1. Find the total amount of poster_qty paper ordered in the orders table.

```
SELECT sum(poster_qty)
FROM orders
```

2. Find the total amount of standard_qty paper ordered in the orders table.

```
SELECT sum(standard_qty)
FROM orders
```

3. Find the total dollar amount of sales using the total_amt_usd in the orders table.

```
SELECT sum(total_amt_usd) AS total_dollar_sales
FROM orders
```

4. Find the total amount spent on standard_amt_usd and gloss_amt_usd paper for each order in the orders table. This should give a dollar amount for each order in the table.

```
SELECT standard_amt_usd + gloss_amt_usd AS standard_gloss
FROM orders
```

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

```
SELECT ROUND(SUM(standard_amt_usd)/SUM(standard_qty),2)
unit_price
FROM orders
```

6. When was the earliest order ever placed? You only need to return the date.

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

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

```
SELECT occurred_at
```

```
FROM orders
ORDER BY 1
LIMIT 1
```

8. When did the most recent (latest) web_event occur?

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

9. Try to perform the result of the previous query without using an aggregation function.

```
SELECT occurred_at
FROM web_events
ORDER BY 1 DESC
LIMIT 1
```

10. 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) standard_average_qty,
       AVG(standard_amt_usd) standard_average_usd,
       AVG(gloss_qty) gloss_average_qty,
       AVG(gloss_amt_usd) gloss_average_usd,
       AVG(poster_qty) poster_average_qty,
       AVG(poster_amt_usd) poster_average_usd
FROM orders
```

11. Via the video, you might be interested in how to calculate the MEDIAN. Though this is more advanced than what we have covered so far try finding - what is the MEDIAN total_usd spent on all orders?

Solution 1

```
SELECT *
FROM (
       SELECT total_amt_usd
       FROM orders
       ORDER BY total_amt_usd
       LIMIT 3457
```

```
     ) AS table1
ORDER BY total_amt_usd DESC
LIMIT 2;
```

Solution 2

```
SELECT round(avg(total_amt_usd),2) median_total_amt_usd
FROM (
       SELECT total_amt_usd
       FROM (
              SELECT total_amt_usd,
                     COUNT(*) OVER() AS ROW_COUNT,
                     row_number() over(ORDER BY
total_amt_usd) AS row_number
              FROM orders
            ) x
WHERE row_number IN ((ROW_COUNT + 1) / 2,(ROW_COUNT + 2) /
2) ) y
```

Helpful link -> https://www.compose.com/articles/metrics-maven-meet-in-the-middle-median-in-postgresql/

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

```
SELECT  a.name account_name,
        o.occurred_at order_date
FROM accounts a
JOIN orders o ON a.id = o.account_id
ORDER BY 2
LIMIT 1
```

13. 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 account_name,
        SUM(o.total_amt_usd) total_sale
FROM accounts a
JOIN orders o ON a.id = o.account_id
GROUP BY 1
```

14. 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  a.name,
        w.occurred_at,
        w.channel
FROM web_events w
JOIN accounts a ON w.account_id = a.id
ORDER BY 2 DESC
LIMIT 1
```

15. 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  COUNT(*) no_of_times,
        channel
FROM web_events
GROUP BY 2
```

16. Who was the primary contact associated with the
earliest web_event?

```
SELECT  a.primary_poc,
        w.occurred_at
FROM accounts a
JOIN web_events w ON w.account_id = a.id
ORDER BY 2
LIMIT 1
```

17. 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 account_name,
       MIN(o.total_amt_usd) min_order_usd
FROM accounts a
JOIN orders o ON a.id = o.account_id
GROUP BY 1
ORDER BY 2
```

18. 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 region,
        COUNT(s.name) reps
FROM sales_reps s
JOIN region r ON r.id = s.region_id
GROUP BY 1
```

19. 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 quantity purchased for each of the paper types for each account.

```
SELECT  a.name,
        ROUND(AVG(o.standard_qty),2) standard_avg,
        ROUND(AVG(o.poster_qty),2) poster_avg,
        ROUND(AVG(o.gloss_qty),2) gloss_avg
FROM accounts a
JOIN orders o ON a.id = o.account_id
GROUP BY 1
ORDER BY 1
```

20. 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,
        ROUND(AVG(o.standard_amt_usd),2) standard_avg_usd,
        ROUND(AVG(o.poster_amt_usd),2) poster_avg_usd,
        ROUND(AVG(o.gloss_amt_usd),2) gloss_avg_usd
FROM accounts a
JOIN orders o ON a.id = o.account_id
GROUP BY 1
ORDER BY 1
```

21. 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(*) no_of_times
FROM web_events w
JOIN accounts a ON w.account_id = a.id
JOIN sales_reps s ON a.sales_rep_id = s.id
GROUP BY 1, 2
ORDER BY 3 DESC
```

22. 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(*) no_of_times
FROM web_events w
JOIN accounts a ON w.account_id = a.id
JOIN sales_reps s ON a.sales_rep_id = s.id
JOIN region r ON r.id = s.region_id
GROUP BY 1,2
ORDER BY 3 DESC
```

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

```
SELECT DISTINCT a.name accounts,
       r.name region
FROM accounts a
JOIN sales_reps s ON a.sales_rep_id = s.id
JOIN region r ON r.id = s.region_id
ORDER BY 1

-- with and without distinct both queries return 351 rows.
Therefore each accounts are unique to a disticnt region.
```

24. Have any sales reps worked on more than one account?

```
SELECT s.name rep,
       COUNT(*) no_of_accounts
FROM accounts a
```

```
JOIN sales_reps s ON a.sales_rep_id = s.id
GROUP BY 1
ORDER BY 1

-- yes, the reps has more than one account.
```

25. How many of the sales reps have more than 5 accounts that they manage?

```
SELECT COUNT(*)
FROM (
       SELECT s.name,
       COUNT(*) no_of_accounts
       FROM sales_reps s
       JOIN accounts a ON s.id = a.sales_rep_id
       GROUP BY 1
       HAVING COUNT(*) > 5
     ) x
```

26. How many accounts have more than 20 orders?

```
SELECT COUNT(*)
FROM (
       SELECT a.name,
       COUNT(*)
       FROM accounts a
       JOIN orders o ON o.account_id = a.id
       GROUP BY 1
       HAVING COUNT(*) > 20
     ) x
```

27. Which account has the most orders?

```
SELECT a.name,
       COUNT(*)
FROM orders o
JOIN accounts a ON a.id = o.account_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1
```

28. Which accounts spent more than 30,000 usd total across all orders?

```
SELECT a.name,
       SUM(total_amt_usd)
FROM orders o
JOIN accounts a ON a.id = o.account_id
GROUP BY 1
HAVING sum(total_amt_usd) > 30000
ORDER BY 2 DESC
```

29. Which accounts spent less than 1,000 usd total across all orders?

```
SELECT a.name,
       SUM(total_amt_usd)
FROM orders o
JOIN accounts a ON a.id = o.account_id
GROUP BY 1
HAVING sum(total_amt_usd) < 1000
ORDER BY 2 DESC
```

30. Which account has spent the most with us?

```
SELECT  a.name,
        SUM(total_amt_usd)
FROM orders o
JOIN accounts a ON a.id = o.account_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1
```

31. Which account has spent the least with us?

```
SELECT a.name,
       SUM(total_amt_usd)
FROM orders o
JOIN accounts a ON a.id = o.account_id
GROUP BY 1
ORDER BY 2
LIMIT 1
```

32. Which accounts used facebook as a channel to contact customers more than 6 times?

```
SELECT  a.name,
        w.channel
FROM accounts a
JOIN web_events w ON a.id = w.account_id
GROUP BY 1, 2
HAVING COUNT(w.channel) > 6
AND channel = 'facebook'        -- notice you can add `and`
to pre-filter in having clause
ORDER BY 1
```

33. Which account used facebook most as a channel?

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

34. Which channel was most frequently used by most
accounts?

```
SELECT  w.channel,
        count(*)
FROM accounts a
JOIN web_events w ON w.account_id = a.id
GROUP BY 1
ORDER BY 2 DESC
```

-- WORKING WITH DATES

35. 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 SUM(total_amt_usd),
       DATE_TRUNC('year',occurred_at)
FROM ORDERS
GROUP BY 2
ORDER BY 1 DESC
```

```
-- why sales for 2017 & 2013 are so small?

SELECT COUNT(*),
       DATE_TRUNC('year',occurred_at)
FROM ORDERS
GROUP BY 2
ORDER BY 1 DESC

-- This query reveals that 2013 have 99 orders and 2017
have only 25 orders
-- Let's look at the month

SELECT COUNT(*),
       DATE_TRUNC('month', occurred_at)
FROM ORDERS
GROUP BY 2
ORDER BY 2 DESC

-- This query reveals that 2013 and 2017 has only one order
recorded.
-- Therefore each year is not evenly represented
```

36. Which month did Parch & Posey have the greatest sales in terms of total dollars? Are all months evenly represented by the dataset?

```
SELECT DATE_PART('month',occurred_at),
       SUM(TOTAL_AMT_USD)
FROM ORDERS
WHERE occurred_at BETWEEN '2014-01-01' AND '2016-12-31'
GROUP BY 1
ORDER BY 2 DESC

-- Since 2013 and 2017 have only one order recorded we will
exclude them to the query.
-- December, November and October has the highest number of
total orders.
```

37. 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),
       COUNT(*)
```

```
FROM ORDERS
GROUP BY 1
ORDER BY 2 DESC

-- 2016 has the highest number of total sales. We are
seeing an upward trend.
```

38. 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),
       COUNT(total)
FROM ORDERS
WHERE occurred_at BETWEEN '2014-01-01' AND '2016-12-31'
GROUP BY 1
ORDER BY 2 DESC

-- Since 2013 and 2017 have only one order recorded we will
exclude them to the query.
-- December, November and October has the highest number of
total orders.
```

39. 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),
       a.name account_name,
       o.gloss_amt_usd total_gloss
FROM orders o
JOIN accounts a ON a.id = o.account_id
AND a.name = 'Walmart'
ORDER BY 3 DESC
LIMIT 1

-- CASE STATEMENTS
```

14. 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.

```
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
```

15. 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 smaller than $3000.

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

16. 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 COUNT(*),
       CASE WHEN TOTAL >= 2000 THEN 'At Least 2000'
            WHEN TOTAL < 2000 AND TOTAL >= 1000 THEN
'Between 1000 and 2000'
            ELSE 'Less than 1000'
       END AS ORDER_LEVEL
FROM ORDERS
GROUP BY 2
```

17. We would like to understand 3 different levels of customers based on the amount associated with their purchases. The top level includes anyone with a Lifetime Value (total sales of all orders) greater than 200,000 usd. The second level is between 200,000 and 100,000 usd. The lowest level 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.

```
WITH t1 AS
        (
         SELECT a.name account_name,
         sum(o.total_amt_usd) AS lifetime_val
         FROM orders o
         JOIN accounts a ON a.id = o.account_id
         GROUP BY 1
         )

SELECT  *,
        CASE WHEN lifetime_val > 200000 THEN 'greater than
200,000'
             WHEN lifetime_val <= 200000 AND lifetime_val
>= 100000 THEN 'between 200,000 and 100,000'
             ELSE 'under 100,000'
        END AS level_of_lifetime_value
FROM t1
```

18. 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 levels as in the previous question. Order with the top spending customers listed first.

```
SELECT a.name account_name,
       SUM(o.total_amt_usd) lifetime_val,
       CASE WHEN SUM(o.total_amt_usd) > 200000 THEN
'greater than 2000'
            WHEN SUM(o.total_amt_usd) <= 200000 AND
SUM(o.total_amt_usd) >= 100000 THEN 'between 200000 and
100000'
       ELSE 'under 100000'
       END AS level_lifetime_value
FROM orders o
JOIN accounts a ON a.id = o.account_id
WHERE o.occurred_at BETWEEN '2016-01-01' AND '2017-12-31'
GROUP BY 1
ORDER BY 2 DESC
```

19. 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.

```
WITH t1 AS
        (
          SELECT s.name rep_name,
          count(*) num_of_orders
          FROM sales_reps s
          JOIN accounts a ON s.id = a.sales_rep_id
          JOIN orders o ON a.id = o.account_id
          GROUP BY 1
          ORDER BY 2 DESC
        )

SELECT rep_name, num_of_orders,
        CASE WHEN num_of_orders >= 200 THEN 'top'
             ELSE 'not'
        END AS rep_level
FROM t1
```

20. 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!

```
WITH t1 AS
        (
          SELECT s.name rep_name,
          COUNT(*) num_of_orders,
          SUM(o.total_amt_usd) total_sales
          FROM orders o
          JOIN accounts a ON a.id = o.account_id
          JOIN sales_reps s ON s.id = a.sales_rep_id
          GROUP BY 1
          ORDER BY 3 DESC
        )

SELECT *,
       CASE WHEN num_of_orders > 200 OR total_sales >
750000 THEN 'top'
            WHEN num_of_orders > 150 OR total_sales >
500000 THEN 'middle'
            ELSE 'low'
       END AS rep_level
FROM t1
```