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;
```

2) 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;
```

3) 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

4) 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

5) 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

6) 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

7) 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');
```

8) 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

9) 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

10) 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

11) 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

12) 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'
```

13) 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

14) 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;
```

15) 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

16) 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;
```

17) 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;
```

18) 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

19) 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;
```