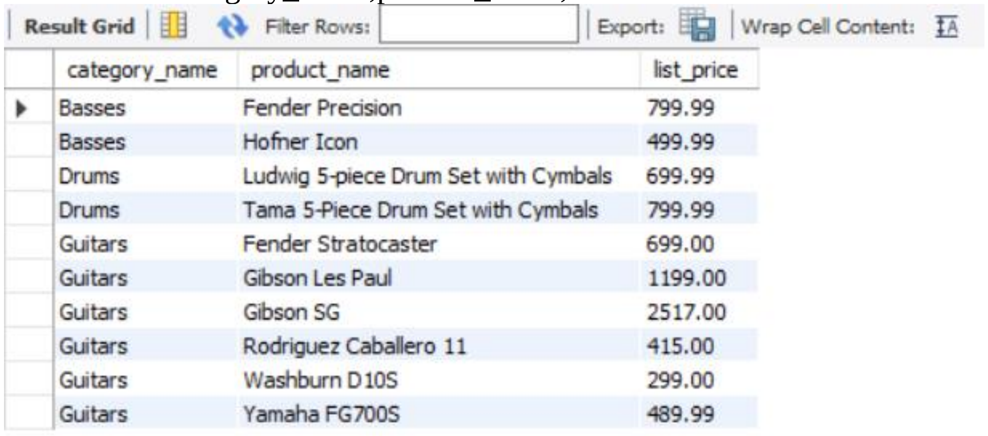
6. Write a SELECT statement that joins the Categories table to the Products table and returns these columns: category_name, product_name, list_price.
Sort the result set by category_name and then by product_name in ascending sequence.

SELECT category_name,product_name,list_price

FROM Products p join categories c

on p.category_id = c.category_id

ORDER BY category_name,product_name;



category_name	product_name	list_price
Basses	Fender Precision	799.99
Basses	Hofner Icon	499.99
Drums	Ludwig 5-piece Drum Set with Cymbals	699.99
Drums	Tama 5-Piece Drum Set with Cymbals	799.99
Guitars	Fender Stratocaster	699.00
Guitars	Gibson Les Paul	1199.00
Guitars	Gibson SG	2517.00
Guitars	Rodriguez Caballero 11	415.00
Guitars	Washburn D10S	299.00
Guitars	Yamaha FG700S	489.99

7. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first_name, last_name, line1, city, state, zip_code.

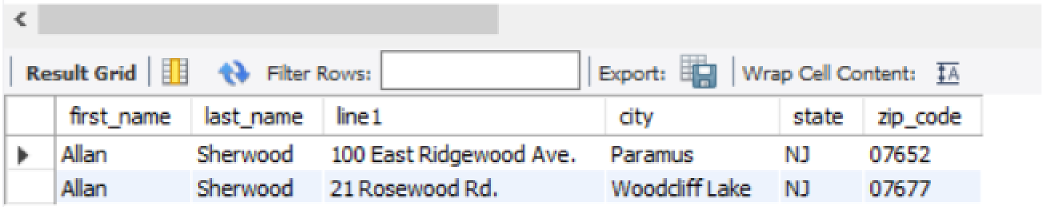
Return one row for each address for the customer with an email address of allan.sherwood@yahoo.com.

SELECT first_name, last_name, line1, city, state, zip_code

FROM addresses a join customers c

ON c.customer_id=a.customer_id

WHERE email_address = 'allan.sherwood@yahoo.com';



first_name	last_name	line1	city	state	zip_code
Allan	Sherwood	100 East Ridgewood Ave.	Paramus	NJ	07652
Allan	Sherwood	21 Rosewood Rd.	Woodcliff Lake	NJ	07677

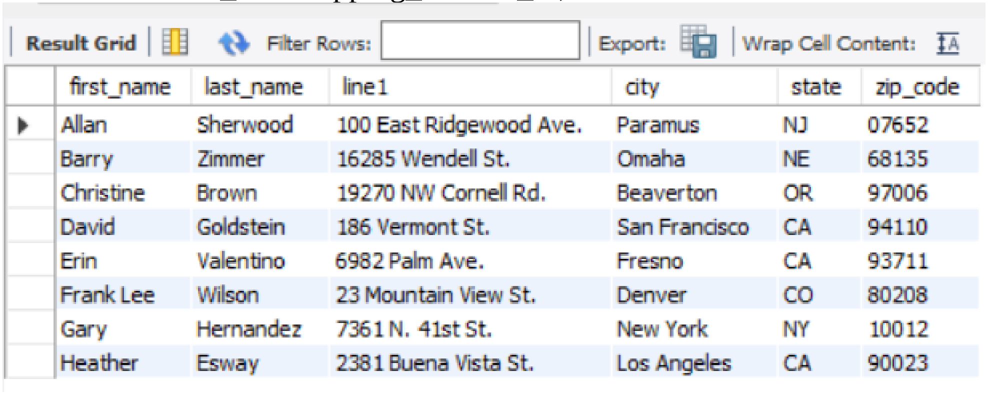
8. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: first_name, last_name, line1, city, state, zip_code.
Return one row for each customer, but only return addresses that are the shipping address for a customer.

SELECT first_name, last_name, line1, city, state, zip_code

FROM addresses a join customers c

ON c.customer_id=a.customer_id

WHERE address_id = shipping_address_id;



first_name	last_name	line1	city	state	zip_code
Allan	Sherwood	100 East Ridgewood Ave.	Paramus	NJ	07652
Barry	Zimmer	16285 Wendell St.	Omaha	NE	68135
Christine	Brown	19270 NW Cornell Rd.	Beaverton	OR	97006
David	Goldstein	186 Vermont St.	San Francisco	CA	94110
Erin	Valentino	6982 Palm Ave.	Fresno	CA	93711
Frank Lee	Wilson	23 Mountain View St.	Denver	CO	80208
Gary	Hernandez	7361 N. 41st St.	New York	NY	10012
Heather	Esway	2381 Buena Vista St.	Los Angeles	CA	90023

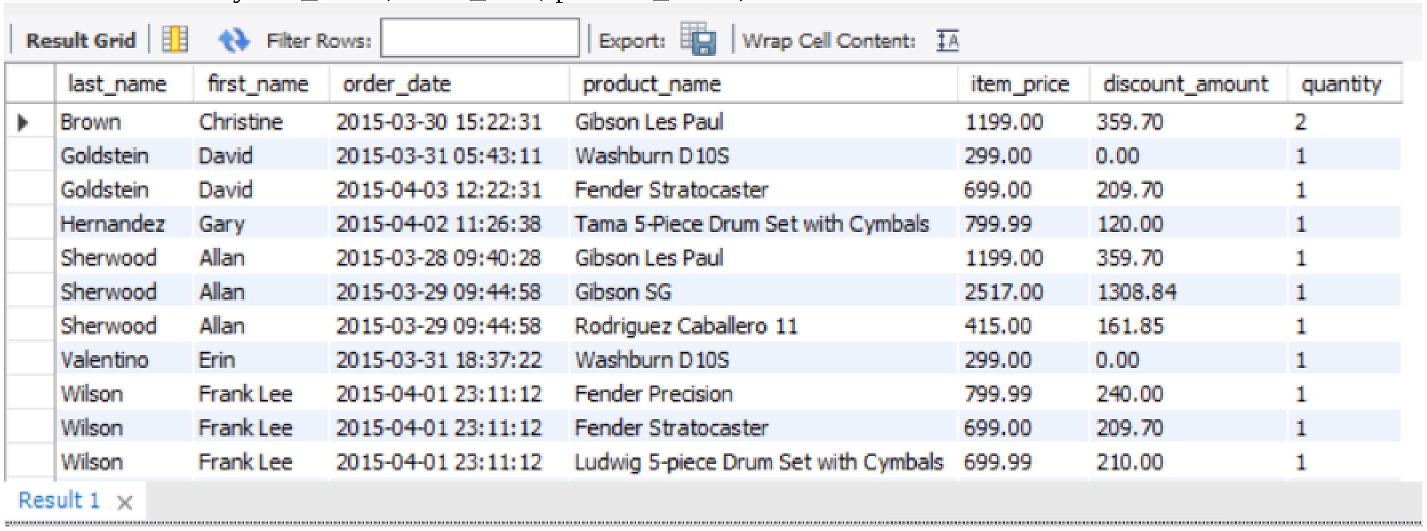
9. Write a SELECT statement that joins the Customers, Orders, Order_Items, and Products tables. This statement should return these columns: last_name, first_name, order_date, product_name, item_price, discount_amount, and quantity.
Use aliases for the tables. Sort the final result set by last_name, order_date, and product_name.

select last_name, first_name, order_date, product_name, item_price, discount_amount,quantity

FROM customers c join orders o

ON c.customer_id = o.customer_id

```
join order_items oi

ON o.order_id=oi.order_id

join products p

ON oi.product_id = p.product_id

ORDER by last_name, order_date, product_name;
```



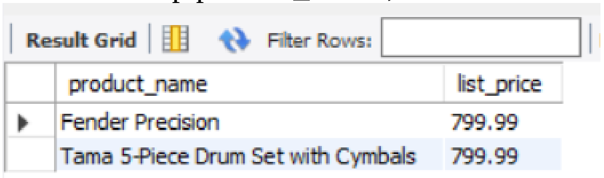
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

last_name	first_name	order_date	product_name	item_price	discount_amount	quantity
Brown	Christine	2015-03-30 15:22:31	Gibson Les Paul	1199.00	359.70	2
Goldstein	David	2015-03-31 05:43:11	Washburn D10S	299.00	0.00	1
Goldstein	David	2015-04-03 12:22:31	Fender Stratocaster	699.00	209.70	1
Hernandez	Gary	2015-04-02 11:26:38	Tama 5-Piece Drum Set with Cymbals	799.99	120.00	1
Sherwood	Allan	2015-03-28 09:40:28	Gibson Les Paul	1199.00	359.70	1
Sherwood	Allan	2015-03-29 09:44:58	Gibson SG	2517.00	1308.84	1
Sherwood	Allan	2015-03-29 09:44:58	Rodriguez Caballero 11	415.00	161.85	1
Valentino	Erin	2015-03-31 18:37:22	Washburn D10S	299.00	0.00	1
Wilson	Frank Lee	2015-04-01 23:11:12	Fender Precision	799.99	240.00	1
Wilson	Frank Lee	2015-04-01 23:11:12	Fender Stratocaster	699.00	209.70	1
Wilson	Frank Lee	2015-04-01 23:11:12	Ludwig 5-piece Drum Set with Cymbals	699.99	210.00	1

Result 1 ×

10. Write a SELECT statement that returns the product_name and list_price columns from the Products table.
Return one row for each product that has the same list price as another product. Sort the result set by product_name.

```
select DISTINCT p.product_name, p.list_price

FROM products p join products p2

ON p.list_price = p2.list_price and p.product_name <>p2.product_name

ORDER BY p.product_name ;
```



Result Grid | Filter Rows:

product_name	list_price
Fender Precision	799.99
Tama 5-Piece Drum Set with Cymbals	799.99

11. Use the UNION operator to generate a result set consisting of three columns from the Orders table:

• **ship_status**	**A calculated column that contains a value of SHIPPED or NOT SHIPPED**
• **order_id**	**The order_id column**
• **order_date**	**The order_date column**

If the order has a value in the ship_date column, the ship_status column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED. Sort the final result set by order_date.

select 'Shipped' as ship_status,order_id, order_date

FROM Orders

where ship_date is not null

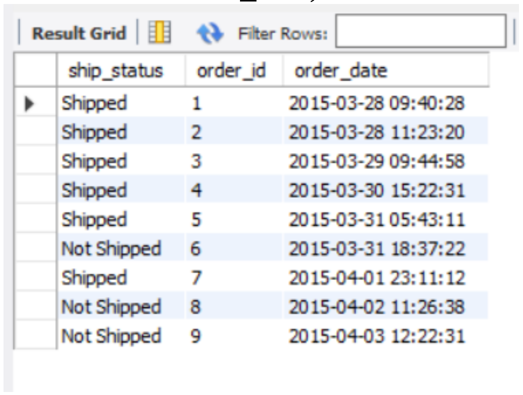
union

select 'Not Shipped' as ship_status,order_id, order_date

FROM Orders

where ship_date is null

ORDER BY order_date;



ship_status	order_id	order_date
Shipped	1	2015-03-28 09:40:28
Shipped	2	2015-03-28 11:23:20
Shipped	3	2015-03-29 09:44:58
Shipped	4	2015-03-30 15:22:31
Shipped	5	2015-03-31 05:43:11
Not Shipped	6	2015-03-31 18:37:22
Shipped	7	2015-04-01 23:11:12
Not Shipped	8	2015-04-02 11:26:38
Not Shipped	9	2015-04-03 12:22:31

12. Write a SELECT statement that returns one row for each category that has products with these columns:

- **The category_name column from the Categories table**
- **The count of the products in the Products table**
- **The list price of the most expensive product in the Products table**

Sort the result set so the category with the most products appears first.

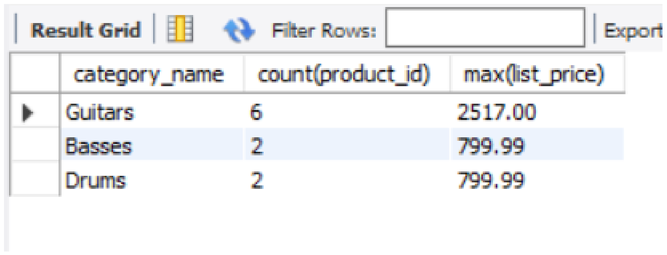
select category_name, count(product_id), max(list_price)

From categories c join products p

on c.category_id = p.category_id

group by category_name

order by count(product_id) DESC;



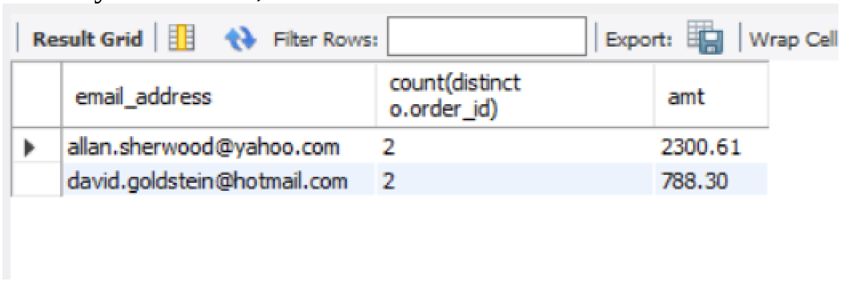
	category_name	count(product_id)	max(list_price)
▶	Guitars	6	2517.00
	Basses	2	799.99
	Drums	2	799.99

13. Write a SELECT statement that returns one row for each customer that has orders with these columns:

- **The email_address from the Customers table**
- **A count of the number of orders**
- **The total amount for each order (Hint: First, subtract the discount amount from the price. Then, multiply by the quantity.)**

Return only those rows where the customer has more than 1 order. Sort the result set in descending sequence by the sum of the line item amounts.

```
select email_address,count(distinct o.order_id), sum(item_price-discount_amount)*quantity as amt

FROM customers c join orders o

on c.customer_id = o.customer_id

join order_items oi

on o.order_id= oi.order_id

group by c.customer_id

having count(distinct o.order_id)>1

order by amt DESC;
```



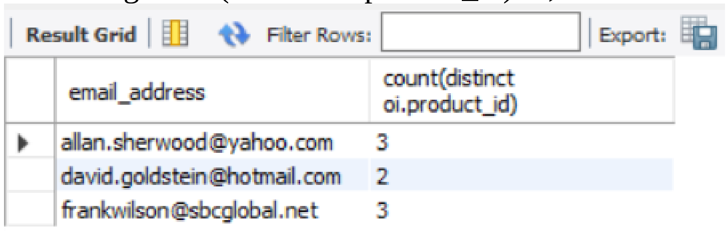
	email_address	count(distinct o.order_id)	amt
▶	allan.sherwood@yahoo.com	2	2300.61
	david.goldstein@hotmail.com	2	788.30

14. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:

- **The email address from the Customers table**
- **The count of distinct products from the customer's orders**

```
select email_address,count(distinct oi.product_id)
FROM customers c join orders o
ON c.customer_id = o.customer_id
join order_items oi
```

```
ON o.order_id=oi.order_id
group by c.customer_id
having count(distinct oi.product_id)>1;
```

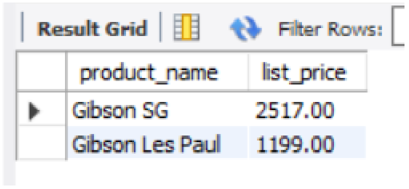


Result Grid | Filter Rows: | Export:

email_address	count(distinct oi.product_id)
allan.sherwood@yahoo.com	3
david.goldstein@hotmail.com	2
frankwilson@sbcglobal.net	3

15. Write a SELECT statement that answers this question: Which products have a list price that's greater than the average list price for all products?
Return the product_name and list_price columns for each product. Sort the results by the list_price column in descending sequence.

```
select product_name,list_price

from products

where list_price> ( select avg(list_price) from products)

Order by List_price DESC;
```

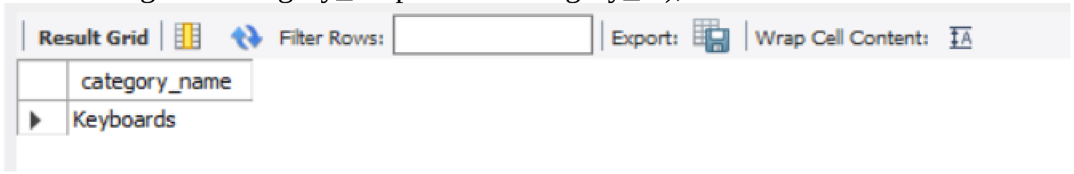


Result Grid | Filter Rows:

product_name	list_price
Gibson SG	2517.00
Gibson Les Paul	1199.00

16. Write a SELECT statement that returns the category_name column from the Categories table. Return one row for each category that has never been assigned to any product in the Products table. To do that, use a subquery introduced with the NOT EXISTS operator.

```
select category_name from categories

where not exists

(select DISTINCT category_id FROM products

where categories.category_id=products.category_id);
```

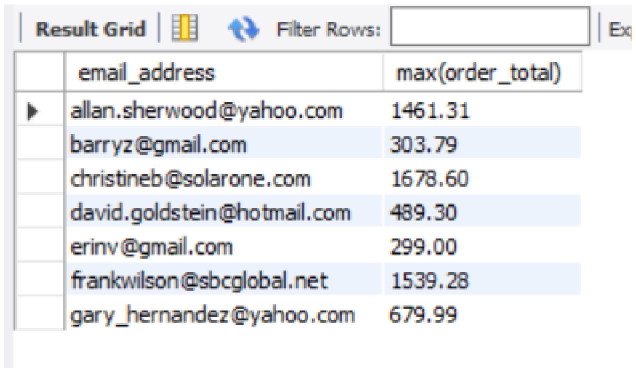


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name
Keyboards

17. Write a SELECT statement that returns three columns: email_address, order_id, and the order total for each customer and order; you must calculate the order total from the columns in the Order_Items table. Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the largest order for that customer.

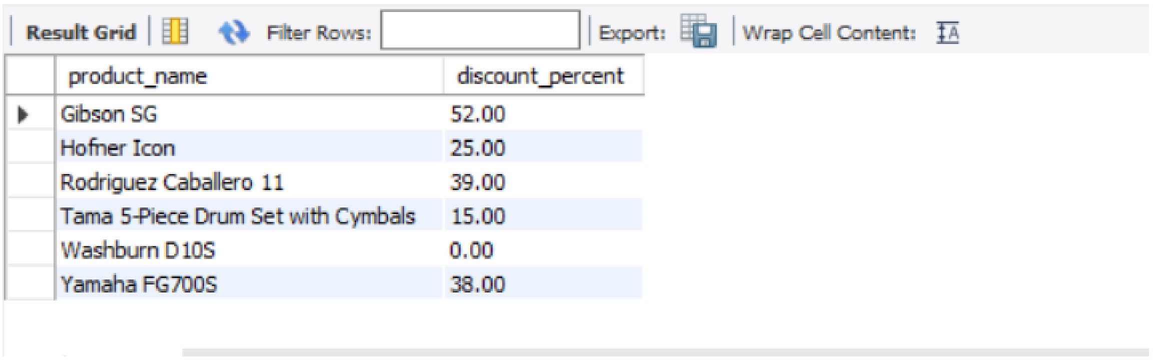
```
select email_address, max(order_total)

from

(select email_address,o.order_id, (item_price-discount_amount)*quantity as order_total

FROM customers c join orders o

on c.customer_id = o.customer_id

join order_items oi

on o.order_id= oi.order_id

group by c.customer_id,o.order_id)t

group by email_address;
```



email_address	max(order_total)
allan.sherwood@yahoo.com	1461.31
barryz@gmail.com	303.79
christineb@solarone.com	1678.60
david.goldstein@hotmail.com	489.30
erinv@gmail.com	299.00
frankwilson@sbcglobal.net	1539.28
gary_hernandez@yahoo.com	679.99

18. Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don't include products that have the same discount percent as another product. Sort the results by the product_name column.

```
select p.product_name, p.discount_percent

FROM products p

where p.discount_percent NOT IN

(select p2.discount_percent

FROM products p2

where p.product_name <> p2.product_name)

ORDER BY product_name;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

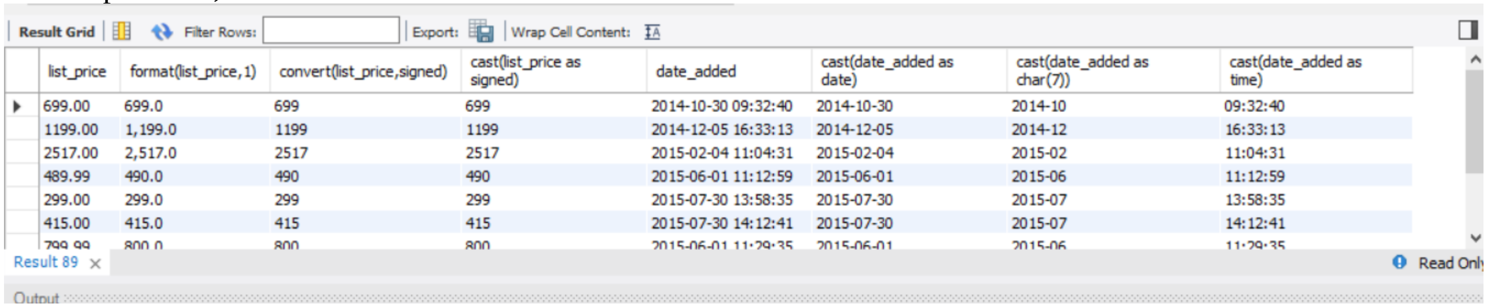
product_name	discount_percent
Gibson SG	52.00
Hofner Icon	25.00
Rodriguez Caballero 11	39.00
Tama 5-Piece Drum Set with Cymbals	15.00
Washburn D10S	0.00
Yamaha FG700S	38.00

19. Write a SELECT statement that returns these columns from the Products table:

- **The list_price column**
- **A column that uses the FORMAT function to return the list_price column with 1 digit to the right of the decimal point**
- **A column that uses the CONVERT function to return the list_price column as an integer**
- **A column that uses the CAST function to return the list_price column as an integer**
- **The date_added column**
- **A column that uses the CAST function to return the date_added column with its date only (year, month, and day)**
- **A column that uses the CAST function to return the date_added column with just the year and the month**
- **A column that uses the CAST function to return the date_added column with its full time only (hour, minutes, and seconds)**

select list_price, format(list_price,1), convert(list_price,signed), cast(list_price as signed),date_added, cast(date_added as date), cast(date_added as char(7)), cast(date_added as time)

FROM products;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

list_price	format(list_price,1)	convert(list_price,signed)	cast(list_price as signed)	date_added	cast(date_added as date)	cast(date_added as char(7))	cast(date_added as time)
699.00	699.0	699	699	2014-10-30 09:32:40	2014-10-30	2014-10	09:32:40
1199.00	1,199.0	1199	1199	2014-12-05 16:33:13	2014-12-05	2014-12	16:33:13
2517.00	2,517.0	2517	2517	2015-02-04 11:04:31	2015-02-04	2015-02	11:04:31
489.99	490.0	490	490	2015-06-01 11:12:59	2015-06-01	2015-06	11:12:59
299.00	299.0	299	299	2015-07-30 13:58:35	2015-07-30	2015-07	13:58:35
415.00	415.0	415	415	2015-07-30 14:12:41	2015-07-30	2015-07	14:12:41
799.99	800.0	800	800	2015-06-01 11:29:35	2015-06-01	2015-06	11:29:35

Result 89 × Read Only

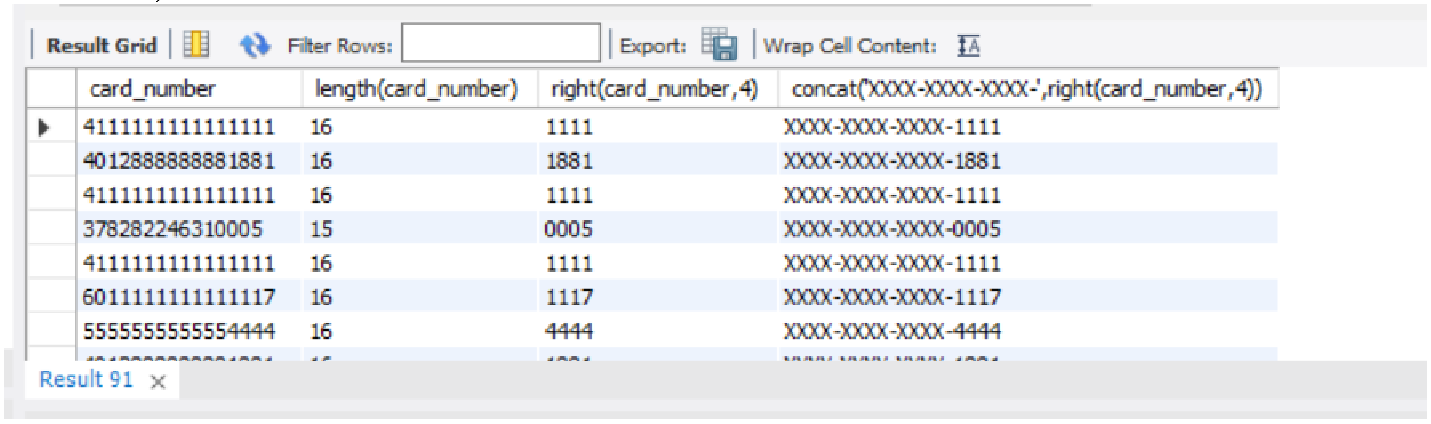
Output

20. Write a SELECT statement that returns these columns from the Orders table:

- **The card_number column**
- **The length of the card_number column**
- **The last four digits of the card_number column**
- **A column that displays the last four digits of the card_number column in this format: XXXX-XXXX-XXXX-1234. In other words, use Xs for the first 12 digits of the card number and actual numbers for the last four digits of the number.**

  select card_number , length(card_number), right(card_number,4), concat('XXXX-XXXX-XXXX-',right(card_number,4))

```
from orders;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

card_number	length(card_number)	right(card_number,4)	concat('XXXX-XXXX-XXXX-',right(card_number,4))
4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
378282246310005	15	0005	XXXX-XXXX-XXXX-0005
4111111111111111	16	1111	XXXX-XXXX-XXXX-1111
6011111111111117	16	1117	XXXX-XXXX-XXXX-1117
5555555555554444	16	4444	XXXX-XXXX-XXXX-4444

Result 91