SELECT AND JOINS BEGINNING

```
--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 primary_poc, channel, occurred_at, name
FROM web_events
JOIN accounts
ON web_events.account_id = accounts.id
WHERE name IN('Walmart')



--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;

--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
FROM sales_reps s
JOIN region r
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

--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.
SELECT r.name Region, a.name Account, o.total_amt_usd
/(total+0.01) Unit_price
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
JOIN region r
ON s.region_id = r.id
WHERE o.standard_qty > 100 AND o.poster_qty < 50



--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 Sales_Representative, 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 IN( 'Midwest')

--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, s.name Sales_Representative, 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 IN( 'Midwest') AND s.name LIKE 'S%'
ORDER BY a.name

--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 Sales_Representative, 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 IN( 'Midwest') AND s.name LIKE '% %K%'
ORDER BY a.name

--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, a.name Account, o.total_amt_usd
/(total+0.01)  Unit_Price
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
JOIN region r
ON s.region_id = r.id
WHERE o.standard_qty > 100

--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, w.channel Channels
FROM accounts a
JOIN web_events w
ON w.account_id = a.id
```

```
WHERE a.id IN(1001)

--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 orders o
JOIN accounts a
ON o.account_id = a.id
WHERE o.occurred_at BETWEEN '2015-01-01' AND '2016-01-01'
ORDER BY o.occurred_at desc
```

SELECT AND JOINS FINISH

SQL AGGREGATIONS BEGINNING

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

SELECT sum(poster_qty) AS poster_total
FROM orders

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

SELECT sum(total_amt_usd) AS sales_total
FROM orders

--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 orders.id, standard_amt_usd + gloss_amt_usd AS
total_standard_glossorders
FROM orders;

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

SELECT min(orders.occurred_at)
FROM orders
```

```
--When was the earliest order ever placed? You only need to return
the date.(without aggregate function)

SELECT orders.occurred_at
FROM orders
ORDER BY orders.occurred_at ASC

--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) avg_standard_qty,
AVG(gloss_qty)avg_gloss_qty, AVG(poster_qty)avg_poster_qty,
AVG(standard_amt_usd) avg_standard_amt_usd,
AVG(gloss_amt_usd)avg_gloss_amt_usd, AVG(poster_amt_usd)
avg_poster_amt_usd
FROM orders

--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 o.account_id = a.id
ORDER BY o.occurred_at

--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, o.total_amt_usd
ORDER BY o.total_amt_usd desc

--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 w.occurred_at desc
 LIMIT 1
--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(w.channel) Times_used
FROM web_events w
GROUP BY w.channel
ORDER BY COUNT(w.channel) desc

--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 w.occurred_at asc
LIMIT 1


--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(o.total_amt_usd)
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY a.name
ORDER BY MIN(o.total_amt_usd)

--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) Sales_rep
FROM region r
JOIN sales_reps s
ON s.region_id = r.id
GROUP BY r.name
ORDER BY COUNT(s.name)
```

```sql
--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, 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;

--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
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name;

--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 Sales_rep, w.channel Channel, COUNT( w.channel)
Times_used
FROM accounts a
JOIN web_events w
ON w.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
GROUP BY s.name, w.channel
ORDER BY  COUNT( w.channel) desc
```

```
--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 Region, w.channel Channel, COUNT( w.channel)
Times_used
FROM accounts a
JOIN web_events w
ON w.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
JOIN region r
ON s.region_id = r.id
GROUP BY r.name, w.channel
ORDER BY  COUNT( w.channel) desc

--Have any sales reps worked on more than one account?

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

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

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

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

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

```
ORDER BY num_accounts desc
```

Date Functions

DATE_TRUNC

allows you to truncate your date to a particular part of your date-time column. Common trunctions are day, month, and year. Here is a great blog post by Mode Analytics on the power of this function.

DATE_PART

can be useful for pulling a specific portion of a date, but notice pulling month or day of the week (dow) means that you are no longer keeping the years in order. Rather you are grouping for certain components regardless of which year they belonged in.

For additional functions you can use with dates, check out the documentation here, but the DATE_TRUNC and DATE_PART functions definitely give you a great start!

You can reference the columns in your select statement in GROUP BY and ORDER BY clauses with numbers that follow the order they appear in the select statement. For example

```
--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;

--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) ord_year,
SUM(total_amt_usd) total_spent
 FROM orders
 GROUP BY 1
 ORDER BY 2 DESC;
```

```
--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('year', occurred_at) ord_year, COUNT(*)
total_sales
 FROM orders
 GROUP BY 1
 ORDER BY 2 DESC;

--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 o.account_id = a.id
 WHERE a.name = 'Walmart'
 GROUP BY 1
 ORDER BY 2 DESC;

--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 o.account_id, o.total_amt_usd,
CASE WHEN o.total_amt_usd > 3000 THEN 'Large'
ELSE 'Small' END AS order_level
FROM orders o

--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;
```

```
--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 SUM(total_amt_usd) as Total_spending, a.name,
   CASE WHEN SUM(total_amt_usd) > 200000 THEN 'Top Customers'
   WHEN SUM(total_amt_usd) >= 100000 AND SUM(total_amt_usd) <
200000        THEN 'Good Customers'
   ELSE 'Minor Customers' END AS client_category
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY 2
ORDER BY 1 desc

--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 SUM(total_amt_usd) as Total_spending, a.name,
   CASE WHEN SUM(total_amt_usd) > 200000 THEN 'Top Customers'
   WHEN SUM(total_amt_usd) >= 100000 AND SUM(total_amt_usd) <
200000        THEN 'Good Customers'
   ELSE 'Minor Customers' END AS client_category,
   DATE_PART ('year', occurred_at)
FROM orders o
JOIN accounts a
ON o.account_id = a.id
WHERE occurred_at > '2015-12-31'
GROUP BY 2, o.occurred_at
ORDER BY 1 desc

--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(*) AS total_orders,
CASE WHEN COUNT(*) > 200 THEN 'TOP SALES_REP'
ELSE 'NOT TOP SALES_REP' END AS Sales_Rep_Category
```

```
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
GROUP BY 1
ORDER BY 2 desc


--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!

SELECT s.name, COUNT(*) AS total_orders, SUM(total_amt_usd) AS
Total_Sales,
CASE WHEN COUNT(*) > 200 OR SUM(total_amt_usd) > 750000 THEN 'TOP
SALES_REP'
WHEN COUNT(*) > 100 OR SUM(total_amt_usd) > 500000 THEN 'MEDIUM
SALES_REP'
ELSE 'LOW SALES_REP' END AS Sales_Rep_Category
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON a.sales_rep_id = s.id
GROUP BY 1
ORDER BY 3 desc
```

SQL AGGREGATIONS FINISHING

SQL SUBQUERIES BEGINNING

```
--We want to find the average number of events for each day for
each channel. The first table will provide us the number of events
for each day and channel, and then we will need to average these
values together using a second query.

SELECT channel,
avg(event_count) as avg_event_count
FROM
(SELECT date_trunc('day', occurred_at)as Day,
COUNT(*) as event_count , w.channel
FROM web_events w
GROUP BY 1,3
ORDER BY 1 ) sub
GROUP BY 1
ORDER BY 2 desc

All the orders that took place in the same month and year and then
pull the average qty of each type of paper.

SELECT AVG(standard_qty) avg_std, AVG(gloss_qty) avg_gls,
AVG(poster_qty) avg_pst
FROM orders
WHERE DATE_TRUNC('month', occurred_at) =
     (SELECT DATE_TRUNC('month', MIN(occurred_at)) FROM orders);

--Provide the name of the sales_rep in each region with the
largest amount of total_amt_usd sales.

SELECT
s.name Sales_rep, SUM(o.total_amt_usd) Total_amt, r.name Region
FROM orders o
JOIN accounts a
ON o.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, 3
ORDER BY 3
```

```
--For the region with the largest sales total_amt_usd), COUNT(*)
Total_orders, how many total orders were placed?

SELECT r.name Region, SUM(o.total_amt_usd), COUNT(*) 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 1

-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.name, SUM(total_amt_usd) tot_spent
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY 1
ORDER BY 2 desc
LIMIT 10) sub
```

SQL SUBQUERIES FINISHING

SQL DATA CLEANING BEGINNING

```
--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
```