1. Try pulling all the data from the accounts table, and all the data from the orders table.

```
SELECT *
FROM accounts a
JOIN orders o ON a.id = o.account_id
```

2. Try pulling standard_qty, gloss_qty, and poster_qty from the orders table, and the website and the primary_poc from the accounts table.

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

3. 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 like 'Walmart'
```

---> for some reason results doesn't show up in postgres

4. 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 s.name rep,
       r.name rgn,
```

```
       a.name acct
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 3
```

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. 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,
        ROUND((o.total_amt_usd /(o.total + 0.01)),2)
unit_price
FROM region r
JOIN sales_reps s ON r.id = s.region_id
JOIN accounts a ON s.id = a.sales_rep_id
JOIN orders o ON a.id = o.account_id
```

6. 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_name,
        s.name sales_rep,
        a.name account_name
FROM sales_reps s
JOIN region r ON s.region_id = r.id
AND r.name = 'Midwest'                          -- I'm pre
filtering here!
JOIN accounts a ON s.id = a.sales_rep_id
ORDER BY 3
```

7. 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. 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_name,
       s.name sales_rep,
       a.name account_name
FROM accounts a
JOIN sales_reps s ON a.sales_rep_id = s.id
AND s.name like 'S%'
JOIN region r ON s.region_id = r.id
AND r.name = 'Midwest'
ORDER BY 3
```

8. 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_name,
       s.name sales_rep,
       a.name account_name
FROM accounts a
JOIN sales_reps s ON a.sales_rep_id = s.id
AND s.name like '% K%'
        --  to easily get the last name!
JOIN region r ON s.region_id = r.id
AND r.name = 'Midwest'

-- right(name,-(position(' ' in name)))
-- just to get the last name
```

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

```
SELECT R.NAME REGION_NAME, A.NAME ACCOUNT_NAME,
```

```
        ROUND((O.STANDARD_AMT_USD / (O.TOTAL + 0.01)),2)
UNIT_PRICE
FROM ACCOUNTS A
JOIN ORDERS O
ON A.ID = O.ACCOUNT_ID
AND O.STANDARD_QTY > 100
JOIN SALES_REPS S
ON A.SALES_REP_ID = S.ID
JOIN REGION R
ON S.REGION_ID = R.ID
```

10. 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. Your final table should have 3 columns: region name, account name, and unit price. Sort for the smallest unit price first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total_amt_usd/(total+0.01).

```
SELECT r.name region_name,
       a.name account_name,
       ROUND((o.standard_amt_usd/(o.total + 0.01)),2)
unit_price
FROM accounts a
JOIN orders o ON a.id = o.account_id
AND o.standard_qty > 100
JOIN sales_reps s ON a.sales_rep_id = s.id
JOIN region r ON s.region_id = r.id
```

11. 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. Your final table should have 3 columns: region name, account name, and unit price. Sort for the largest unit price first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (total_amt_usd/(total+0.01).

```
SELECT  r.name region_name,
```

```
        a.name account_name,
        ROUND((total_amt_usd/(total + 0.01)),2) unit_price
FROM orders o
JOIN accounts a ON o.account_id = a.id
AND (standard_qty > 100 AND poster_qty > 50)
JOIN sales_reps s ON a.sales_rep_id = s.id
JOIN region r ON r.id = s.region_id
ORDER BY 3 DESC
```

12. What are the different channels used by account id 1001? Your final table should have only 2 columns: account name and the different channels. You can try SELECT DISTINCT to narrow down the results to only the unique values.

```
SELECT  DISTINCT a.name account_name,
        w.channel channel
FROM accounts a
JOIN web_events w ON w.account_id = a.id
AND a.id = 1001
```

13. 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 account_name,
        o.total order_total,
        o.total_amt_usd order_total_amt_usd
FROM orders o
JOIN accounts a ON a.id = o.account_id
AND occurred_at BETWEEN '2015-01-01' AND '2016-01-01'
-- Note that the endpoint is 2016
ORDER BY 1
```