ORDER BY

1. Top 5 orders in terms of largest total\_amt\_usd. Including id, account\_id, and total\_amt\_usd in the result.

USE [parch\_and\_posey]

GO

SELECT TOP 5 id, account\_id, total\_amt\_usd

FROM orders

ORDER BY total\_amt\_usd DESC;

1. Get lowest 20 orders in terms of smallest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd in the result.

SELECT TOP 20 id, account\_id, total\_amt\_usd

FROM orders

ORDER BY total\_amt\_usd;

1. Query that displays the order id, account if, and total dollar amount for all orders sorted first by the account if )in ascending) and then by the total dollar amount (in descending).

SELECT id, account\_id, total\_amt\_usd

FROM orders

ORDER BY account\_id ASC, total\_amt\_usd DESC;

WHERE

1. Query to only show orders from top customer (account id 4251)

SELECT TOP 100 \*

FROM Orders

WHERE account\_id = 4251

ORDER By occurred\_at

Derived Columns

1. Write a query that finds the percentage of revenue that comes from poster paper for each order.

SELECT TOP 10 id, (standard\_amt\_usd/total\_amt\_usd)\*100 AS std\_percent, total\_amt\_usd

FROM orders;

LIKE Operator

1. Filtering out companies whose name start with ‘S’

SELECT name

FROM accounts

WHERE name LIKE 'S%';

IN Operator - to filter data based on several possible value

1. Find the account names, primary point of contacts, and sales representative id for Walmart, Target, and Nordstrom.

SELECT name, primary\_poc, sales\_rep\_id

FROM accounts

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

1. Using the web\_events table to find all information regarding individuals who were contacted via the channels ‘organic’ or ‘adwords’.

SELECT \*

FROM web\_events

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

AND Operator

1. Viewing order details of all the orders that occurred between 2016-04-01 and 2016-10-01

SELECT \*

FROM orders

WHERE occurred\_at >= '2016-04-01' AND occurred\_at <= '2016-10-01'

ORDER BY occurred\_at DESC;

BETWEEN Operator

1. Finding full 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

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

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

ORDER BY occurred\_at DESC;

JOIN Operator

1. Show a table that provides the region for each sales rep along with their associated accounts. Your final table should include 3 columns: region name, sales rep name, and the account name. Sort the accounts alphabetically (A-Z) according to account name.

SELECT s.name AS sales\_rep\_name, a.name AS account\_name, r.name AS region\_name

FROM sales\_reps s

JOIN accounts a

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

JOIN region r

ON s.region\_id = r.id

ORDER BY a.name;

JOINS and Filtering

1. A table that provides the region for each sales rep along with their associated accounts only for the Midwest region. Your final table should include three columns: region name, sales rep name, and the account name.

Sort the accounts alphabetically (A-Z) according to account name.

SELECT r.name RegionName, s.name SalesRepName, a.name AccountName

FROM sales\_reps s

JOIn accounts a

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

JOIN region r

ON s.region\_id = r.id

AND r.name = 'Midwest'

1. 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 Midwest region. Your final table should include 3 columns: region name, sales rep name, and account name. Sort account name (A-Z).

(Method 1 – using AND)

SELECT r.name RegionName, s.name SalesRepName, a.name AccountName

FROM sales\_reps s

JOIn accounts a

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

JOIN region r

ON s.region\_id = r.id

AND (r.name = 'Midwest' AND s.name LIKE 'S%')

ORDER BY a.name

(Method 2 – using WHERE)

SELECT r.name RegionName, s.name SalesRepName, a.name AccountName

FROM sales\_reps s

JOIn accounts a

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

JOIN region r

ON s.region\_id = r.id

WHERE (r.name = 'Midwest' AND s.name LIKE 'S%')

ORDER BY a.name

SQL Aggregations

1. Find the **standard\_amt\_usd** per unit of **standard\_qty** paper. Your solution should use both an aggregation and a mathematical operator.

SELECT SUM(standard\_amt\_usd)/SUM(standard\_qty) AS standard\_price\_per\_unit

FROM orders;

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\_qty) avg\_standard, AVG(gloss\_qty) avg\_gloss, AVG(poster\_qty) avg\_poster,

AVG(standard\_amt\_usd) avg\_standard\_usd, AVG(gloss\_amt\_usd) avg\_gloss\_usd, AVG(poster\_amt\_usd) avg\_poster\_usd

FROM orders;

GROUP BY

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) smallest\_order\_USD

FROM orders o

JOIN accounts a

ON o.account\_id = a.id

GROUP BY a.name

ORDER BY smallest\_order\_USD;

1. 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 AS region, COUNT(s.name) AS sales\_rep

FROM sales\_reps s

JOIN region r

ON s.region\_id = r.id

GROUP BY r.name

ORDER BY sales\_rep;

1. Have any **sales reps** worked on more than one account?

(Query 1)

SELECT s.id, s.name, COUNT(a.name) num\_accounts

FROM accounts a

JOIN sales\_reps s

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

GROUP BY s.id, s.name

ORDER BY num\_accounts;

(Test of Query 1)

SELECT DISTINCT id, name

FROM sales\_reps;

HAVING Function

1. How many accounts spent less than 1,000 usd total across all orders?

SELECT a.id, a.name, SUM(o.total\_amt\_usd) total\_spent

FROM accounts a

JOIN orders o

ON a.id = o.account\_id

GROUP BY a.id, a.name

HAVING SUM(o.total\_amt\_usd) < 1000

ORDER BY total\_spent;