**Sales data exploration**

1.Provide a table that provides the region for each sales\_rep along with their associated accounts.

This time only for the Midwest region. Sort the accounts alphabetically (A-Z) according to account name.\*/

--As we want the region name, the sales rep name and the account name, we have to work with 3 tables: regions, sales\_reps and accounts

--So we select the name column from the regions table, the name column from the sales\_reps table and the name column from the accounts table

--Then we join the region table with the sales\_reps table and that with the account tables

-- As we only want the Midwest region, we filter with a where clause

-- And finally we sort by the account name

SELECT r.name as Region, sr.name as Rep\_name, ac.name as account\_name

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id

WHERE r.name='Midwest'

ORDER BY ac.name ASC



\*2.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 first name starting with S and in the Midwest region.

Sort the accounts alphabetically (A-Z) according to account name.\*/

Here we add another condition; the first name of the sales rep name should starts with an S.

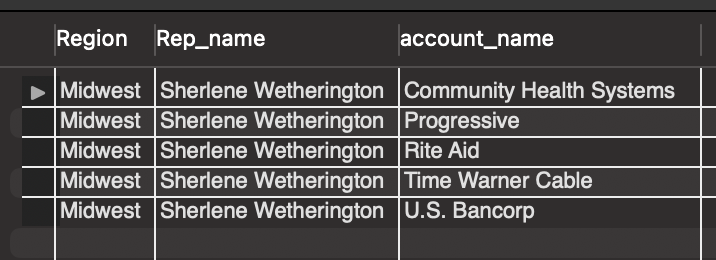
--Therefore, in the where clause we add one more condition

SELECT r.name as Region, sr.name as Rep\_name, ac.name as account\_name

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id

WHERE r.name='Midwest' AND sr.name LIKE 'S%'

ORDER BY ac.name ASC



3. 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.

Sort the accounts alphabetically (A-Z) according to account name.\*/

In this one, the sales rep last name should starts with a K

--Hence, we replace the second condition we had with another one that extracts the records that fullfill the condition above

SELECT r.name as Region, sr.name as Rep\_name, ac.name as account\_name

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id

WHERE r.name='Midwest' AND sr.name LIKE '% K%'

ORDER BY ac.name ASC

A screenshot of a computer

Description automatically generated with medium confidence

4. 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.

In order to avoid a division by zero error, adding .01 to the denominator here is helpful total\_amt\_usd/(total+0.01).\*/

--We want the region, the account name, and the unit price; so we are going to work with the region, the accounts and the orders tables

--We also going to need the sales\_reps table to join the tables mentioned above

--And then, we add a condition: we only want the orders with more than 100 standard paper

SELECT r.name AS region, ac.name AS account\_name, o.total\_amt\_usd/(o.total + 0.01) AS unit\_price

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id JOIN orders o ON ac.id = o.account\_id

WHERE o.standard\_qty >

A picture containing text, screenshot, menu, font

Description automatically generated

5. 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 and the poster order quantity

exceeds 50. Sort for the smallest unit price first.\*/

--To the query above we add a new condition: the orders with more than 100 standard papers and with more than 50 poster papers

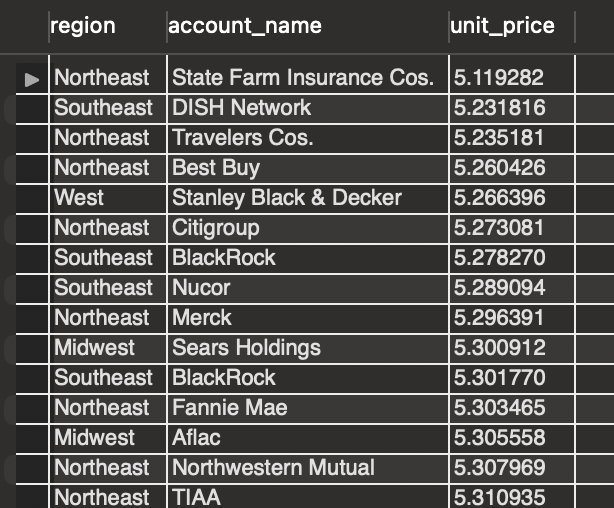
--And then we sort the table to visualize the account name with the lowest unit price

SELECT r.name as region, ac.name as account\_name, o.total\_amt\_usd/(o.total + 0.01) as unit\_price

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id JOIN orders o ON ac.id = o.account\_id

WHERE o.standard\_qty > 100 AND o.poster\_qty > 50

ORDER BY unit\_price ASC



6. 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 and the poster order quantity

exceeds 50. Sort for the largest unit price first.\*/

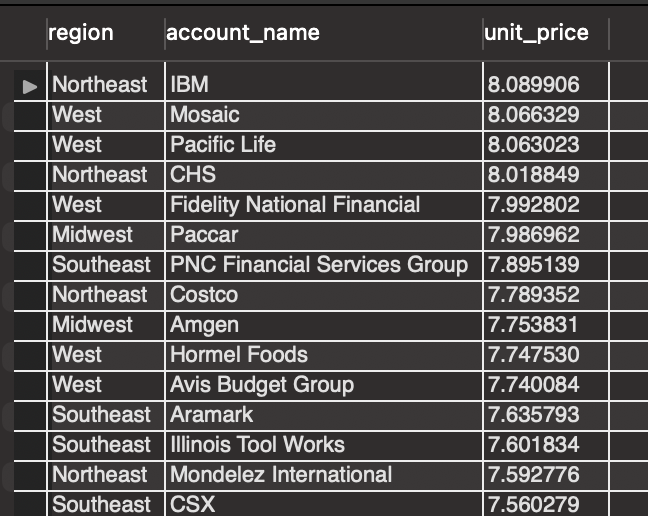
--Here we sort the table to visualize the account name with the highest unit price

SELECT r.name as region, ac.name as account\_name, o.total\_amt\_usd/(o.total + 0.01) as unit\_price

FROM region r JOIN sales\_reps sr ON r.id=sr.region\_id JOIN accounts ac ON sr.id=ac.sales\_rep\_id JOIN orders o ON ac.id = o.account\_id

WHERE o.standard\_qty > 100 AND o.poster\_qty > 50

ORDER BY unit\_price DESC



7. For each account, determine the average amount of each type of paper they purchased across their orders.\*/

--We need the column name of the accounts table, and the mean of each type of paper each one of the accounts purchased

--Thus, we apply the aggregate function avg() in each type of paper

--As we want the account name and the mean of each type of paper each one of the accounts purchased across their orders

--we join the accounts table with the orders table

--Lastly we group by the account\_name

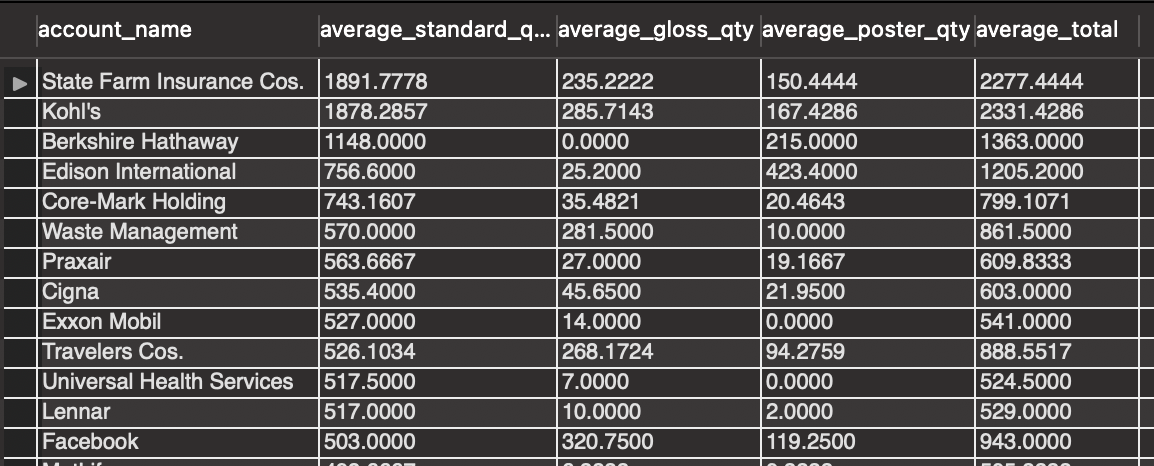
SELECT ac.name AS account\_name, AVG(o.standard\_qty) AS average\_standard\_qty, AVG(o.gloss\_qty) AS average\_gloss\_qty,

AVG(o.poster\_qty) AS average\_poster\_qty, AVG(total) as average\_total

FROM accounts ac JOIN orders o ON ac.id=o.account\_id

GROUP BY ac.name

ORDER BY average\_standard\_qty DESC



8.For each account, determine the average amount spent per order on each paper type. \*/

--Here we do the same but we apply the avg() function on the amount spent on each paper type

SELECT ac.name AS account\_name, AVG(o.standard\_amt\_usd) AS avg\_standard\_amt\_usd, AVG(o.gloss\_amt\_usd) AS avg\_gloss\_amt\_usd,

AVG(o.poster\_amt\_usd) AS avg\_poster\_amt\_usd

FROM accounts ac JOIN orders o ON ac.id=o.account\_id

GROUP BY ac.name

A picture containing text, screenshot, menu, number

Description automatically generated

--In the next query we can visualize the account\_name that had spend the most

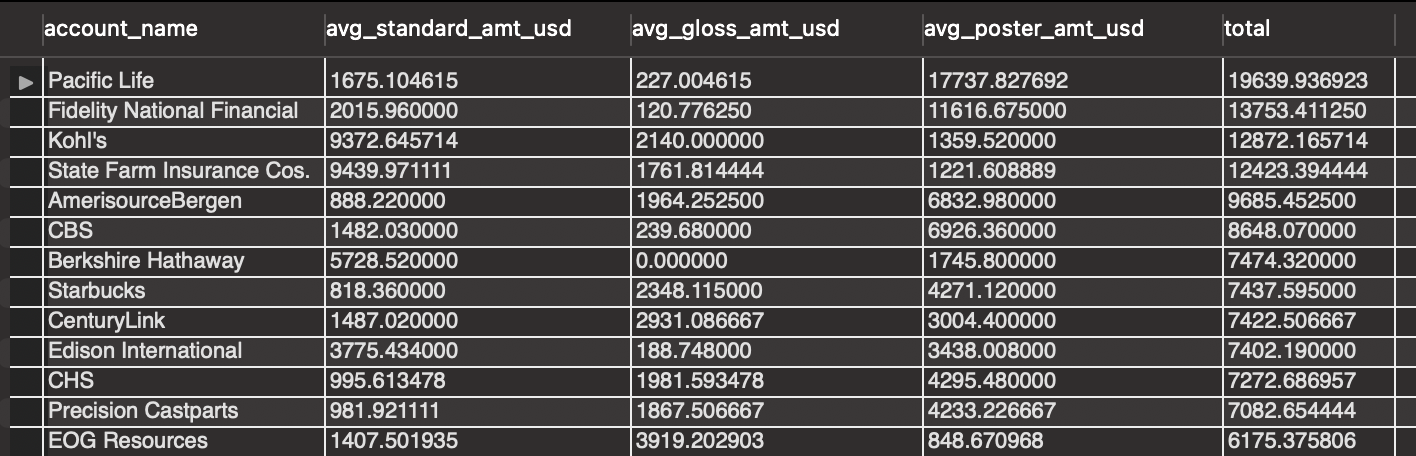
SELECT ac.name AS account\_name, AVG(o.standard\_amt\_usd) AS avg\_standard\_amt\_usd, AVG(o.gloss\_amt\_usd) AS avg\_gloss\_amt\_usd,

AVG(o.poster\_amt\_usd) AS avg\_poster\_amt\_usd, AVG(o.standard\_amt\_usd)+AVG(o.gloss\_amt\_usd)+AVG(o.poster\_amt\_usd) as total

FROM accounts ac JOIN orders o ON ac.id=o.account\_id

GROUP BY ac.name

ORDER BY total DESC



9. Determine the number of times a particular channel was used in the web\_events table for each sales rep.

Order your table with the highest number of occurrences first.\*/

--We want the sales rep name, the channel and the number of occurrences of these channels so we will work with web\_events table, the sales\_reps table

--and inderectly with the accounts table

--Therefore, we select the name column from the sales\_reps table, the channel column from the web\_events table and we apply de count() function

--to the channels

--Then, as we are working with the web\_events and sales\_reps tables we want to join them. But the only way to do this is through the

--accounts table, so we do that.

--We group by the sales rep name and the channel

--And finally we sort by the number of ocurrences

SELECT sr.name AS sales\_rep\_name, we.channel AS channel, count(channel) AS number\_of\_occurrences

FROM web\_events we JOIN accounts ac ON we.account\_id=ac.id JOIN sales\_reps sr ON ac.sales\_rep\_id=sr.id

GROUP BY sr.name, we.channel

ORDER BY number\_of\_occurrences DESC

A picture containing text, screenshot, font, number

Description automatically generated

10. 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.\*/

--Here we can see the yearly totals

--We extract the year of the occurred\_at column from the orders table and apply the sum() function to the total\_amt\_usd also from de orders table

--Group by year and lastly sort it by the total\_usd column

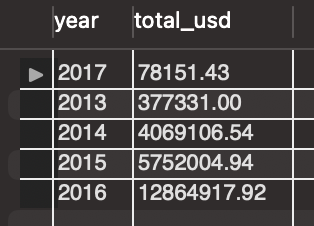
--We can see sales have been increasing year over year

SELECT EXTRACT(YEAR FROM occurred\_at) AS year, SUM(total\_amt\_usd) AS total\_usd

FROM orders

GROUP BY year

ORDER BY total\_usd ASC



--In this one we can see the total by year and month

--AS we did earier, we extract the year of the occurred\_at column and we also extract the month.

--We group by year and month

--We can observe that for 2013 and 2017 there is only one month of sales for each of these years

SELECT EXTRACT(YEAR FROM occurred\_at) AS year, EXTRACT(MONTH FROM occurred\_at) AS month, SUM(total\_amt\_usd) AS total\_usd

FROM orders

WHERE EXTRACT(YEAR FROM occurred\_at) = 2013 OR EXTRACT(YEAR FROM occurred\_at)= 2017

GROUP BY year, month

ORDER BY year

A screenshot of a graph

Description automatically generated with low confidence

--Here we extract the day too

--We take the records from the 2017 year

--We can visualize the total for each day of the month

--We observe that for the 2017 year we only have the first and second day of January

SELECT EXTRACT(YEAR FROM occurred\_at) AS year, EXTRACT(MONTH FROM occurred\_at) AS month, EXTRACT(DAY FROM occurred\_at) AS day,

SUM(total\_amt\_usd) AS total\_usd

FROM orders

WHERE EXTRACT(YEAR FROM occurred\_at)=2017

GROUP BY year, month, day

ORDER BY total\_usd ASC

A picture containing text, font, screenshot, number

Description automatically generated

--Here we compare the total in January 1rst 2017 with the others years

--We add more conditions so we can visualize what we mentioned above

--At this rate, we might expect 2017 to have the largest sales

SELECT EXTRACT(YEAR FROM occurred\_at) AS year, EXTRACT(MONTH FROM occurred\_at) AS month, EXTRACT(DAY FROM occurred\_at) AS day,

SUM(total\_amt\_usd) AS total\_usd

FROM orders

WHERE EXTRACT(MONTH FROM occurred\_at)=1 AND EXTRACT(DAY FROM occurred\_at)=1

GROUP BY year, month, day

ORDER BY total\_usd ASC



-- Lets see the percentage of growth in each year

WITH CTE\_GROWTH AS

(SELECT EXTRACT(YEAR FROM occurred\_at) AS year, EXTRACT(MONTH FROM occurred\_at) AS month, EXTRACT(DAY FROM occurred\_at) AS day,

SUM(total\_amt\_usd) AS total\_usd

FROM orders

WHERE EXTRACT(MONTH FROM occurred\_at)=1 AND EXTRACT(DAY FROM occurred\_at)=1

GROUP BY year, month, day

ORDER BY total\_usd ASC)

SELECT year, month, day,total\_usd, total\_usd - LAG(total\_usd) OVER (ORDER BY year ASC) AS growth,

(total\_usd - LAG (total\_usd) OVER (ORDER BY year ASC))/LAG (total\_usd) OVER (ORDER BY year ASC)\*100 AS percentage\_growth

FROM CTE\_GROWTH

A screenshot of a graph

Description automatically generated with low confidence

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

--We select the name account from the accounts table, extract the year and month from occurre\_at column and apply the sum() function

-- to the gloss\_amt\_usd column from the orders table

--We join the accounts table with the orders table because we need information from both tables

--And we only want what Walmart spend so we add a condition with the where clause

--Then we group by account\_name, year and month

--And we order all by total spend on gloss paper and as we only want the year and month Walmart spend the most, we take only the first row.

SELECT ac.name AS account\_name, EXTRACT(YEAR FROM o.occurred\_at) AS year, EXTRACT(MONTH FROM o.occurred\_at) AS month, SUM(o.gloss\_amt\_usd) AS gloss\_total\_usd

FROM accounts ac JOIN orders o ON ac.id=o.account\_id

WHERE ac.name = 'Walmart'

GROUP BY account\_name, year, month

ORDER BY gloss\_total\_usd DESC

limit 1

A picture containing text, screenshot, font, line

Description automatically generated