1) Try using LIMIT yourself below by writing a query that displays all the data in the occurred_at, account_id, and channel columns of the web_events table, and limits the output to only the first 15 rows.

```
SELECT occurred_at, account_id, channel
FROM web_events
LIMIT 15;
```

2) Write a query to return the 10 earliest orders in the **orders** table. Include the id, occurred_at, and total_amt_usd.

```
SELECT id, occurred_at, total_amt_usd
FROM orders
ORDER BY occurred_at
LIMIT 10;
```

3) Write a query to return the top 5 **orders** in terms of largest total_amt_usd. Include the id, account_id, and total_amt_usd.

```
SELECT id, account_id, total_amt_usd
FROM orders
ORDER BY total_amt_usd DESC
LIMIT 5;
```

4) Write a query that displays the order ID, account ID, and total dollar amount for all the orders, sorted first by the account ID (in ascending order), and then by the total dollar amount (in descending order).

```
SELECT id, account_id, total_amt_usd
FROM orders
ORDER BY account_id, total_amt_usd DESC;
```

5) Now write a query that again displays order ID, account ID, and total dollar amount for each order, but this time sorted first by total dollar amount (in descending order), and then by account ID (in ascending order).

```
SELECT id, account_id, total_amt_usd
FROM orders
ORDER BY total_amt_usd DESC, account_id;
```

6) Compare the results of these two queries above. How are the results different when you switch the column you sort on first?

From question 4

id	account_id	total_amt_usd
4308	1001	9426.71
4309	1001	9230.67
4316	1001	9134.31
4317	1001	8963.91
4314	1001	8863.24
4307	1001	8757.18

From question 5

id	account_id	total_amt_usd
4016	4251	232207.07
3892	4161	112875.18
3963	4211	107533.55
5791	2861	95005.82
3778	4101	93547.84
6590	4111	93505.69

7) Pulls the first 5 rows and all columns from the **orders** table that have a dollar amount of gloss_amt_usd greater than or equal to 1000.

```
SELECT *
FROM orders
WHERE gloss_amt_usd >= 1000
LIMIT 5;
```

8) Filter the accounts table to include the company name, website, and the primary point of contact (primary_poc) just for the Exxon Mobil company in the **accounts** table.

```
SELECT name, website, primary_poc
FROM accounts
WHERE name = 'Exxon Mobil';
```

9) 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.

```
SELECT id, account_id, standard_amt_usd/standard_qty AS unit_price
FROM orders
LIMIT 10;
```

10) Write a query that finds the percentage of revenue that comes from poster paper for each order. You will need to use only the columns that end with _usd. (Try to do this without using the total column.) Display the id and account_id fields also.

```
SELECT id, account_id,
   poster_amt_usd/(standard_amt_usd + gloss_amt_usd + poster_amt_usd) A
S post_per
FROM orders
LIMIT 10;
```

11) Use the **accounts** table to find all the companies whose names start with 'C'.

```
SELECT name
FROM accounts
WHERE name LIKE 'C%';
```

12) Use the **accounts** table to find all companies whose names contain the string 'one' somewhere in the name.

```
SELECT name
FROM accounts
WHERE name LIKE '%one%';
```

13) Use the **accounts** table to find all companies whose names end with 's'.

```
SELECT name
FROM accounts
WHERE name LIKE '%s';
```

14) Use the **accounts** table to find the account name, primary_poc, and sales_rep_id for Walmart, Target, and Nordstrom.

```
SELECT name, primary_poc, sales_rep_id
FROM accounts
WHERE name IN ('Walmart', 'Target', 'Nordstrom');
```

15) Use the **web_events** table to find all information regarding individuals who were contacted via the **channel** of organic or adwords.

```
SELECT *
FROM web_events
WHERE channel IN ('organic', 'adwords');
```

16) Use the **accounts** table to find the account name, primary poc, and sales rep id for all stores except Walmart, Target, and Nordstrom.

```
SELECT name, primary_poc, sales_rep_id
FROM accounts
WHERE name NOT IN ('Walmart', 'Target', 'Nordstrom');
```

17) Use the **web_events** table to find all information regarding individuals who were contacted via any method except using organic or adwords methods.

```
SELECT *
FROM web_events
WHERE channel NOT IN ('organic', 'adwords');
```

18) Use the **accounts** table to find all the companies whose names do not start with 'C'.

```
SELECT name
FROM accounts
WHERE name NOT LIKE 'C%';
```

19) Write a query that returns all the orders where the standard_qty is over 1000, the poster_qty is 0, and the gloss_qty is 0.

```
SELECT *
FROM orders
WHERE standard_qty > 1000 AND poster_qty = 0 AND gloss_qty = 0;
```

20) Using the **accounts** table, find all the companies whose names do not start with 'C' and end with 's'.

```
SELECT name
FROM accounts
WHERE name NOT LIKE 'C%' AND name LIKE '%s';
```

21) When you use the BETWEEN operator in SQL, do the results include the values of your endpoints, or not? Figure out the answer to this important question by writing a query that displays the order date and gloss_qty data for all orders where gloss_qty is between 24 and 29.

```
SELECT occurred_at, gloss_qty
FROM orders
WHERE gloss_qty BETWEEN 24 AND 29;

You should notice that there are a number of rows in the output of this
query where the gloss_qty values are 24 or 29. So the answer to the que
stion is that yes, the BETWEEN operator in SQL is inclusive; that is, th
e endpoint values are included. So the BETWEEN statement in this query i
s equivalent to having written "WHERE gloss_qty >= 24 AND gloss_qty <= 2
9."
```

22) Use the **web_events** table to find all information regarding individuals who were contacted via the organic or adwords channels, and started their account at any point in 2016, sorted from newest to oldest.

```
SELECT *
FROM web_events
WHERE channel IN ('organic', 'adwords') AND occurred_at BETWEEN '20
16-01-01' AND '2017-01-01'
ORDER BY occurred_at DESC;

You will notice that using BETWEEN is tricky for dates! While BETWEEN is
generally inclusive of endpoints, it assumes the time is at 00:00:00 (i.
e. midnight) for dates. This is the reason why we set the right-side end
point of the period at '2017-01-01'.
```

23) Write a query that returns a list of **orders** where the standard_qty is zero and either the gloss_qty or poster_qty is over 1000.

```
SELECT *
FROM orders
WHERE standard_qty = 0 AND
          (gloss_qty > 1000 OR poster_qty > 1000);
```

24) Find all the company names that start with a 'C' or 'W', and the primary contact **contains** 'ana' or 'Ana', but it doesn't contain 'eana'.

```
SELECT *
FROM accounts
WHERE (name LIKE 'C%' OR name LIKE 'W%') AND
           ((primary_poc LIKE '%ana%' OR primary_poc LIKE '%Ana%')
       AND primary_poc NOT LIKE '%eana%');
```

25) Pull all the data from the **accounts** table, and all the data from the **orders** table.ta from the **accounts** table, and all the data from the **orders** table.

```
SELECT orders.*, accounts.*
FROM accounts
JOIN orders
ON accounts.id = orders.account_id;

this result is the same as if you switched the tables in the FROM and JO
IN. Additionally, which side of the = a column is listed doesn't matter.
```

26) Pull the **standard_qty**, **gloss_qty**, and **poster_qty** from the **orders** table, and the **website** and the **primary_poc** from the **accounts** table.

```
SELECT orders.standard_qty, orders.gloss_qty,
       orders.poster_qty,  accounts.website,
       accounts.primary_poc
FROM orders
JOIN accounts
ON orders.account_id = accounts.id
```

27) Provide a table for all **web_events** associated with account **name** of Walmart. There should be three columns. Be sure to include the primary_poc, time of the event, and the channel for each event. Additionally, you might choose to add a fourth column to assure only Walmart events were chosen.

```
SELECT a.primary_poc, w.occurred_at, w.channel, a.name
FROM web_events w
JOIN accounts a
ON w.account_id = a.id
WHERE a.name = 'Walmart';
```

28) Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```
SELECT r.name region, s.name rep, a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
ORDER BY a.name;
```

29) Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. A few accounts have 0 for **total**, so I divided by (total + 0.01) to assure not dividing by zero.

```
SELECT r.name region, a.name account,
       o.total_amt_usd/(o.total + 0.01) unit_price
FROM region r
JOIN sales_reps s
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
ON o.account_id = a.id;
```

30) Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. This time only for the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```
SELECT r.name region, s.name rep, a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
WHERE r.name = 'Midwest'
ORDER BY a.name;
```

31) Provide a table that provides the **region** for each **sales_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a **last** name starting with K and in the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```
SELECT r.name region, s.name rep, a.name account
FROM sales_reps s
JOIN region r
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
WHERE r.name = 'Midwest' AND s.name LIKE '% K%'
ORDER BY a.name;
```

32) Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total_amt_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100. Your final table should have 3 columns: **region name**, **account name**, and **unit price**.

```
SELECT r.name region, a.name account,
       o.total_amt_usd/(o.total + 0.01) unit_price
FROM region r
JOIN sales_reps s
ON s.region_id = r.id
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
ON o.account_id = a.id
WHERE o.standard_qty > 100;
```

33) What are the different **channel**s used by **account id** 1001? Your final table should have only 2 columns: **account name** and the different **channel**s.

```
SELECT DISTINCT a.name, w.channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
WHERE a.id = '1001';
```

34) Find all the orders that occurred in 2015. Your final table should have 4 columns: **occurred_at**, **account name**, **order total**, and **order total_amt_usd**.

```
SELECT o.occurred_at, a.name, o.total, o.total_amt_usd
FROM accounts a
JOIN orders o
ON o.account_id = a.id
WHERE o.occurred_at BETWEEN '01-01-2015' AND '01-01-2016'
ORDER BY o.occurred_at DESC;
```

35) Find the total amount of **poster_qty** paper ordered in the **orders** table.

```
SELECT SUM(poster_qty) AS total_poster_sales
FROM orders;
```

36) Find the total amount for each individual order that was spent on **standard** and **gloss** paper in the orders table. This should give a dollar amount for each order in the table.

```
SELECT standard_amt_usd + gloss_amt_usd AS total_standard_gloss
FROM orders;
```

Notice, this solution did not use an aggregate.

37) Find the **standard_amt_usd** per unit of **standard_qty** paper. Your solution should use both an aggregation and a mathematical operator.

```
SELECT SUM(standard_amt_usd)/SUM(standard_qty) AS standard_price_per
_unit
FROM orders;
```

38) When was the earliest order ever placed?

```
SELECT MIN(occurred_at)
FROM orders;
```

39) Try performing the same query as in question 1 without using an aggregation function.

```
SELECT occurred_at
FROM orders
ORDER BY occurred_at
LIMIT 1;
```

40) When did the most recent (latest) **web_event** occur?

```
SELECT MAX(occurred_at)
FROM web_events;
```

41) Find the mean (**AVERAGE**) amount spent per order on each paper type, as well as the mean amount of each paper type purchased per order. Your final answer should have 6 values - one for each paper type for the average number of sales, as well as the average amount.

```
SELECT AVG(standard_qty) mean_standard,
        AVG(gloss_qty) mean_gloss,
        AVG(poster_qty) mean_poster,
        AVG(standard_amt_usd) mean_standard_usd,
        AVG(gloss_amt_usd) mean_gloss_usd,
        AVG(poster_amt_usd) mean_poster_usd
FROM orders;
```

42) Which **account** (by name) placed the earliest order? Your solution should have the **account name** and the **date** of the order.

```
SELECT a.name, o.occurred_at
FROM accounts a
JOIN orders o
ON a.id = o.account_id
ORDER BY occurred_at
LIMIT 1;
```

43) Find the total sales in **usd** for each account. You should include two columns - the total sales for each company's orders in **usd** and the company **name**.

```
SELECT a.name, SUM(total_amt_usd) total_sales
FROM orders o
JOIN accounts a
ON a.id = o.account_id
GROUP BY a.name;
```

44) Via what **channel** did the most recent (latest) **web_event** occur, which **account** was associated with this **web_event**? Your query should return only three values - the **date**, **channel**, and **account name**.

```
SELECT w.occurred_at, w.channel, a.name
FROM web_events w
JOIN accounts a
ON w.account_id = a.id
ORDER BY w.occurred_at DESC
LIMIT 1;
```

45) Find the total number of times each type of **channel** from the **web_events** was used. Your final table should have two columns - the **channel** and the number of times the channel was used.

```
SELECT w.channel, COUNT(*)
FROM web_events w
GROUP BY w.channel
```

46) Who was the **primary contact** associated with the earliest **web_event**?

```
SELECT a.primary_poc
FROM web_events w
JOIN accounts a
ON a.id = w.account_id
ORDER BY w.occurred_at
LIMIT 1;
```

47) What was the smallest order placed by each **account** in terms of **total usd**. Provide only two columns - the account **name** and the **total usd**. Order from smallest dollar amounts to largest.

```
SELECT a.name, MIN(total_amt_usd) smallest_order
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
ORDER BY smallest_order;
```

48) Find the number of **sales reps** in each region. Your final table should have two columns - the **region** and the number of **sales_reps**. Order from fewest reps to most reps.

```
SELECT r.name, COUNT(*) num_reps
FROM region r
JOIN sales_reps s
ON r.id = s.region_id
GROUP BY r.name
ORDER BY num_reps;
```

49) 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
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name;
```

50) 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
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name, w.channel
ORDER BY num_events DESC;
```

51) 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
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
JOIN sales_reps s
ON s.id = a.sales_rep_id
JOIN region r
ON r.id = s.region_id
GROUP BY r.name, w.channel
ORDER BY num_events DESC;
```

52) Use **DISTINCT** to test if there are any accounts associated with more than one region.

The below two queries have the same number of resulting rows (351), so we know that every account is associated with only one region. If each account was associated with more than one region, the first query should have returned more rows than the second query.

```
SELECT a.id as "account id", r.id as "region id",
a.name as "account name", r.name as "region name"
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
JOIN region r
ON r.id = s.region_id;
```

and

```
SELECT DISTINCT id, name
FROM accounts;
```

53) Have any **sales reps** worked on more than one account?

*Actually all of the sales reps have worked on more than one account. The fewest number of accounts any sales rep works on is 3. There are 50 sales reps, and they all have more than one account. Using **DISTINCT** in the second query assures that all of the sales reps are accounted for in the first query.*

```
SELECT s.id, s.name, COUNT(*) num_accounts
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.id, s.name
ORDER BY num_accounts;
```

And

```
SELECT DISTINCT id, name
FROM sales_reps;
```

54) How many of the **sales reps** have more than 5 accounts that they manage?

```
SELECT s.id, s.name, COUNT(*) num_accounts
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.id, s.name
HAVING COUNT(*) > 5
ORDER BY num_accounts;
```

55) Which account has the most orders?

```
SELECT a.id, a.name, COUNT(*) num_orders
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.id, a.name
ORDER BY num_orders DESC
LIMIT 1;
```

56) 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
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.id, a.name
HAVING SUM(o.total_amt_usd) > 30000
ORDER BY total_spent;
```

57) Which account has spent the most with us?

```
SELECT a.id, a.name, SUM(o.total_amt_usd) total_spent
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.id, a.name
ORDER BY total_spent DESC
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
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
GROUP BY a.id, a.name, w.channel
HAVING COUNT(*) > 6 AND w.channel = 'facebook'
ORDER BY use_of_channel;
```

59) Which account used facebook most as a **channel**?

```
SELECT a.id, a.name, w.channel, COUNT(*) use_of_channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
WHERE w.channel = 'facebook'
GROUP BY a.id, a.name, w.channel
ORDER BY use_of_channel DESC
LIMIT 1;
```

60) Which channel was most frequently used by most accounts?

```
SELECT a.id, a.name, w.channel, COUNT(*) use_of_channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
GROUP BY a.id, a.name, w.channel
ORDER BY use_of_channel DESC
LIMIT 10;
```

61) 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;
```

62) 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
FROM orders
WHERE occurred_at BETWEEN '2014-01-01' AND '2017-01-01'
GROUP BY 1
ORDER BY 2 DESC;
```

The greatest sales amounts occur in December (12).

63) 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
FROM orders
GROUP BY 1
ORDER BY 2 DESC;
```

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.

64) 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
FROM orders
WHERE occurred_at BETWEEN '2014-01-01' AND '2017-01-01'
GROUP BY 1
ORDER BY 2 DESC;
```

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.

65) 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
FROM orders o
JOIN accounts a
ON a.id = o.account_id
WHERE a.name = 'Walmart'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

66) Write a query to display for each order, the account ID, total amount of the order, and the level of the order - 'Large' or 'Small' - depending on if the order is $3000 or more, or less than $3000

```
SELECT account_id, total_amt_usd,
CASE WHEN total_amt_usd > 3000 THEN 'Large'
ELSE 'Small' END AS order_level
FROM orders;
```

67) Write a query to display the number of orders in each of three categories, based on the total number of items in each order. The three categories are: 'At Least 2000', 'Between 1000 and 2000' and 'Less than 1000'.

```
SELECT CASE WHEN total >= 2000 THEN 'At Least 2000'
   WHEN total >= 1000 AND total < 2000 THEN 'Between 1000 and 2000'
   ELSE 'Less than 1000' END AS order_category,
COUNT(*) AS order_count
FROM orders
GROUP BY 1;
```

68) 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,
     CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
     WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
     ELSE 'low' END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY a.name
ORDER BY 2 DESC;
```

69) 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,
     CASE
       WHEN SUM(total_amt_usd) > 200000 THEN 'top'
       WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
     ELSE 'low'
     END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
WHERE occurred_at > '2015-12-31'
GROUP BY 1
ORDER BY 2 DESC;
```

70) 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,
     CASE
       WHEN COUNT(*) > 200 THEN 'top'
       ELSE 'not'
     END AS sales_rep_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name
ORDER BY 2 DESC;
```

71) 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,
CASE
   WHEN COUNT(*) > 200 OR SUM(o.total_amt_usd) > 750000 THEN 'top'
   WHEN COUNT(*) > 150 OR SUM(o.total_amt_usd) > 500000 THEN 'midd
le'
   ELSE 'low'
END AS sales_rep_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name
ORDER BY 3 DESC;
```

72) Provide the **name** of the **sales_rep** in each **region** with the largest amount of **total_amt_usd** sales

```
SELECT t3.rep_name, t3.region_name, t3.total_amt
FROM(SELECT region_name, MAX(total_amt) total_amt
      FROM(SELECT s.name rep_name, r.name region_name,
                  SUM(o.total_amt_usd) total_amt
             FROM sales_reps s
             JOIN accounts a
             ON a.sales_rep_id = s.id
             JOIN orders o
             ON o.account_id = a.id
             JOIN region r
             ON r.id = s.region_id
             GROUP BY 1, 2) t1
     GROUP BY 1) t2
JOIN (SELECT s.name rep_name, r.name region_name,
             SUM(o.total_amt_usd) total_amt
      FROM sales_reps s
      JOIN accounts a
      ON a.sales_rep_id = s.id
      JOIN orders o
      ON o.account_id = a.id
      JOIN region r
      ON r.id = s.region_id
      GROUP BY 1,2
      ORDER BY 3 DESC) t3
ON t3.region_name = t2.region_name AND t3.total_amt = t2.total_amt;
```

73) For the region with the largest sales **total_amt_usd**, how many **total** orders were placed?

```
SELECT r.name, COUNT(o.total) total_orders
FROM sales_reps s
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
ON o.account_id = a.id
JOIN region r
ON r.id = s.region_id
GROUP BY r.name
HAVING SUM(o.total_amt_usd) = (
      SELECT MAX(total_amt)
      FROM (SELECT r.name region_name,
                   SUM(o.total_amt_usd) total_amt
              FROM sales_reps s
              JOIN accounts a
              ON a.sales_rep_id = s.id
              JOIN orders o
              ON o.account_id = a.id
              JOIN region r
              ON r.id = s.region_id
              GROUP BY r.name) sub);
```

74) **How many accounts** had more **total** purchases than the account **name** which has bought the most **standard_qty** paper throughout their lifetime as a customer?

```
SELECT COUNT(*)
FROM (SELECT a.name
       FROM orders o
       JOIN accounts a
       ON a.id = o.account_id
       GROUP BY 1
       HAVING SUM(o.total) > (SELECT total
                              FROM (SELECT a.name act_name, SUM(o.
standard_qty) tot_std, SUM(o.total) total
                                     FROM accounts a
                                     JOIN orders o
                                     ON o.account_id = a.id
                                     GROUP BY 1
                                     ORDER BY 2 DESC
                                     LIMIT 1) inner_tab)
        ) counter_tab;
```

75) For the customer that spent the most (in total over their lifetime as a customer) **total_amt_usd**, how many **web_events** did they have for each channel?

```
SELECT a.name, w.channel, COUNT(*)
FROM accounts a
JOIN web_events w
ON a.id = w.account_id AND
    a.id = (SELECT id
            FROM (SELECT a.id, a.name, SUM(o.total_amt_usd) tot_spent
                    FROM orders o
                    JOIN accounts a
                    ON a.id = o.account_id
                    GROUP BY a.id, a.name
                    ORDER BY 3 DESC
                    LIMIT 1) inner_table)
GROUP BY 1, 2
ORDER BY 3 DESC;
```

76) What is the lifetime average amount spent in terms of **total_amt_usd** for the top 10 total spending **accounts**?

```
SELECT AVG(tot_spent)
FROM  (SELECT a.id, a.name, SUM(o.total_amt_usd) tot_spent
        FROM orders o
        JOIN accounts a
        ON a.id = o.account_id
        GROUP BY a.id, a.name
        ORDER BY 3 DESC
        LIMIT 10) temp;
```

77) What is the lifetime average amount spent in terms of **total_amt_usd**, including only the companies that spent more per order, on average, than the average of all orders.

```
SELECT AVG(avg_amt)
FROM (SELECT o.account_id, AVG(o.total_amt_usd) avg_amt
        FROM orders o
        GROUP BY 1
        HAVING AVG(o.total_amt_usd)>(SELECT AVG(o.total_amt_usd)
                                              AS avg_all
                                       FROM orders o)) temp_table;
```

78) Provide the **name** of the **sales_rep** in each **region** with the largest amount of **total_amt_usd** sales.

```
WITH t1 AS (
  SELECT s.name rep_name, r.name region_name,
          SUM(o.total_amt_usd) total_amt
  FROM sales_reps s
  JOIN accounts a
  ON a.sales_rep_id = s.id
  JOIN orders o
  ON o.account_id = a.id
  JOIN region r
  ON r.id = s.region_id
  GROUP BY 1,2
  ORDER BY 3 DESC), t2 AS (
                              SELECT region_name,
                                      MAX(total_amt) total_amt
                              FROM t1
                              GROUP BY 1)

SELECT t1.rep_name, t1.region_name, t1.total_amt
FROM t1
JOIN t2
ON t1.region_name = t2.region_name AND t1.total_amt = t2.total_amt;
```

79) For the account that purchased the most (in total over their lifetime as a customer) **standard_qty** paper, **how many accounts** still had more in **total** purchases?

```
WITH t1 AS (
  SELECT a.name account_name, SUM(o.standard_qty) total_std,
          SUM(o.total) total
  FROM accounts a
  JOIN orders o
  ON o.account_id = a.id
  GROUP BY 1
  ORDER BY 2 DESC
  LIMIT 1), t2 AS (SELECT a.name
                    FROM orders o
                    JOIN accounts a
                    ON a.id = o.account_id
                    GROUP BY 1
                    HAVING SUM(o.total) > (SELECT total FROM t1))

SELECT COUNT(*)
FROM t2;
```

80) For the region with the largest sales **total_amt_usd**, how many **total** orders were placed?

```
WITH t1 AS (
   SELECT r.name region_name, SUM(o.total_amt_usd) total_amt
   FROM sales_reps s
   JOIN accounts a
   ON a.sales_rep_id = s.id
   JOIN orders o
   ON o.account_id = a.id
   JOIN region r
   ON r.id = s.region_id
   GROUP BY r.name), t2 AS (
                              SELECT MAX(total_amt)
                              FROM t1)

SELECT r.name, COUNT(o.total) total_orders
FROM sales_reps s
JOIN accounts a
ON a.sales_rep_id = s.id
JOIN orders o
ON o.account_id = a.id
JOIN region r
ON r.id = s.region_id
GROUP BY r.name
HAVING SUM(o.total_amt_usd) = (SELECT * FROM t2);
```

81) For the customer that spent the most (in total over their lifetime as a customer) **total_amt_usd**, how many **web_events** did they have for each channel?

```
WITH t1 AS (
   SELECT a.id, a.name, SUM(o.total_amt_usd) tot_spent
   FROM orders o
   JOIN accounts a
   ON a.id = o.account_id
   GROUP BY a.id, a.name
   ORDER BY 3 DESC
   LIMIT 1)

SELECT a.name, w.channel, COUNT(*)
FROM accounts a
JOIN web_events w
ON a.id = w.account_id AND a.id =  (SELECT id FROM t1)
GROUP BY 1, 2
ORDER BY 3 DESC;
```

82) What is the lifetime average amount spent in terms of **total_amt_usd** for the top 10 total spending **accounts**?

```
WITH t1 AS (
   SELECT a.id, a.name, SUM(o.total_amt_usd) tot_spent
   FROM orders o
   JOIN accounts a
   ON a.id = o.account_id
   GROUP BY a.id, a.name
   ORDER BY 3 DESC
   LIMIT 10)

SELECT AVG(tot_spent)
FROM t1;
```

83) What is the lifetime average amount spent in terms of **total_amt_usd**, including only the companies that spent more per order, on average, than the average of all orders.

```
WITH t1 AS (
   SELECT AVG(o.total_amt_usd) avg_all
   FROM orders o
   JOIN accounts a
   ON a.id = o.account_id
), t2 AS (SELECT o.account_id,
                  AVG(o.total_amt_usd) avg_amt
          FROM orders o
          GROUP BY 1
          HAVING AVG(o.total_amt_usd) > (SELECT * FROM t1))

SELECT AVG(avg_amt)
FROM t2;
```

84) In the **accounts** table, there is a column holding the **website** for each company. The last three digits specify what type of web address they are using. A list of extensions (and pricing) is provided here. Pull these extensions and provide how many of each website type exist in the **accounts** table.

```
SELECT RIGHT(website, 3) AS domain, COUNT(*) num_companies
FROM accounts
GROUP BY 1
ORDER BY 2 DESC;
```

85) There is much debate about how much the name **(or even the first letter of a company name)** matters. Use the **accounts** table to pull the first letter of each company name to see the distribution of company names that begin with each letter (or number).

```
SELECT LEFT(UPPER(name), 1) AS first_letter, COUNT(*) num_companies
FROM accounts
GROUP BY 1
ORDER BY 2 DESC;
```

86) Use the **accounts** table and a **CASE** statement to create two groups: one group of company names that start with a number and a second group of those company names that start with a letter. What proportion of company names start with a letter?

```
SELECT SUM(num) nums, SUM(letter) letters
FROM (SELECT name,
       CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5',
'6','7','8','9') THEN 1 ELSE 0 END AS num,
       CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5',
'6','7','8','9') THEN 0 ELSE 1 END AS letter
      FROM accounts) t1;
```

87) Consider vowels as a, e, i, o, and u. What proportion of company names start with a vowel, and what percent start with anything else?

```
SELECT SUM(vowels) vowels, SUM(other) other
FROM (SELECT name,
            CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
                 THEN 1 ELSE 0 END AS vowels,
            CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
                 THEN 0 ELSE 1 END AS other
         FROM accounts) t1;
```

88) Use the accounts table to create **first** and **last** name columns that hold the first and last names for the primary_poc.

```
SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1 ) first_name,
     RIGHT(primary_poc, LENGTH(primary_poc) –
          STRPOS(primary_poc, ' ')) last_name
FROM accounts;
```

89) Now see if you can do the same thing for every rep name in the sales_reps table. Again provide **first** and **last** name columns.

```
SELECT LEFT(name, STRPOS(name, ' ') -1 ) first_name,
       RIGHT(name, LENGTH(name) - STRPOS(name, ' ')) last_name
FROM sales_reps;
```

90) Each company in the accounts table wants to create an email address for each primary_poc. The email address should be the first name of the **primary_poc** . last name **primary_poc** @ company name .com.

```
WITH t1 AS (
     SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1) first_name,
             RIGHT(primary_poc, LENGTH(primary_poc) –
                    STRPOS(primary_poc, ' ')) last_name, name
     FROM accounts)

SELECT first_name, last_name,
        CONCAT(first_name, '.', last_name, '@', name, '.com')
FROM t1;
```

91) You may have noticed that in the previous solution some of the company names include spaces, which will certainly not work in an email address. See if you can create an email address that will work by removing all of the spaces in the account name, but otherwise your solution should be just as in question 1. Some helpful documentation is **here**.

```
WITH t1 AS (
 SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1 ) first_name,
         RIGHT(primary_poc, LENGTH(primary_poc) –
               STRPOS(primary_poc, ' ')) last_name, name
 FROM accounts)

SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@',
REPLACE(name, ' ', ''), '.com')
FROM  t1;
```

92) We would also like to create an initial password, which they will change after their first log in. The first password will be the first letter of the primary_poc's first name (lowercase), then the last letter of their first name (lowercase), the first letter of their last name (lowercase), the last letter of their last name (lowercase), the number of letters in their first name, the number of letters in their last name, and then the name of the company they are working with, all capitalized with no spaces.

```
WITH t1 AS (
  SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1 ) first_name,
        RIGHT(primary_poc, LENGTH(primary_poc) –
              STRPOS(primary_poc, ' ')) last_name, name
  FROM accounts)

SELECT first_name, last_name,
     CONCAT(first_name, '.', last_name, '@', name, '.com'),
      LEFT(LOWER(first_name), 1) || RIGHT(LOWER(first_name), 1) ||
      LEFT(LOWER(last_name), 1) ||
      RIGHT(LOWER(last_name), 1) ||
      LENGTH(first_name) || LENGTH(last_name) ||
      REPLACE(UPPER(name), ' ', '')
FROM t1;
```