Diagram

Description automatically generated

**Questions: CASE**

1. 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'

ELSE 'SMALL' END AS LEVEL\_OF\_ORDER

FROM orders;

1. 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 Categories,

COUNT(\*) Num\_orders

FROM orders

GROUP BY 1;

1. 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

a.name ACCOUNT,

SUM(o.total\_amt\_usd),

CASE WHEN SUM(o.total\_amt\_usd)>200000 THEN 'greater than 200,000'

WHEN SUM(o.total\_amt\_usd)>=100000 and SUM(o.total\_amt\_usd)<=200000 THEN '200,000 and 100,000'

ELSE 'under 100,000' END AS level\_of\_CLV

FROM orders o

JOIN accounts a

ON o.account\_id=a.id

GROUP BY 1

ORDER BY 2 DESC;

1. 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,

SUM(o.total\_amt\_usd),

CASE WHEN SUM(o.total\_amt\_usd)>200000 THEN 'greater than 200,000'

WHEN SUM(o.total\_amt\_usd)>=100000 and SUM(o.total\_amt\_usd)<=200000 THEN '200,000 and 100,000'

ELSE 'under 100,000' END AS level\_of\_CLV

FROM orders o

JOIN accounts a

ON o.account\_id=a.id

WHERE DATE\_PART('YEAR', o.occurred\_at) in ('2016','2017')

GROUP BY 1

ORDER BY 2 DESC;

1. 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 sales\_rep,

COUNT(\*) total\_orders,

CASE WHEN COUNT(\*) > 200 THEN 'top'

ELSE 'not' END sales\_rep\_performance

FROM orders o

JOIN accounts a

ON o.account\_id = a.id

JOIN sales\_reps s

ON s.id=a.sales\_rep\_id

GROUP BY 1

ORDER BY 2 DESC;

1. 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 sales\_rep,

COUNT(\*) total\_orders,

SUM(o.total\_amt\_usd) total\_sales,

CASE WHEN COUNT(\*) > 200

OR SUM(o.total\_amt\_usd) >750000

THEN 'top'

WHEN COUNT(\*) > 150 and COUNT(\*)<=200

OR SUM(o.total\_amt\_usd) >500000

THEN 'middle'

ELSE 'low' END sales\_rep\_performance

FROM orders o

JOIN accounts a

ON o.account\_id = a.id

JOIN sales\_reps s

ON s.id=a.sales\_rep\_id

GROUP BY 1

ORDER BY 3 DESC;