SQL answers

**Solutions: GROUP BY Part II**

1. For each account, determine the average amount of each type of paper they purchased across their orders. Your result should have four columns - one for the account **name** and one for the average spent on each of the paper types. **SELECT** a.**name**, **AVG**(o.standard\_qty) avg\_stand, **AVG**(o.gloss\_qty) avg\_gloss, **AVG**(o.poster\_qty) avg\_post
2. **FROM** accounts a
3. **JOIN** orders o
4. **ON** a.**id** = o.account\_id
5. **GROUP** **BY** a.**name**;
6. For each account, determine the average amount spent per order on each paper type. Your result should have four columns - one for the account **name** and one for the average amount spent on each paper type.**SELECT** a.**name**, **AVG**(o.standard\_amt\_usd) avg\_stand, **AVG**(o.gloss\_amt\_usd) avg\_gloss, **AVG**(o.poster\_amt\_usd) avg\_post
7. **FROM** accounts a
8. **JOIN** orders o
9. **ON** a.**id** = o.account\_id
10. **GROUP** **BY** a.**name**;
11. Determine the number of times a particular **channel** was used in the **web\_events** table for each **sales rep**. Your final table should have three columns - the **name of the sales rep**, the **channel**, and the number of occurrences. Order your table with the highest number of occurrences first.**SELECT** s.**name**, w.channel, **COUNT**(\*) num\_events
12. **FROM** accounts a
13. **JOIN** web\_events w
14. **ON** a.**id** = w.account\_id
15. **JOIN** sales\_reps s
16. **ON** s.**id** = a.sales\_rep\_id
17. **GROUP** **BY** s.**name**, w.channel
18. **ORDER** **BY** num\_events **DESC**;
19. Determine the number of times a particular **channel** was used in the **web\_events** table for each **region**. Your final table should have three columns - the **region name**, the **channel**, and the number of occurrences. Order your table with the highest number of occurrences first.**SELECT** r.**name**, w.channel, **COUNT**(\*) num\_events
20. **FROM** accounts a
21. **JOIN** web\_events w
22. **ON** a.**id** = w.account\_id
23. **JOIN** sales\_reps s
24. **ON** s.**id** = a.sales\_rep\_id
25. **JOIN** region r
26. **ON** r.**id** = s.region\_id
27. **GROUP** **BY** r.**name**, w.channel

**ORDER** **BY** num\_events **DESC**;

**Questions: HAVING**

Use the **SQL** environment below to assist with answering the following questions. Whether you get stuck or you just want to double check your solutions, my answers can be found at the top of the next concept.

1. How many of the **sales reps** have more than 5 accounts that they manage?
2. How many **accounts** have more than 20 orders?
3. Which account has the most orders?
4. How many accounts spent more than 30,000 usd total across all orders?
5. How many accounts spent less than 1,000 usd total across all orders?
6. Which account has spent the most with us?
7. Which account has spent the least with us?
8. Which accounts used facebook as a **channel** to contact customers more than 6 times?
9. Which account used facebook most as a **channel**?

Which channel was most frequently used by most accounts?

**Solutions: HAVING**

1. How many of the **sales reps** have more than 5 accounts that they manage? **SELECT** s.**id**, s.**name**, **COUNT**(\*) num\_accounts
2. **FROM** accounts a
3. **JOIN** sales\_reps s
4. **ON** s.**id** = a.sales\_rep\_id
5. **GROUP** **BY** s.**id**, s.**name**
6. **HAVING** **COUNT**(\*) > 5
7. **ORDER** **BY** num\_accounts;
8. and technically, we can get this using a **SUBQUERY** as shown below. This same logic can be used for the other queries, but this will not be shown. **SELECT** **COUNT**(\*) num\_reps\_above5
9. **FROM**(**SELECT** s.**id**, s.**name**, **COUNT**(\*) num\_accounts
10. **FROM** accounts a
11. **JOIN** sales\_reps s
12. **ON** s.**id** = a.sales\_rep\_id
13. **GROUP** **BY** s.**id**, s.**name**
14. **HAVING** **COUNT**(\*) > 5
15. **ORDER** **BY** num\_accounts) **AS** Table1;
16. How many **accounts** have more than 20 orders?   **SELECT** a.**id**, a.**name**, **COUNT**(\*) num\_orders
17. **FROM** accounts a
18. **JOIN** orders o
19. **ON** a.**id** = o.account\_id
20. **GROUP** **BY** a.**id**, a.**name**
21. **HAVING** **COUNT**(\*) > 20
22. **ORDER** **BY** num\_orders;
23. Which account has the most orders? **SELECT** a.**id**, a.**name**, **COUNT**(\*) num\_orders
24. **FROM** accounts a
25. **JOIN** orders o
26. **ON** a.**id** = o.account\_id
27. **GROUP** **BY** a.**id**, a.**name**
28. **ORDER** **BY** num\_orders **DESC**
29. **LIMIT** 1;
30. How many accounts spent more than 30,000 usd total across all orders? **SELECT** a.**id**, a.**name**, **SUM**(o.total\_amt\_usd) total\_spent
31. **FROM** accounts a
32. **JOIN** orders o
33. **ON** a.**id** = o.account\_id
34. **GROUP** **BY** a.**id**, a.**name**
35. **HAVING** **SUM**(o.total\_amt\_usd) > 30000
36. **ORDER** **BY** total\_spent;
37. How many accounts spent less than 1,000 usd total across all orders? **SELECT** a.**id**, a.**name**, **SUM**(o.total\_amt\_usd) total\_spent
38. **FROM** accounts a
39. **JOIN** orders o
40. **ON** a.**id** = o.account\_id
41. **GROUP** **BY** a.**id**, a.**name**
42. **HAVING** **SUM**(o.total\_amt\_usd) < 1000
43. **ORDER** **BY** total\_spent;
44. Which account has spent the most with us? **SELECT** a.**id**, a.**name**, **SUM**(o.total\_amt\_usd) total\_spent
45. **FROM** accounts a
46. **JOIN** orders o
47. **ON** a.**id** = o.account\_id
48. **GROUP** **BY** a.**id**, a.**name**
49. **ORDER** **BY** total\_spent **DESC**
50. **LIMIT** 1;
51. Which account has spent the least with us? **SELECT** a.**id**, a.**name**, **SUM**(o.total\_amt\_usd) total\_spent
52. **FROM** accounts a
53. **JOIN** orders o
54. **ON** a.**id** = o.account\_id
55. **GROUP** **BY** a.**id**, a.**name**
56. **ORDER** **BY** total\_spent
57. **LIMIT** 1;
58. Which accounts used facebook as a **channel** to contact customers more than 6 times? **SELECT** a.**id**, a.**name**, w.channel, **COUNT**(\*) use\_of\_channel
59. **FROM** accounts a
60. **JOIN** web\_events w
61. **ON** a.**id** = w.account\_id
62. **GROUP** **BY** a.**id**, a.**name**, w.channel
63. **HAVING** **COUNT**(\*) > 6 **AND** w.channel = 'facebook'
64. **ORDER** **BY** use\_of\_channel;
65. Which account used facebook most as a **channel**?  **SELECT** a.**id**, a.**name**, w.channel, **COUNT**(\*) use\_of\_channel
66. **FROM** accounts a
67. **JOIN** web\_events w
68. **ON** a.**id** = w.account\_id
69. **WHERE** w.channel = 'facebook'
70. **GROUP** **BY** a.**id**, a.**name**, w.channel
71. **ORDER** **BY** use\_of\_channel **DESC**
72. **LIMIT** 1;
73. Which channel was most frequently used by most accounts?**SELECT** a.**id**, a.**name**, w.channel, **COUNT**(\*) use\_of\_channel
74. **FROM** accounts a
75. **JOIN** web\_events w
76. **ON** a.**id** = w.account\_id
77. **GROUP** **BY** a.**id**, a.**name**, w.channel
78. **ORDER** **BY** use\_of\_channel **DESC**
79. **LIMIT** 10;

 All of the top 10 are direct.

**Questions: Working With DATEs**

Use the **SQL** environment below to assist with answering the following questions. Whether you get stuck or you just want to double check your solutions, my answers can be found at the top of the next concept.

1. Find the sales in terms of total dollars for all orders in each year, ordered from greatest to least. Do you notice any trends in the yearly sales totals?
2. Which **month** did Parch & Posey have the greatest sales in terms of total dollars? Are all months evenly represented by the dataset?
3. Which **year** did Parch & Posey have the greatest sales in terms of total number of orders? Are all years evenly represented by the dataset?
4. Which **month** did Parch & Posey have the greatest sales in terms of total number of orders? Are all months evenly represented by the dataset?

In which **month** of which **year** did Walmart spend the most on gloss paper in terms of dollars?

**Solutions: Working With DATEs**

1. Find the sales in terms of total dollars for all orders in each year, ordered from greatest to least. Do you notice any trends in the yearly sales totals?

**SELECT** DATE\_PART('year', occurred\_at) ord\_year, **SUM**(total\_amt\_usd) total\_spent

**FROM** orders

**GROUP** **BY** 1

**ORDER** **BY** 2 **DESC**;

When we look at the yearly totals, you might notice that 2013 and 2017 have much smaller totals than all other years. If we look further at the monthly data, we see that for 2013 and 2017 there is only one month of sales for each of these years (12 for 2013 and 1 for 2017). Therefore, neither of these are evenly represented. Sales have been increasing year over year, with 2016 being the largest sales to date. At this rate, we might expect 2017 to have the largest sales.

1. Which **month** did Parch & Posey have the greatest sales in terms of total dollars? Are all months evenly represented by the dataset?  In order for this to be 'fair', we should remove the sales from 2013 and 2017. For the same reasons as discussed above. **SELECT** DATE\_PART('month', occurred\_at) ord\_month, **SUM**(total\_amt\_usd) total\_spent
2. **FROM** orders
3. **WHERE** occurred\_at **BETWEEN** '2014-01-01' **AND** '2017-01-01'
4. **GROUP** **BY** 1
5. **ORDER** **BY** 2 **DESC**;
6. The greatest sales amounts occur in December (12).
7. Which **year** did Parch & Posey have the greatest sales in terms of total number of orders? Are all years evenly represented by the dataset? **SELECT** DATE\_PART('year', occurred\_at) ord\_year, **COUNT**(\*) total\_sales
8. **FROM** orders
9. **GROUP** **BY** 1
10. **ORDER** **BY** 2 **DESC**;
11. Again, 2016 by far has the most amount of orders, but again 2013 and 2017 are not evenly represented to the other years in the dataset.
12. Which **month** did Parch & Posey have the greatest sales in terms of total number of orders? Are all months evenly represented by the dataset?**SELECT** DATE\_PART('month', occurred\_at) ord\_month, **COUNT**(\*) total\_sales
13. **FROM** orders
14. **WHERE** occurred\_at **BETWEEN** '2014-01-01' **AND** '2017-01-01'
15. **GROUP** **BY** 1
16. **ORDER** **BY** 2 **DESC**;
17. December still has the most sales, but interestingly, November has the second most sales (but not the most dollar sales. To make a fair comparison from one month to another 2017 and 2013 data were removed.
18. In which **month** of which **year** did Walmart spend the most on gloss paper in terms of dollars?**SELECT** DATE\_TRUNC('month', o.occurred\_at) ord\_date, **SUM**(o.gloss\_amt\_usd) tot\_spent
19. **FROM** orders o
20. **JOIN** accounts a
21. **ON** a.**id** = o.account\_id
22. **WHERE** a.**name** = 'Walmart'
23. **GROUP** **BY** 1
24. **ORDER** **BY** 2 **DESC**
25. **LIMIT** 1;

 May 2016 was when Walmart spent the most on gloss paper

**Example**

In a quiz question in the previous Basic SQL lesson, you saw this question:

1. Create a column that divides the standard\_amt\_usd by the standard\_qty to find the unit price for standard paper for each order. Limit the results to the first 10 orders, and include the id and account\_id fields. **NOTE - you will be thrown an error with the correct solution to this question. This is for a division by zero. You will learn how to get a solution without an error to this query when you learn about CASE statements in a later section.**

Let's see how we can use the **CASE** statement to get around this error.

**SELECT** **id**, account\_id, standard\_amt\_usd/standard\_qty **AS** unit\_price

**FROM** orders

**LIMIT** 10;

Now, let's use a **CASE** statement. This way any time the **standard\_qty** is zero, we will return 0, and otherwise we will return the **unit\_price**.

**SELECT** account\_id, **CASE** **WHEN** standard\_qty = 0 **OR** standard\_qty **IS** NULL **THEN** 0

**ELSE** standard\_amt\_usd/standard\_qty **END** **AS** unit\_price

**FROM** orders

**LIMIT** 10;

Now the first part of the statement will catch any of those division by zero values that were causing the error, and the other components will compute the division as necessary. You will notice, we essentially charge all of our accounts 4.99 for standard paper. It makes sense this doesn't fluctuate, and it is more accurate than adding 1 in the denominator like our quick fix might have been in the earlier lesson.

You can try it yourself using the environment below

This one is pretty tricky. Try running the query yourself to make sure you understand what is happening. The next concept will give you some practice writing **CASE** statements on your own. In this video, we showed that getting the same information using a **WHERE** clause means only being able to get one set of data from the **CASE** at a time.

There are some advantages to separating data into separate columns like this depending on what you want to do, but often this level of separation might be easier to do in another programming language - rather than with SQL.

**Questions: CASE**

Use the **SQL** environment below to assist with answering the following questions. Whether you get stuck or you just want to double check your solutions, my answers can be found at the top of the next concept.

1. We would like to understand 3 different levels of customers based on the amount associated with their purchases. The top branch includes anyone with a Lifetime Value (total sales of all orders) greater than 200,000 usd. The second branch is between 200,000 and 100,000 usd. The lowest branch is anyone under 100,000 usd. Provide a table that includes the **level** associated with each **account**. You should provide the **account name**, the **total sales of all orders** for the customer, and the **level**. Order with the top spending customers listed first.
2. We would now like to perform a similar calculation to the first, but we want to obtain the total amount spent by customers only in 2016 and 2017. Keep the same **level**s as in the previous question. Order with the top spending customers listed first.
3. We would like to identify top performing **sales reps**, which are sales reps associated with more than 200 orders. Create a table with the **sales rep name**, the total number of orders, and a column with top or not depending on if they have more than 200 orders. Place the top sales people first in your final table.

The previous didn't account for the middle, nor the dollar amount associated with the sales. Management decides they want to see these characteristics represented as well. We would like to identify top performing **sales reps**, which are sales reps associated with more than 200 orders or more than 750000 in total sales. The middle group has any **rep** with more than 150 orders or 500000 in sales. Create a table with the **sales rep name**, the total number of orders, total sales across all orders, and a column with top, middle, or low depending on this criteria. Place the top sales people based on dollar amount of sales first in your final table. You might see a few upset sales people by this criteria!

**Solutions: CASE**

1. We would like to understand 3 different branches of customers based on the amount associated with their purchases. The top branch includes anyone with a Lifetime Value (total sales of all orders) greater than 200,000 usd. The second branch is between 200,000 and 100,000 usd. The lowest branch is anyone under 100,000 usd. Provide a table that includes the **level** associated with each **account**. You should provide the **account name**, the **total sales of all orders** for the customer, and the **level**. Order with the top spending customers listed first. **SELECT** a.**name**, **SUM**(total\_amt\_usd) total\_spent,
2. **CASE** **WHEN** **SUM**(total\_amt\_usd) > 200000 **THEN** 'top'
3. **WHEN** **SUM**(total\_amt\_usd) > 100000 **THEN** 'middle'
4. **ELSE** 'low' **END** **AS** customer\_level
5. **FROM** orders o
6. **JOIN** accounts a
7. **ON** o.account\_id = a.**id**
8. **GROUP** **BY** a.**name**
9. **ORDER** **BY** 2 **DESC**;
10. We would now like to perform a similar calculation to the first, but we want to obtain the total amount spent by customers only in 2016 and 2017. Keep the same **level**s as in the previous question. Order with the top spending customers listed first. **SELECT** a.**name**, **SUM**(total\_amt\_usd) total\_spent,
11. **CASE** **WHEN** **SUM**(total\_amt\_usd) > 200000 **THEN** 'top'
12. **WHEN** **SUM**(total\_amt\_usd) > 100000 **THEN** 'middle'
13. **ELSE** 'low' **END** **AS** customer\_level
14. **FROM** orders o
15. **JOIN** accounts a
16. **ON** o.account\_id = a.**id**
17. **WHERE** occurred\_at > '2015-12-31'
18. **GROUP** **BY** 1
19. **ORDER** **BY** 2 **DESC**;
20. We would like to identify top performing **sales reps**, which are sales reps associated with more than 200 orders. Create a table with the **sales rep name**, the total number of orders, and a column with top or not depending on if they have more than 200 orders. Place the top sales people first in your final table.**SELECT** s.**name**, **COUNT**(\*) num\_ords,
21. **CASE** **WHEN** **COUNT**(\*) > 200 **THEN** 'top'
22. **ELSE** 'not' **END** **AS** sales\_rep\_level
23. **FROM** orders o
24. **JOIN** accounts a
25. **ON** o.account\_id = a.**id**
26. **JOIN** sales\_reps s
27. **ON** s.**id** = a.sales\_rep\_id
28. **GROUP** **BY** s.**name**
29. **ORDER** **BY** 2 **DESC**;
30. It is worth mentioning that this assumes each name is unique - which has been done a few times. We otherwise would want to break by the name and the id of the table.
31. The previous didn't account for the middle, nor the dollar amount associated with the sales. Management decides they want to see these characteristics represented as well. We would like to identify top performing **sales reps**, which are sales reps associated with more than 200 orders or more than 750000 in total sales. The middle group has any **rep** with more than 150 orders or 500000 in sales. Create a table with the **sales rep name**, the total number of orders, total sales across all orders, and a column with top, middle, or low depending on this criteria. Place the top sales people based on dollar amount of sales first in your final table.**SELECT** s.**name**, **COUNT**(\*), **SUM**(o.total\_amt\_usd) total\_spent,
32. **CASE** **WHEN** **COUNT**(\*) > 200 **OR** **SUM**(o.total\_amt\_usd) > 750000 **THEN** 'top'
33. **WHEN** **COUNT**(\*) > 150 **OR** **SUM**(o.total\_amt\_usd) > 500000 **THEN** 'middle'
34. **ELSE** 'low' **END** **AS** sales\_rep\_level
35. **FROM** orders o
36. **JOIN** accounts a
37. **ON** o.account\_id = a.**id**
38. **JOIN** sales\_reps s
39. **ON** s.**id** = a.sales\_rep\_id
40. **GROUP** **BY** s.**name**
41. **ORDER** **BY** 3 **DESC**;

 You might see a few upset sales people by this criteria!

**RECAP**

Each of the sections has been labeled to assist if you need to revisit a particular topic. Intentionally, the solutions for a particular section are actually not in the labeled section, because my hope is this will force you to practice if you have a question about a particular topic we covered.

You have now gained a ton of useful skills associated with **SQL**. The combination of **JOINs** and **Aggregations** are one of the reasons **SQL** is such a powerful tool.

If there was a particular topic you struggled with, I suggest coming back and revisiting the questions with a fresh mind. The more you practice the better, but you also don't want to get stuck on the same problem for an extended period of time!