**/\*END TO END SQL PROJECT "parch\_and\_posey" DATA. Below are the table and qustions that covers END TO END SQL CONCEPTS\*/**

Create database parch\_and\_posey

use parch\_and\_posey

Create table accounts (

Id int not null primary key,

`name` varchar(30) not null,

website varchar(30) unique,

lat decimal(30,10),

`long` decimal(30,10),

primary\_poc varchar(30) unique,

sales\_rep\_id int);

create table orders(

Id int not null,

account\_id int not null,

occured timestamp not null,

standard\_quantity int,

gloss\_quantity int,

poster\_quantity int,

total int,

standard\_amount\_usd int,

gloss\_amount\_usd int,

poster\_amount\_usd int,

total\_amount\_usd int

);

create table region(

id int not null primary key,

`name` varchar(30) not null);

create table sales\_rep(

id int,

`name` varchar(30) not null,

region\_id Int not null,

foreign key (region\_id) references region(ID) );

create table web\_events(

id int,

account\_id int not null,

occured\_at datetime,

`channel` varchar(30));

alter table sales\_rep

add constraint salesrep\_pkey primary key (id)

alter table accounts

add constraint accountsf\_key foreign key (sales\_rep\_id) references sales\_rep (id)

set session sql\_mode = ''

load data infile

'E:/web\_events.csv'

INTO TABLE web\_events

fields terminated by ','

enclosed by '"'

lines terminated by '\n'

ignore 1 rows

select \* from orders;

select \* from accounts;

select \* from region;

select \* from sales\_rep;

select \* from web\_events;

/////////////////////////////////// ------- **Basic SQL.sql -----** ///////////////////////////////////

/\*Try writing your own query to select only the id, account\_id, and occurred\_at columns for all orders in the orders table.\*/

select id,account\_id,occured from orders

/\*Try using LIMIT yourself below by writing a query that displays all the data in the occurred, account\_id, and channel columns of the web\_events table, and limits the output to only the first 15 rows.\*/

select \* from web\_events limit 15orders

/\*10 earliest orders in the orders table. Include the id, occurred at, and total\_amt\_usd.\*/

select id,occured,total\_amount\_usd from orders order by occured desc

/\*Use the web\_events table to find all information regarding individuals who were contacted via the channel of organic or adwords.\*/

select web\_events.`channel`,accounts.id,accounts.`name` from accounts left outer join web\_events

on accounts.id = web\_events.account\_id where channel in ('organic','adwords')

/\*top 5 orders in terms of largest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd\*/

select id,account\_id,total\_amount\_usd from orders order by total\_amount\_usd desc limit 5

/\*the lowest 20 orders in terms of smallest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd\*/

select id,account\_id,total\_amount\_usd from orders order by total\_amount\_usd limit 30

/\*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\_amount\_usd from orders

order by account\_id, total\_amount\_usd desc

/\*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\_amount\_usd

FROM orders

ORDER BY total\_amount\_usd DESC, account\_id;

/\*Compare the results of these two queries above. How are the results different when you switch the column you sort on first\*/

/\*Ans:In query #1, all of the orders for each account ID are grouped together, and then within each of those groupings, the orders appear from the greatest order amount to the least. In query #2, since you sorted by the total dollar amount first, the orders appear from greatest to least regardless of which account ID they were from. Then they are sorted by account ID next. (The secondary sorting by account ID is difficult to see here, since only if there were two orders with equal total dollar amounts would there need to be any sorting by account ID.)\*/

/\*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\_amount\_usd > 100

/\*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'

/\*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, round(standard\_amount\_usd / standard\_quantity,2) as unit\_price from orders limit 10

/\*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\_amount\_usd / (standard\_amount\_usd + gloss\_amount\_usd + poster\_amount\_usd) as poster\_amt\_percentage from orders limit 10

/\*All the companies whose names start with 'C'.\*/

select \* from accounts

where `name` like 'C%'

/\*All companies whose names contain the string 'one' somewhere in the name.\*/

select \* from accounts

where `name` like '% one %'

---below is for with space

select \* from accounts

where `name` like '%one%'

/\*All companies whose names end with 's'.\*/

select \* from accounts

where `name` like '%s'

/\*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')

/\*Use the web\_events table to find all information regarding individuals who were contacted via the channel of organic or adwords.\*/

select accounts.`name` , web\_events.`channel` from accounts left outer join web\_events on accounts.id = web\_events.account\_id

where `channel` in ( 'organic','adwords')

/\*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')

/\*Use the web\_events table to find all information regarding individuals who were contacted via any method except using organic or adwords methods.\*/

select accounts.`name` , web\_events.`channel` from accounts left outer join web\_events on accounts.id = web\_events.account\_id

where `channel` not in ( 'organic','adwords')

/\*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\_quantity > 1000 and

poster\_quantity = 0 and

gloss\_quantity = 0

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

select \* from accounts

where `name` not like 'c%' and `name` like '%s'

/\*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. Then look at your output to see if the BETWEEN operator included the begin and end values or not.\*/

select \* from orders where gloss\_quantity between 22 and 47

/\*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

select \* from web\_events where channel in ('organic','adwords')

and

occured\_at between '2016-01-01 17:13:58' and '2016-12-31 17:13:58' order by occured\_at desc

/\*Find list of orders ids where either gloss\_qty or poster\_qty is greater than 4000. Only include the id field in the resulting table.\*/

select id from orders where gloss\_quantity > 4000 or poster\_quantity > 4000

/\*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\_quantity = 0

and

(gloss\_quantity > 1000 or poster\_quantity > 1000)

---/\*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'

**//////////////////////////------ JOINS --------//////////////////////////**

**LEFT OUTER JOIN, RIGHT OUTER JOIN, INNER JOIN, FULL OUTER JOIN**

/\*Try pulling all the data from the accounts table, and all the data from the orders table.\*/

select accounts.\* ,orders.\* from accounts join orders on accounts.id = orders.account\_id

/\*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 orders.standard\_quantity,orders.poster\_quantity,accounts.website,accounts.primary\_poc from accounts join orders

on accounts.id = orders.account\_id

/\*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 accounts.`name`,accounts.primary\_poc,web\_events.occured\_at, web\_events.`channel` from accounts join web\_events on accounts.id = web\_events.account\_id where `name` = '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 sales\_rep.`name` as sales\_rep\_name,accounts.`name` as accounts\_name , region.`name` as region\_name from accounts

join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

join region on sales\_rep.region\_id = region.id order by accounts.`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 orders.id,(orders.total\_amount\_usd / total + 0.01) as per\_unit, accounts.`name`, region.`name` as region\_name from accounts

join orders on accounts.id = orders.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id

join region on region.id = sales\_rep.region\_id

order by 1;

/\*--------------------------------------------\*/

/\*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 sales\_rep.`name` as sales\_rep\_name,accounts.`name` as accounts\_name , region.`name` as region\_name from accounts

join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

join region on sales\_rep.region\_id = region.id where region.`name` = 'midwest' order by accounts.`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 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 sales\_rep.`name` as sales\_rep\_name,accounts.`name` as accounts\_name , region.`name` as region\_name from accounts

join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

join region on sales\_rep.region\_id = region.id

having sales\_rep.`name` like 's%' and lower(region.`name`) like 'midwest'

order by accounts.`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 sales\_rep.`name` as sales\_rep\_name,accounts.`name` as accounts\_name , region.`name` as region\_name from accounts

join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

join region on sales\_rep.region\_id = region.id

having lower(sales\_rep.`name`) like '% k%' and lower(region.`name`) like 'midwest'

order by accounts.`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 accounts.`name`, region.`name`,(orders.total\_amount\_usd/total + 0.1) as unit\_price from accounts join orders on

accounts.id = orders.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id

join region on region.id = sales\_rep.region\_id

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

\*/

select accounts.`name`, region.`name`,(orders.total\_amount\_usd/total + 0.1) as unit\_price from accounts join orders on

accounts.id = orders.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id

join region on region.id = sales\_rep.region\_id

where standard\_quantity > 100 and poster\_quantity > 50

order by 3

/\*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 accounts.`name`, region.`name`,(orders.total\_amount\_usd/total + 0.1) as unit\_price from accounts join orders on

accounts.id = orders.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id

join region on region.id = sales\_rep.region\_id

where standard\_quantity > 100 and poster\_quantity > 50

order by 3 desc

/\*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 accounts.id,web\_events.`channel` from accounts join web\_events on accounts.id = web\_events.account\_id

where accounts.id = '1001'

Or

select accounts.id,web\_events.`channel` from accounts join web\_events on accounts.id = web\_events.account\_id

where accounts.id = '1001' group by web\_events.channel

/\*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 a.`name`,o.total,o.total\_amount\_usd ,o.occured from accounts a join orders o on a.id = o.account\_id

where o.occured between '2015-01-01 17:13:58' and '2016-01-01 17:13:58'

order by 4 desc

**//////////////////////////////-----AGGREGATION FUNCTIONS----///////////////////////////////**

**AVG - COUNT - MAX - MIN - SUM - STD - STDDEV**

**MEAN, MEDIAN, MODE calculation along WITH percentile\_cont**

**USE OF CTE – COMMON TABLE EXPRESSION**

/\*Find the total amount of poster\_qty paper ordered in the orders table.\*/

select sum(gloss\_quantity) as total\_amount\_of\_posters\_qty from orders

/\*Find the total amount of standard\_qty paper ordered in the orders table.\*/

select sum(standard\_quantity) as total\_amount\_of\_posters\_qty from orders

/\*Find the total dollar amount of sales using the total\_amt\_usd in the orders table.\*/

select sum(total\_amount\_usd) as total\_amount\_of\_posters\_qty 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 id, standard\_amount\_usd + gloss\_amount\_usd as Total\_amount from orders

/\*Find the standard\_amt\_usd per unit of standard\_qty paper. Your solution should use both an aggregation

and a mathematical operator.\*/

select id, standard\_amount\_usd / standard\_quantity as amount\_per\_unit from orders

/\*When was the earliest order ever placed? You only need to return the date.\*/

SELECT MIN(occured) FROM orders

/\*Try performing the same query as above 1 without using an aggregation function.\*/

select occured from orders order by occured limit 1

/\*When did the most recent (latest) web\_event occur?\*/

select max(occured\_at) from web\_events

/\*Try to perform the result of the previous query without using an aggregation function.\*/

select id, occured\_at from web\_events order by occured\_at desc limit 1

/\*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\_quantity) as avg\_std\_qty,avg(gloss\_quantity) as avg\_gloss\_qty,avg(poster\_quantity) as avg\_poster\_qty, avg(standard\_amount\_usd)

as standard\_amount\_usd, avg(gloss\_amount\_usd) as gloss\_amount\_usd,avg(poster\_amount\_usd) as poster\_amount\_usd from orders

///\* Find median for Total\_amount\_usd/

Below is correct but not working in mysql\*/

with CTE as(

select total\_amount\_usd, row\_number() over (order by total\_amount\_usd) as RN\_asc,

row\_number() over (order by total\_amount\_usd desc) as RN\_desc from orders)

select \* from cte where abs (RN\_asc - RN\_desc) <= 1

select \* , percentile\_cont(0.5) within group (order by total\_amount\_usd) over() as median from orders

/\*below is the table created and claculated median in 2 methods in snowflake\*/

create database median\_calculation

use median\_calculation

create or replace table numbers(

Number integer)

Insert into numbers values(1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11)

select percentile\_cont(0.5) within group (order by number) over() as `median` from numbers

with CTE as(

select number, row\_number() over (order by number) as RN\_asc,

row\_number() over (order by number desc) as RN\_desc from numbers)

select \* from cte where abs (RN\_asc - RN\_desc) <= 1

///////////////////

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

select accounts.name,orders.occured from accounts join orders on accounts.id = orders.account\_id order by occured limit 1

/\*Find the total sales in usd for each account. You should include two columns - the total sales for each company's orders in usd \*/

select accounts.name,sum(orders.total\_amount\_usd) as total\_usd from accounts join orders on accounts.id = orders.account\_id group by accounts.name

/\*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 accounts.name,web\_events.channel,web\_events.occured\_at from accounts join web\_events on accounts.id = web\_events.account\_id

order by web\_events.occured\_at desc

/\*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 channel, count(channel) as total\_number\_of\_times from web\_events group by channel order by total\_number\_of\_times desc

/\*Who was the Sales Rep associated with the earliest web\_event?\*/

select accounts.sales\_rep\_id,web\_events.occured\_at, sales\_rep.name from accounts join web\_events on accounts.id = web\_events.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id order by web\_events.occured\_at limit 1

**//////////--- USE OF AGGREGATIONS ALONG WITH ADVANCE JOIN CONCEPT -----////////////**

/\*Who was the primary contact associated with the earliest web\_event?\*/

select accounts.sales\_rep\_id,web\_events.occured\_at, sales\_rep.name, accounts.primary\_poc as primary\_contact from accounts join web\_events on accounts.id = web\_events.account\_id

join sales\_rep on sales\_rep.id = accounts.sales\_rep\_id order by web\_events.occured\_at 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 accounts.name,min(orders.total\_amount\_usd) as min\_amount from accounts join orders on accounts.id = orders.account\_id group by accounts.name order by min\_amount

/\*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 region.name,count(sales\_rep.name) as total\_sales\_rep from sales\_rep left join region on sales\_rep.region\_id = region.id

group by region.name order by total\_sales\_rep

/\*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 accounts.name, avg(gloss\_quantity) as avg\_gloss\_qty, avg(poster\_quantity) as poster\_quantity, avg(standard\_quantity) as avg\_standard\_quantity

from orders join accounts on accounts.id = orders.account\_id group by accounts.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 accounts.name, avg(standard\_amount\_usd) as avg\_std\_amt, avg(gloss\_amount\_usd) as avg\_gloss\_amt,avg(poster\_amount\_usd) as avg\_std\_amt

from orders join accounts on accounts.id=orders.account\_id group by accounts.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 sales\_rep.name,web\_events.channel as channel , count(web\_events.channel) as count\_of\_channel\_per\_re from web\_events

left join accounts on accounts.id = web\_events.account\_id join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

group by sales\_rep.name order by 3 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 region.name,web\_events.channel,count(web\_events.channel) from web\_events left join accounts on web\_events.account\_id = accounts.id

join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id join region on region.id = sales\_rep.region\_id group by region.name,channel

order by 3 desc

/\*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.

\*/

with cte as (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\_rep s on

a.sales\_rep\_id = s.id join region r on s.region\_id = r.id)

select count(\*) from cte

and

select count(distinct (id)), name from accounts

/\*Have any sales reps worked on more than one account?\*/

select sales\_rep.name,sales\_rep.id, count(\*) from sales\_rep join accounts on accounts.sales\_rep\_id = sales\_rep.id

group by 1,2

order by 3

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

select sales\_rep.name,count(accounts.name) as count\_of\_account\_holding from accounts join sales\_rep on accounts.sales\_rep\_id = sales\_rep.id

group by 1

having count(accounts.name) > 5

/\*How many accounts have more than 20 orders?\*/

select accounts.name as account\_name, count(\*) as count\_of\_orders from accounts join orders on accounts.id = orders.account\_id

group by 1

having count\_of\_orders > 20

/\*Which account has the most orders?\*/

select accounts.name as account\_name, count(\*) as count\_of\_orders from accounts join orders on accounts.id = orders.account\_id

group by 1

order by count\_of\_orders desc

limit 1

/\*Which accounts spent more than 30,000 usd total across all orders?\*/

select accounts.name as account\_name,sum(total\_amount\_usd) as total\_usd from accounts join orders on

accounts.id = orders.account\_id group by 1

having total\_usd > 30000

/\*Which accounts spent less than 1,000 usd total across all orders?\*/

select accounts.name as account\_name,sum(total\_amount\_usd) as total\_usd from accounts join orders on

accounts.id = orders.account\_id group by 1

having total\_usd < 1000

/\*Which account has spent the most with us?\*/

select accounts.name as account\_name,sum(total\_amount\_usd) as total\_usd from accounts join orders on

accounts.id = orders.account\_id group by 1

order by total\_usd desc

limit 1

/\*Which account has spent the least with us?\*/

select accounts.name as account\_name,sum(total\_amount\_usd) as total\_usd from accounts join orders on

accounts.id = orders.account\_id group by 1

order by total\_usd

limit 1

/\*Which accounts used facebook as a channel to contact customers more than 6 times?\*/

select accounts.name as account\_name, count(channel) as count\_of\_facebook from accounts join web\_events on

accounts.id = web\_events.account\_id

where channel = 'facebook'

group by 1

having count\_of\_facebook > 6

order by count\_of\_facebook desc

/\*Which channel was most frequently used by most accounts?\*/

select channel, count(\*) as 'usage' from web\_events group by channel order by 'usage' desc limit 1

/\*Which channel was most frequently used by most accounts? (including account name)\*/

select accounts.name, channel, count(\*) as 'usage' from web\_events join accounts on accounts.id = web\_events.account\_id

group by 1,2 order by 3 desc

---if i dont give 1,2 here and if i give 1, it will group count of usage by channel name, so we may get

count of all channel. in order to get count of 1 particular channel (direct,indirect etc, we should group by

1,2)

**////////////---- USE OF DATE FUNCTION ALONG WITH AGGREGATE AND JOINS ----////////////////**

**EXTRACT - MONTH , YEAR , DATE and Other date functions**

/\*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?\*/

/\*Answer: 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.\*/

select extract(year FROM occured) as 'year', sum(total\_amount\_usd) as total\_sales from orders group by 1 order by 1 desc

/\*Which month did Parch & Posey have the greatest sales in terms of total dollars?

Are all months evenly represented by the dataset?\*/

/\*Answer: In order for this to be 'fair', we should remove the sales from 2013 and 2017.

For the same reasons as discussed above.

The greatest sales amounts occur in December (12).

\*/

select extract(year FROM occured) as 'year', sum(total\_amount\_usd) as total\_sales from orders

where occured between '2014-1-01' and '2017-01-01'

group by 1 order by 1 desc

/\*Which year did Parch & Posey have the greatest sales in terms of total number of orders?

Are all years evenly represented by the dataset?\*/

/\*Answer: 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.\*/

select extract(year FROM occured) as 'year', count(\*) as total\_orders from orders group by 1 order by 2 desc limit 1

/\*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 extract(month FROM occured) as 'year', count(\*) as total\_orders from orders group by 1 order by 2 desc limit 1

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

with cte as (SELECT accounts.name as account\_name, EXTRACT(YEAR FROM occured) AS year, EXTRACT(MONTH FROM occured) AS month, sum(gloss\_amount\_usd)

as total\_gloss\_amount FROM orders join accounts on accounts.id = orders.account\_id

group by 2,3

order by 2,3 desc)

select account\_name,year,month,total\_gloss\_amount from cte where account\_name = 'walmart' order by 4 desc limit 1

/\*Write a query to display for each order, the account ID, total amount of the order, and the level of the order - â€˜Largeâ€™ or â€™Smallâ€™

- depending on if the order is $3000 or more, or smaller than $3000.\*/

select account\_id,total\_amount\_usd,

case when total\_amount\_usd >3000 then 'large' else 'small'

end as remark

from orders

/\*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'

when total < 1000 then 'less\_than\_1000'

end as order\_remark,

count(\*) as total\_number\_of\_orders from orders group by 1

**///////////////----- USE OF CASE STATEMENT --------////////////////////// - DATE FUNCTIONS**

/\*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.\*/

select accounts.name, sum(total\_amount\_usd) as total,

case

when sum(total\_amount\_usd) >= 200000 then 'Top'

when sum(total\_amount\_usd) < 200000 and sum(total\_amount\_usd) >= 100000 then 'middle'

else 'low'

end as remarks from orders join accounts on accounts.id = orders.account\_id group by 1 order by 2 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 accounts.name, sum(total\_amount\_usd) as total,

case

when sum(total\_amount\_usd) >= 200000 then 'Top'

when sum(total\_amount\_usd) < 200000 and sum(total\_amount\_usd) >= 100000 then 'middle'

else 'low'

end as remarks from orders join accounts on accounts.id = orders.account\_id

where extract(year from occured) between 2016 and 2017

group by 1 order by 2 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 sales\_rep.name, count(orders.id) as total\_count\_of\_orders,

case

when count(orders.id) > 200 then 'top'

else 'low'

end as remark

from sales\_rep join accounts on sales\_rep.id = accounts.sales\_rep\_id

join orders on accounts.id = orders.account\_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 sales\_rep.name, count(orders.id) as total\_count\_of\_orders,sum(total\_amount\_usd) as total\_usd,

case

when count(orders.id) > 200 then 'top'

else 'low'

end as remark,

case

when sum(total\_amount\_usd) > 750000 or count(orders.id) > 200 then 'top'

when sum(total\_amount\_usd) > 50000 or count(orders.id) > 150 then 'middle'

else 'low'

end as remark\_2\_on\_value

from sales\_rep join accounts on sales\_rep.id = accounts.sales\_rep\_id

join orders on accounts.id = orders.account\_id group by 1 order by 2 desc

**////////////////////// ----- Sub Queries & Temporary Tables ----- ////////////////////**

/\*number of events that occur for each day for each channel\*/

select DAYNAME(occured\_at) as 'daynumber',channel, count(\*) as count\_of\_events from web\_events group by 1,2 order by 1 desc;

/\*find the average number of events for each channel. Average per day\*/

select channel, avg(count\_of\_events) as AVG\_COUNT from

(select DAYNAME(occured\_at) as 'daynumber',channel, count(\*) as count\_of\_events from web\_events group by 1,2 order by 1 desc) as test

group by 1

order by 2 desc

/\*list of orders happended at the first month in P&P history , ordered by occurred\_at \*/

select \* from orders where extract(month from occured) = (select extract(month from min(occured)) from orders)

Having extract(year from occured) = 2013;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**////////////////------- ADVANCED JOINS CONCEPT -------------------//////////////////////**

ALONG WITH UPPER, LOWER, RIGHT, LEFT , LOCATE, CONCATE, TRIM, LENGTH and Others

select \* from orders;

select \* from accounts;

select \* from region;

select \* from sales\_rep;

select \* from web\_events;

/\*sales rep total sales for each region\*/

select s.`name` as sales\_rep\_name, r.`name` as region\_name, sum(o.total\_amount\_usd) as total\_sales from orders o

left outer join accounts a on o.account\_id = a.id

left outer join sales\_rep s on s.id = a.sales\_rep\_id

left outer join region r on r.id = s.region\_id

group by 1,2

///////////////////////////////////////

/\*maximum total sales in each region\*/

select s.`name` as sales\_rep\_name, r.`name` as region\_name, max(o.total\_amount\_usd) as max\_total\_sales from orders o

left outer join accounts a on o.account\_id = a.id

left outer join sales\_rep s on s.id = a.sales\_rep\_id

left outer join region r on r.id = s.region\_id

group by 1,2

////////////////////////////////

/\*1) Provide the name of the sales\_rep in each region with the largest amount of total\_amt\_usd sales.\*/

select region\_name, sales\_rep\_name, total\_max\_sales from

(select r.id as region\_id, r.`name` as region\_name, sr.id as sales\_rep\_id, sr.`name` as sales\_rep\_name, sum(o.total\_amount\_usd)

as total\_sales\_per\_rep from

orders o left join accounts a on o.account\_id = a.id

left join sales\_rep sr on a.sales\_rep\_id = sr.id

left join region r on sr.region\_id = r.id

group by 1,2,3,4) as temp1

join (select temp2.region\_id as region\_id, max(total\_sales\_per\_rep) as total\_max\_sales from

(select r.id as region\_id, r.`name` as region\_name, sr.id as sales\_rep\_id, sr.`name` as sales\_rep\_name, sum(o.total\_amount\_usd)

as total\_sales\_per\_rep from

orders o left join accounts a on o.account\_id = a.id

left join sales\_rep sr on a.sales\_rep\_id = sr.id

left join region r on sr.region\_id = r.id

group by 1,2,3,4) as temp2

group by 1) as temp3

on temp3.region\_id = temp1.region\_id

and temp1.total\_sales\_per\_rep = temp3.total\_max\_sales

//////////////////

/\*For the region with the largest (sum) of sales total\_amt\_usd, how many total (count) orders were placed?\*/

select count(\*) as total\_orders, r.`name` from orders o left join accounts a on a.id = o.account\_id

left join sales\_rep sr on a.sales\_rep\_id = sr.id

left join region r on r.id = sr.region\_id where r.name = (

select region\_name from (

select r.`name` as region\_name,sum(o.total\_amount\_usd) total\_sales from

orders o left join accounts a on o.account\_id = a.id

left join sales\_rep sr on a.sales\_rep\_id = sr.id

left join region r on sr.region\_id = r.id

group by 1 order by 2 desc limit 1) as temp)

//////////////////////////////////////////////////////////////////

**/\*\*\*\* CTE Common Table Expressions \*\*\*\*/**

/\*find the average number of events for each channel per day.\*/

with CTE as (select date(occured\_at) as `day`, `channel`, count(\*) as total\_event

from web\_events

group by 1,2)

select `channel`, avg(total\_event) as avg\_total\_event from CTE

........................................................

**///////////////\* CLEANING AND EXTRACTING DATA/\*//////////////////////////////////////**

/\*Check how many distinct domains were there in the websites under accounts data/\*/

select right(website,3), count(\*) from accounts

group by 1

order by 2 desc

..........................................

/\*2) 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 upper(left(name,1)) as letter, count(\*) as total from accounts group by 1 order by 2 desc;

..................................................................

/\*3) 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) as starts\_with\_number, sum(let) as starts\_with\_letters from

(select case when upper(left(`name`,1)) in ('0','2','3','4','5','6','7','8','9') then 1 else 0 end as num,

case when upper(left(`name`,1)) in ('0','2','3','4','5','6','7','8','9') then 0 else 1 end as let

from accounts) as temp1

.......................................................

/\*4) 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(num) as starts\_with\_ovels, sum(numv) as starts\_with\_no\_ovels from

(select case when upper(left(`name`,1)) in ('a','e','i','o','u') then 1 else 0 end as num,

case when upper(left(`name`,1)) in ('a','e','i','o','u') then 0 else 1 end as numv

from accounts) as temp1

.................................................................

/\*1)Use the accounts table to create first and last name columns that hold the first and last names for the primary\_poc.\*/

select primary\_poc, left(primary\_poc, locate(' ',primary\_poc) -1) as first\_name ,

right(primary\_poc, locate(' ',primary\_poc) +1) as last\_name

from accounts;

**/\*\*\*\*\*\* CONCATE or || \*\*\*\*\*\*\*\*\*\*\*\*\*/**

/\*1/2)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 CTE as (select primary\_poc, trim(left(primary\_poc, locate(' ',primary\_poc) -1)) as first\_name ,

trim(right(primary\_poc, locate(' ',primary\_poc) +1)) as last\_name, `name`

from accounts)

select concat(first\_name,last\_name,"@",`name`) as mail\_id from CTE

/\*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 CTE as (select trim(left(primary\_poc, locate(' ',primary\_poc) -1)) as first\_name ,

trim(right(primary\_poc, locate(' ',primary\_poc) +1)) as last\_name, primary\_poc,`name`

from accounts)

select primary\_poc, concat(right(primary\_poc,1),left(primary\_poc,1),right(last\_name,1),left(last\_name,1),

length(first\_name),length(last\_name),upper(`name`)) as `password` from CTE

**//////////////---------- ADVANCED WINDOW FUNCTIONS -----------//////////////////////////////**

CUME\_DIST() - DENSE\_RANK() - FIRST\_VALUE() - LAG() - LAST\_VALUE() - LEAD() - NTH\_VALUE()

NTILE() - PERCENT\_RANK() - RANK() - ROW\_NUMBER()

/\*create a running total of standard\_amt\_usd (in the orders table) over order time with no date truncation.

Your final table should have two columns:

one with the amount being added for each new row, and a second with the running total.\*/

select \* from orders;

select standard\_amount\_usd, sum(standard\_amount\_usd) over (order by occured) as running\_total from orders;

/\*create a running total of total\_amount\_usd (in the orders table) over order time,

but this time, date truncate occurred\_at by year and partition by that same year-truncated occurred\_at variable.

Your final table should have three columns: One with the amount being added for each row,

one for the truncated date, and a final column with the running total within each year.\*/

select year(occured) as year, total\_amount\_usd as total\_sales, sum(total\_amount\_usd) over (partition by year(occured) order by

occured) as running\_total from orders

/\*Ranking Total Paper Ordered by Account\*/

select id, account\_id,total, rank() over(partition by account\_id order by total desc) as `rank` from orders;

**/\*-------------- Window Aliases -----------------\*/**

Create a dense rank, min.max,avg,total using Aliases

select id,

account\_id,

standard\_quantity,

dense\_rank() over main\_window as ranking,

count(standard\_quantity) over main\_window as total\_new\_orders,

sum(standard\_quantity) over main\_window as total\_new\_orders,

avg(standard\_quantity) over main\_window as total\_new\_orders,

min(standard\_quantity) over main\_window as total\_new\_orders,

max(standard\_quantity) over main\_window as total\_new\_orders

from orders

window main\_window as (partition by account\_id order by year(occured))

.......................

**/\* ----------------- LEAD and LAG ------------------------\*/**

**/\*LAG \*/**

select account\_id,sum(standard\_quantity) as standard\_sum,

lag(sum(standard\_quantity)) over (order by account\_id) as `lag`,

sum(standard\_quantity)-lag(sum(standard\_quantity)) over (order by account\_id) as difference

from orders

group by 1

................................................

/\*you want to determine how the current order's total revenue

("total" meaning from sales of all types of paper)

compares to the next order's total revenue.

there should be four columns: occurred\_at, total\_amt\_usd, lead, and lead\_difference.\*/

select occured,total\_amount\_usd,

lead(total\_amount\_usd) over(order by occured) as `lead`,

(total\_amount\_usd - lead(total\_amount\_usd) over(order by occured)) as difference\_revenue

from orders

...................................................

SELECT id, account\_id, occured,standard\_quantity,

NTILE(4) OVER (ORDER BY standard\_quantity) AS quartile,

NTILE(5) OVER (ORDER BY standard\_quantity) AS quintile,

NTILE(100) OVER (ORDER BY standard\_quantity) AS percentile

FROM orders

ORDER BY standard\_quantity DESC;

/\*Use the NTILE functionality to divide the accounts into 4 levels in terms of the amount of standard\_qty

for their orders. Your resulting table should have the account\_id, the occurred\_at time for each order,

the total amount of standard\_qty paper purchased, and one of four levels in a standard\_quartile column.\*/

select account\_id, occured,standard\_quantity,

ntile(4) over (partition by account\_id order by standard\_quantity) as tile4,

ntile(5) over (Partition by account\_id order by standard\_quantity) as tile5,

ntile(100) over(partition by account\_id order by standard\_quantity) as percentile

from orders

/\*Use the NTILE functionality to divide the accounts into two levels in terms of the amount of gloss\_qty

for their orders. Your resulting table should have the account\_id,

the occurred\_at time for each order, the total amount of gloss\_qty paper purchased,

and one of two levels in a gloss\_half column.

\*/

select account\_id, occured, gloss\_quantity,

ntile(2) over(partition by account\_id order by gloss\_quantity) as ntile2 from orders;

/\*For percentile division use NITILE(100)?\*;

**///////--- ONCE AGAIN ADVANCED JOINS CONCEPT WITH OTHER CONCEPT COMBINED -----//////**

/\*Say you're an analyst at Parch & Posey and you want to see:

each account who has a sales rep and each sales rep that has an account

(all of the columns in these returned rows will be full)

but also each account that does not have a sales rep and each sales rep that does not have an account

(some of the columns in these returned rows will be empty)\*/

SELECT accounts.id, accounts.name, sales\_rep.id, sales\_rep.`name`

FROM accounts

LEFT OUTER JOIN sales\_rep ON accounts.sales\_rep\_id = sales\_rep.id

UNION

SELECT accounts.id, accounts.name, sales\_rep.id, sales\_rep.`name`

FROM accounts

RIGHT OUTER JOIN sales\_rep ON accounts.sales\_rep\_id = sales\_rep.id

WHERE accounts.id IS NULL;

/\*above we did (WHERE accounts.id IS NULL) because we are doing full outer join, which is just opposite of inner join\*/

SELECT accounts.name,accounts.primary\_poc,sales\_rep.name

FROM accounts

LEFT JOIN sales\_rep

ON accounts.sales\_rep\_id = sales\_rep.id

AND accounts.primary\_poc < sales\_rep.name

/\*List down all web events one after another datewise ascending order , account id wise\*/

select

we1.id as web\_id,

we1.account\_id as account\_id,

we1.occured\_at as occured\_at,

we1.`channel` as `channel`,

we2.id as web\_id,

we2.account\_id as account\_id,

we2.occured\_at as occured\_at,

we2.`channel` as `channel`

from web\_events we1 left outer join web\_events we2 on

we1.account\_id = we2.account\_id

and

we1.occured\_at > we2.occured\_at

and

we1.occured\_at <= we2.occured\_at + interval 1 day

ORDER BY we1.account\_id, we2.occured\_at;

**///////////------ /\* UNION ALL vs UNION \*/ --------////////////////////////////////////////**

/\*Nice! UNION only appends distinct values.

More specifically, when you use UNION, the dataset is appended, and any rows in the appended table

that are exactly identical to rows in the first table are dropped.

If youâ€™d like to append all the values from the second table, use UNION ALL.

Youâ€™ll likely use UNION ALL far more often than UNION.\*/

SELECT \* FROM accounts a1

WHERE name LIKE 'Walmart'

UNION ALL

SELECT \* FROM accounts a2

WHERE name LIKE 'Disney'

/\*Perform the union in your first query (under the Appending Data via UNION header) in a common table expression and

name it double\_accounts. Then do a COUNT the number of times a name appears in the double\_accounts table.

If you do this correctly, your query results should have a count of 2 for each name.\*/

WITH double\_accounts AS(

SELECT \* FROM accounts a1

UNION ALL

SELECT \* FROM accounts a2)

SELECT name, COUNT(\*)

FROM double\_accounts

GROUP BY name;

/\* PERFORMANCE TUNING \*/

EXPLAIN

SELECT \*

FROM web\_events

WHERE occured\_at >= '2016-01-01'

AND occured\_at < '2016-02-01'

LIMIT 100;

//////////////////////////////////////////////////////////////////////////////////////////////////////////