**Name : Jinsar Ahmed**

**Roll No: 20SW116**

**Section: 2**

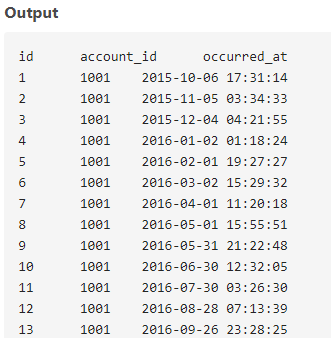
**Lab No: 06**

**SQL for Data Analysis**

**Task#01: Run the following queries:**

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

**Query:** SELECT id, account\_id, occurred\_at FROM orders;

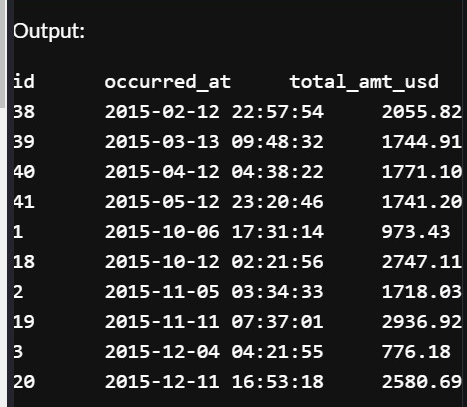


1. **Write a query to return the 10 earliest orders in the orders table. Include the id, occurred\_at, and total\_amt\_usd.**

**Query:** SELECT id, occurred\_at, total\_amt\_usd FROM orders

ORDER BY occurred\_at

LIMIT 10;



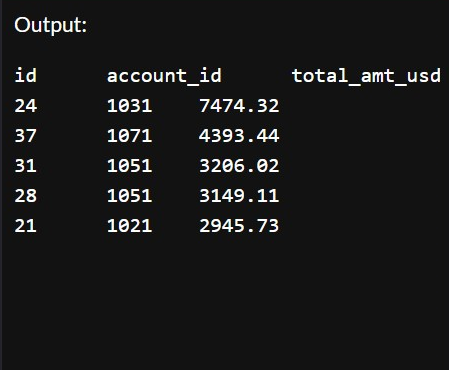
1. **Write a query to return the top 5 orders in terms of largest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd.**

**Query:** SELECT id, account\_id, total\_amt\_usd

FROM orders

ORDER BY total\_amt\_usd DESC

LIMIT 5;



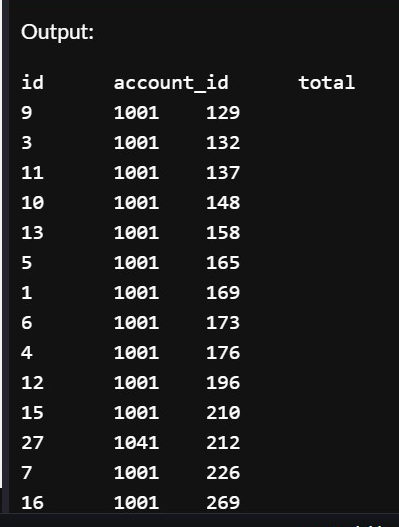
1. **Write a query to return the bottom 20 orders in terms of least total. Include the id, account\_id, and total.**

**Query:** SELECT id, account\_id, total

FROM orders

ORDER BY total

LIMIT 20;



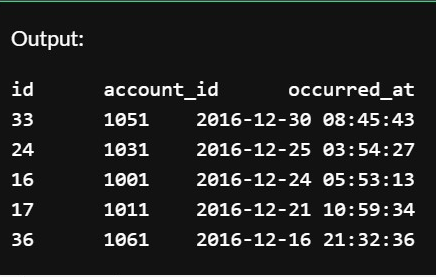
1. **Write a query that returns the top 5 rows from orders ordered according to newest to oldest, but with the largest total\_amt\_usd for each date listed first for each date. You will notice each of these dates shows up as unique because of the time element. When you learn about truncating dates in a later lesson,**

**you will better be able to tackle this question on a day, month, or yearly basis.**

**Query:** SELECT \* FROM orders

ORDER BY occurred\_at DESC, total\_amt\_usd DESC

LIMIT 5;



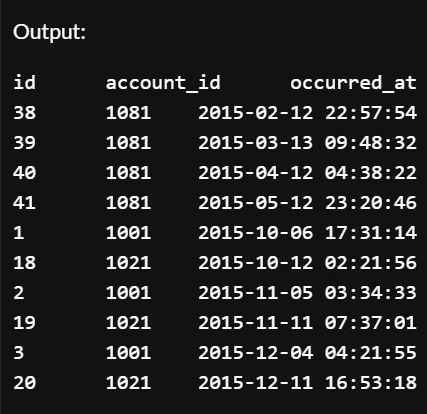
1. **Write a query that returns the top 10 rows from orders ordered according to oldest to newest, but with the smallest total\_amt\_usd for each date listed first for each date. You will notice each of these dates shows up as unique because of the time element.When you learn about truncating dates in a later lesson,**

**you will better be able to tackle this question on a day, month, or yearly basis.**

**Query:** SELECT \* FROM orders

ORDER BY occurred\_at ,total\_amt\_usd

LIMIT 10;

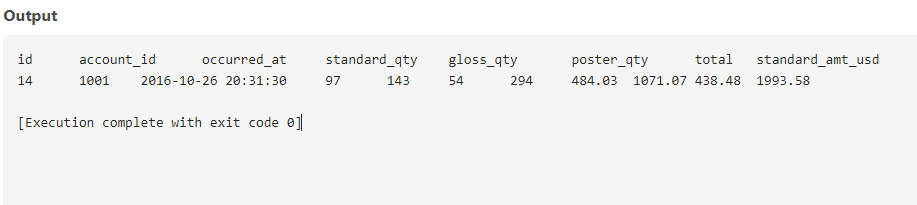


1. **Write a query that Pull the first 5 rows and all columns from the orders table that have a dollar amount of gloss\_amt\_usd greater than or equal to 1000.**

**Query:** SELECT \* FROM orders

WHERE gloss\_amt\_usd >= 1000

LIMIT 5;

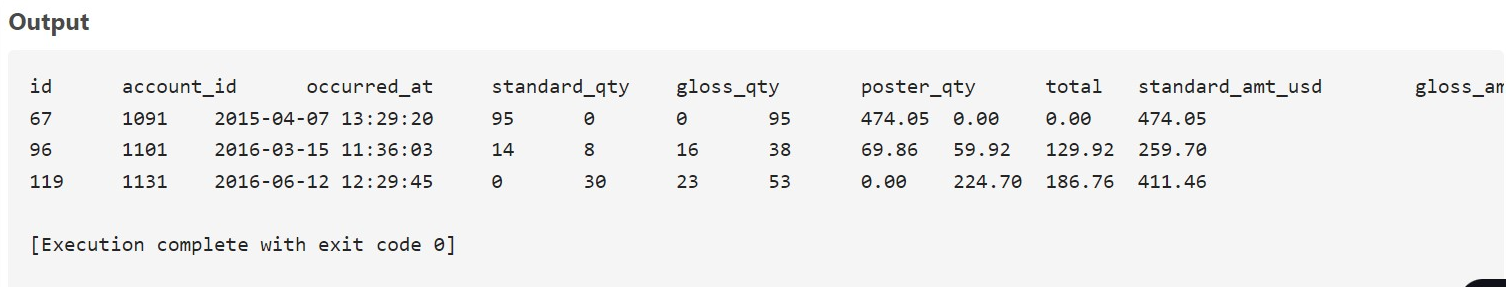


1. **Write a query that Pull the first 10 rows and all columns from the orders table that have a total\_amt\_usd less than 500.**

**Query:** SELECT \* FROM orders

WHERE total\_amt\_usd < 500

LIMIT 10;



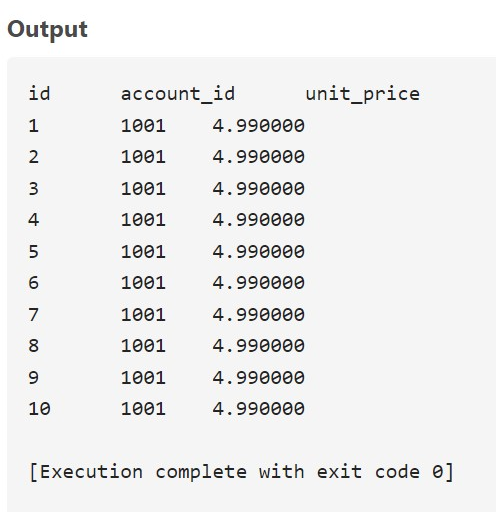
1. **Using the orders table, Create a column that divides the standard\_amt\_usd by the standard\_qty to find the unit price for standard paper for each order.**

**Limit the results to the first 10 orders, and include the id and account\_id fields.**

**Query:** SELECT id, account\_id, standard\_amt\_usd/standard\_qty as unit\_price

FROM orders

LIMIT 10;

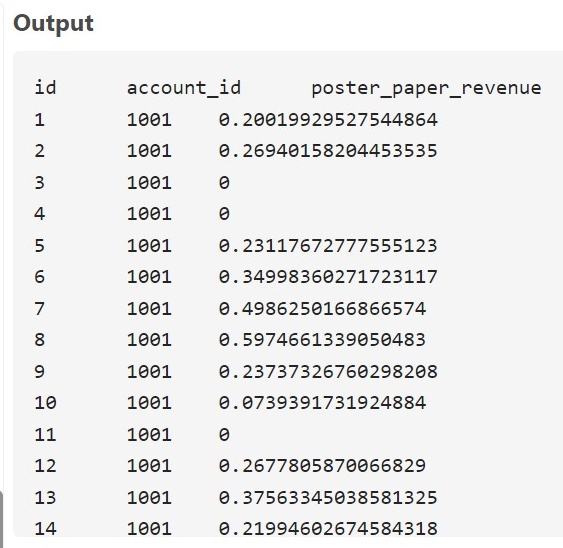


**10. Using the orders table , Write a query that finds the percentage of revenue that comes from poster paper for each order. You will need to use only the columns that end with \_usd. (Try to do this without using the total column). Include the id and account\_id fields. NOTE - you will be thrown an error with the correct solution to this question. This is for a division by zero. You will learn how to get a solution without an error to this query when you learn about CASE statements in a later section.**

**For now, you might just add some very small value to your denominator as a work around.**

**Query:** SELECT id, account\_id, poster\_amt\_usd/(poster\_amt\_usd + gloss\_amt\_usd + standard\_amt\_usd + 1e-10) as poster\_paper\_revenue

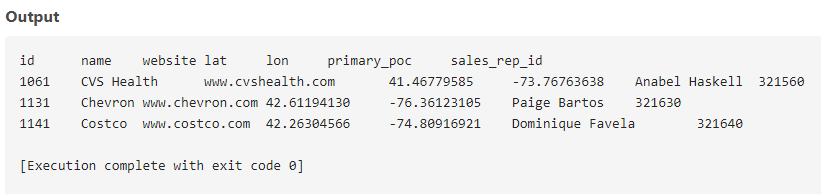
FROM orders;



**11. Use the accounts table to find, All the companies whose names start with 'C'.**

**Query:** SELECT \*

FROM accounts

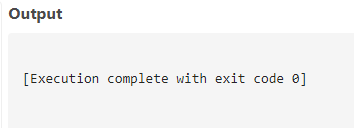
 WHERE name LIKE 'C%';

**12. Use the accounts table to find, All companies whose names contain the string 'one' somewhere in the name.**

**Query:** SELECT \*

FROM accounts

WHERE name LIKE '%one%';

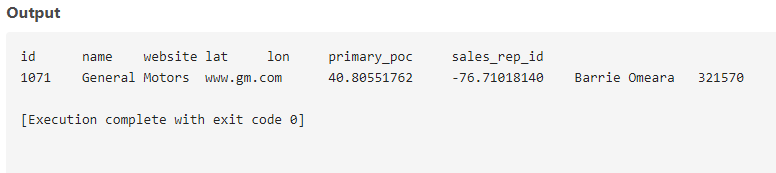


**13. Use the accounts table to find, All companies whose names end with 's'.**

**Query:** SELECT \*

FROM accounts

WHERE name LIKE '%s';



**14. Use the accounts table to find the account name, primary\_poc, and sales\_rep\_id for Walmart, Target, and Nordstrom.**

**Query:** SELECT name, primary\_poc, sales\_rep\_id

FROM accounts

WHERE name in ('Walmart', 'Target','Nordstrom');

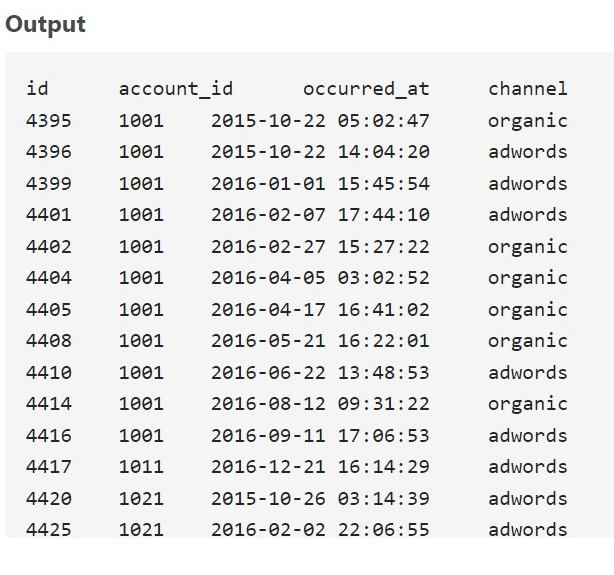


**15. Use the web\_events table to find all information regarding individuals who were contacted via the channel of organic or adwords.**

**Query:** SELECT \*

FROM web\_events

WHERE channel in ('organic','adwords');



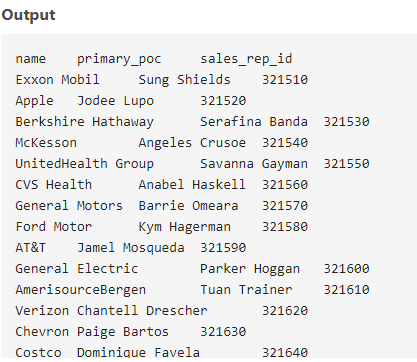
**16. We can pull all of the rows that were excluded from the queries in the previous two concepts with our new operator.**

1. *Use the accounts table to find the account name, primary poc, and sales rep id for all stores except Walmart, Target, and Nordstrom.*

**Query:** SELECT name,primary\_poc,sales\_rep\_id

FROM accounts

where name NOT IN ('Walmart', 'Target','Nordstrom');

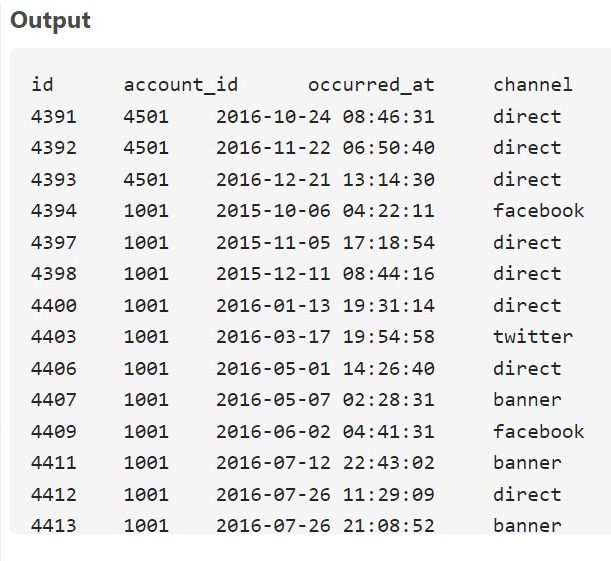


1. *Use the web\_events table to find all information regarding individuals who were contacted via any method except using organic or adwords methods*.

**Query:** SELECT \*

FROM web\_events

where channel NOT IN ('organic', 'adwords');



**17. Use the accounts table to find:**

1. *All the companies whose names do not start with 'C'.*

**Query:** SELECT \*

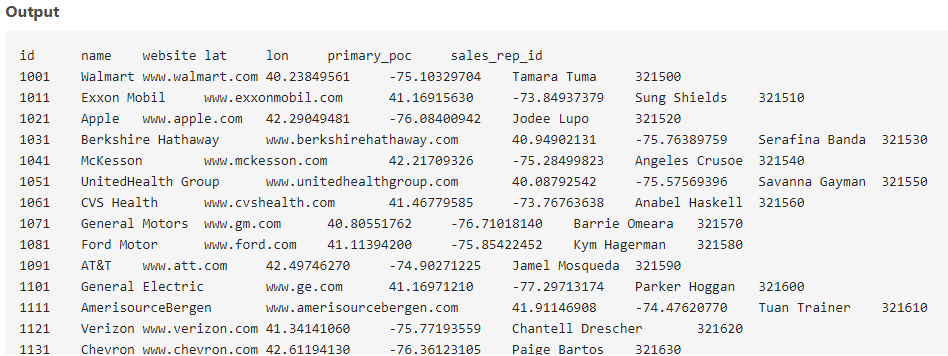
FROM accounts

 WHERE name NOT LIKE 'C%';

1. *All companies whose names do not contain the string 'one' somewhere in the name.*

**Query:** SELECT \*

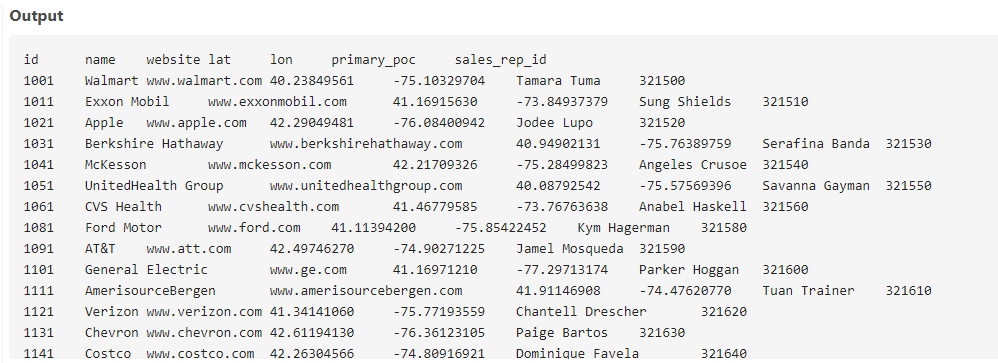
FROM accounts

 WHERE name NOT LIKE '%one%';

1. *All companies whose names do not end with 's'.*

**Query:** SELECT \*

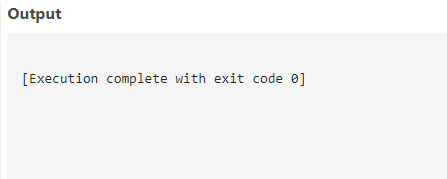
FROM accounts

 WHERE name NOT LIKE '%s';

**18. Write a query that returns all the orders where the standard\_qty is over 1000, the poster\_qty is 0, and the gloss\_qty is 0.**

**Query:** SELECT \* FROM orders

WHERE standard\_qty>1000 AND poster\_qty=0 AND gloss\_qty=0;

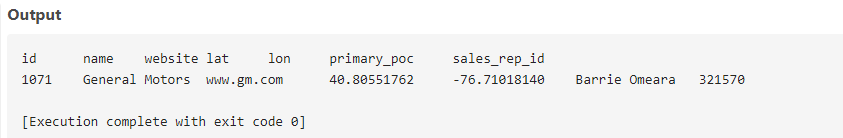


**19. Using the accounts table find all the companies whose names do not start with 'C' and end with 's'.**

**Query:** SELECT \*

FROM accounts

WHERE name NOT LIKE 'C%' AND name LIKE '%s';



**20. Use the web\_events table to find all information regarding individuals who were contacted via organic or adwords and started their account at any point in 2016 sorted from newest to oldest.**

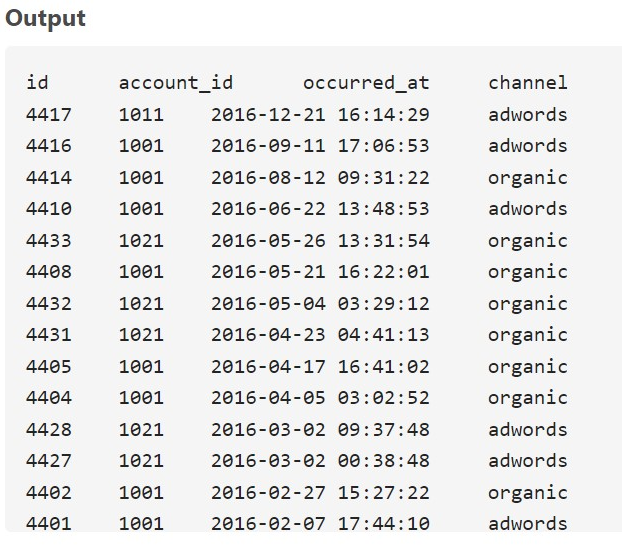
**Query:** SELECT \*

FROM web\_events

WHERE channel IN ('organic', 'adwords')

AND occurred\_at BETWEEN '2016-01-01' AND '2017-01-01'

ORDER BY occurred\_at DESC;

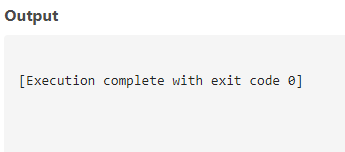


**21. Write a query that returns all the orders where the standard\_qty is over 1000, the poster\_qty is 0, and the gloss\_qty is 0.**

**Query:** SELECT \*

FROM orders

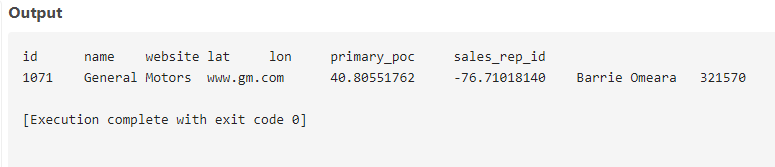
WHERE standard\_qty > 1000 AND poster\_qty = 0 AND gloss\_qty = 0;



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

**Query:** SELECT \* FROM accounts

WHERE name NOT LIKE 'C%' AND name LIKE '%s'



**23. Use the web\_events table to find all information regarding individuals who were contacted via organic or adwords and started their account at any point in 2016 sorted from newest to oldest.**

**Query:** SELECT \* FROM web\_events

where channel IN ('organic', 'adwords')

AND occurred\_at >='2016-01-01' AND occurred\_at <'2017-01-01'

ORDER BY occurred\_at DESC;

