Week 4: Data Aggregation in SQL

LittleDabase, Employee Self Join, and Parch and Posey Database

GROUP BY

SQL aggregate functions like COUNT, AVG, and SUM have something in common: they all aggregate across the entire table. But what if you want to aggregate only part of a table? For example, you might want to count the number of orders for each account.

In situations like this, you'd need to use the GROUP BY clause. GROUP BY allows you to separate data into groups, which can be aggregated independently of one another.

Notice that you can group by multiple columns, but you have to separate column names with commas just as with ORDER BY.

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 and the company name.

```
SELECT a.name, SUM(o.total_amt_usd) as total_sales
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
ORDER BY total_sales desc
LIMIT 3;
```

name	total_sales
EOG Resources	382873.3
Mosaic	345618.6
IBM	326819.5

2. 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(*) total
FROM web_events
GROUP BY channel
ORDER BY total desc
```

channel	total
direct	5298
facebook	967
organic	952
adwords	906
banner	476
twitter	474

3. **YOUR TURN** What was the largest order placed by each account in terms of total usd. Provide only two columns - the account name and the total usd. Order the data by dollar amount.

```
SELECT a.name, MAX(total_amt_usd) largest_order
FROM accounts a
JOIN orders o
```

```
ON a.id = o.account_id
GROUP BY a.name
ORDER BY largest_order
LIMIT 3;
```

name	largest_order
Nike	390.25
Delta Air Lines	859.64
Level 3 Communications	881.73

4. **YOUR TURN** 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, COUNT(*) num_reps
FROM region r
JOIN sales_reps s
ON r.id = s.region_id
GROUP BY r.name
ORDER BY num_reps;
```

name	num_reps
Midwest	9
Southeast	10
West	10
Northeast	21

5. **YOUR TURN** 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 spent on each of the paper types.

```
SELECT a.name, AVG(o.standard_qty) avg_stand, AVG(o.gloss_qty) avg_gloss, AVG(o.poster_qty) avg_post
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
LIMIT 3;
```

name	avg_stand	avg_gloss	avg_post
Comcast	294.00000	18.28571	28.85714
Microsoft	88.46154	32.23077	62.30769
Monsanto	353.61404	42.71930	26.24561

6. 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 s.name, w.channel, COUNT(*) num_events
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name, w.channel
```

```
ORDER BY num_events DESC
LIMIT 5;
```

name	channel	num_events
Earlie Schleusner	direct	234
Vernita Plump	direct	232
Moon Torian	direct	194
Georgianna Chisholm	direct	188
Tia Amato	direct	185

HAVING

The WHERE clause doesn't allow you to filter on aggregate columns, that's where the HAVING clause comes in. The HAVING clause comes after GROUP BY and before ORDER BY.

7. How many of the sales reps have more than 5 accounts that they manage?

```
SELECT s.name, COUNT(*) num_accounts
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name
HAVING COUNT(*) > 5
ORDER BY num_accounts;
```

name	num_accounts
Necole Victory	6
Eugena Esser	6
Sibyl Lauria	6
Debroah Wardle	6
Elba Felder	6
Samuel Racine	6
Babette Soukup	7
Michel Averette	7
Nelle Meaux	7
Soraya Fulton	7
Derrick Boggess	7
Gianna Dossey	7
Elna Condello	7
Charles Bidwell	7
Cliff Meints	7
Julia Behrman	8
Delilah Krum	8
Tia Amato	8
Elwood Shutt	9
Dawna Agnew	9
Marquetta Laycock	9
Saran Ram	10
Arica Stoltzfus	10
Brandie Riva	10
Moon Torian	10
Hilma Busick	10
Micha Woodford	11
Calvin Ollison	11
Earlie Schleusner	11
Maryanna Fiorentino	11
Vernita Plump	11
Dorotha Seawell	11
Maren Musto	11
Georgianna Chisholm	15

Technically, we can get this using a SUBQUERY as shown below:

```
SELECT COUNT(*) num_reps_above5
FROM(SELECT s.name, COUNT(*) num_accounts
     FROM accounts a
     JOIN sales_reps s
     ON s.id = a.sales_rep_id
     GROUP BY s.name
     HAVING COUNT(*) > 5
     ORDER BY num_accounts) AS Table1;
```

num_reps_above5
34

8. ** YOUR TURN** How many accounts have more than 20 orders?

```sql
SELECT a.name, COUNT(*) num_orders
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
HAVING COUNT(*) > 20
ORDER BY num_orders
LIMIT 5;
```

name	num_orders
Thrivent Financial for Lutherans	21
Performance Food Group	21
Anthem	21
Raytheon	22
Jabil Circuit	22

9. ** YOUR TURN** Which account has the most orders?

```sql
SELECT a.name, COUNT(*) num_orders
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
ORDER BY num_orders DESC
LIMIT 1;
```

name	num_orders
Leucadia National	71

10. **YOUR TURN** How many accounts spent more than $30,000 total with Parch and Posey throughout the years?

```sql
select count(*)
from (
    select a.name, sum(o.total_amt_usd) total_spent
    from orders o
    join accounts a
    on o.account_id = a.id
    group by a.name
    having sum(o.total_amt_usd) > 30000
    order by total_spent
    ) table1
```

count
204

11. **YOUR TURN** Which account has spent the most with us?

```sql
SELECT a.name, SUM(o.total_amt_usd) total_spent
FROM accounts a
JOIN orders o
ON a.id = o.account_id
GROUP BY a.name
```

```
ORDER BY total_spent DESC
LIMIT 1;
```

name	total_spent
EOG Resources	382873.3

12. **YOUR TURN** Which account used facebook most as a channel?

```
SELECT a.name, w.channel, COUNT(*) use_of_channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
WHERE w.channel = 'facebook'
GROUP BY a.name, w.channel
ORDER BY use_of_channel DESC
LIMIT 1;
```

name	channel	use_of_channel
Gilead Sciences	facebook	16

13. **YOUR TURN** Which accounts used facebook as a channel to contact customers more than 6 times?

```
SELECT a.name, w.channel, COUNT(*) use_of_channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
WHERE w.channel = 'facebook'
GROUP BY a.name, w.channel
HAVING COUNT(*) > 6
ORDER BY use_of_channel
LIMIT 5;
```

name	channel	use_of_channel
eBay	facebook	7
Avis Budget Group	facebook	7
Laboratory Corp. of America	facebook	7
Wells Fargo	facebook	7
Parker-Hannifin	facebook	7

14. **YOUR TURN** What is the total number of times each channel was used by different accounts?

```
SELECT a.name, w.channel, COUNT(*) use_of_channel
FROM accounts a
JOIN web_events w
ON a.id = w.account_id
GROUP BY a.name, w.channel
ORDER BY use_of_channel DESC
LIMIT 10;
```

Figure 1: DATE_TRUNC

name	channel	use_of_channel
Leucadia National	direct	52
New York Life Insurance	direct	51
Colgate-Palmolive	direct	51
Philip Morris International	direct	49
FirstEnergy	direct	48
BlackRock	direct	48
ADP	direct	48
AutoNation	direct	48
Charter Communications	direct	48
Altria Group	direct	47

DATE Function with GROUP BY

GROUPing BY a date column is not usually very useful in SQL, as these columns tend to have transaction data down to a second. Keeping date information at such a granular data is both a blessing and a curse, as it gives really precise information (a blessing), but it makes grouping information together directly difficult (a curse).

Lucky for us, there are a number of built in SQL functions that are aimed at helping us improve our experience in working with dates:

- **DATE_TRUNC** allows you to truncate your date to a particular part of your date-time column. Common trunctions are day, month, and year.
- **DATE_PART** can be useful for pulling a specific portion of a date, but notice pulling month or day of the week (dow) means that you are no longer keeping the years in order. Rather you are grouping for certain components regardless of which year they belonged in.

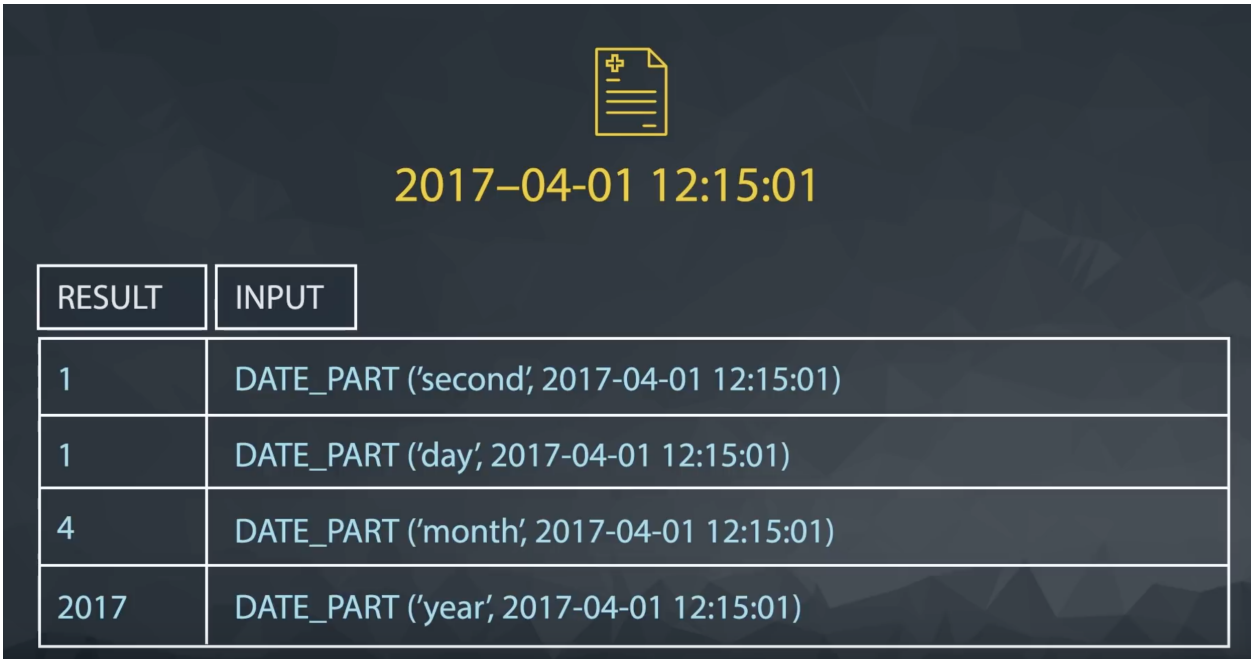

Figure 2: DATE_PART

15. Find the sales ($) for all orders in each year, ordered from largest to smallest. Do you notice any trends in the yearly sales totals?

```
SELECT DATE_PART('year', occurred_at) ord_year,  SUM(total_amt_usd) total_spent
FROM orders
GROUP BY 1 -- 1 refers to the first argument in the SELECT statememnt (ord_year)
ORDER BY 1 DESC
LIMIT 5;
```

ord_year	total_spent
2017	78151.43
2016	12864917.92
2015	5752004.94
2014	4069106.54
2013	377331.00

16. **YOUR TURN** Which month did Parch & Posey have the largest sales ($)?

```
SELECT DATE_PART('month', occurred_at) ord_month, SUM(total_amt_usd) total_spent
FROM orders
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

ord_month	total_spent
12	3129412
10	2427506
11	2390034
9	2017217
7	1978731

17. **YOUR TURN** In which year and month did Walmart spend($) the most on gloss paper?

```
SELECT DATE_TRUNC('month', o.occurred_at) ord_date, SUM(o.gloss_amt_usd) tot_spent
FROM orders o
JOIN accounts a
ON a.id = o.account_id
WHERE a.name = 'Walmart'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

ord_date	tot_spent
2016-05-01	9257.64

CASE

The CASE statement is SQL's way of handling if/then logic. The CASE statement is followed by at least one pair of WHEN and THEN statements—SQL's equivalent of IF/THEN in Excel. Because of this pairing, you might be tempted to call this SQL CASE WHEN, but CASE is the accepted term.

Every CASE statement must end with the END statement. The ELSE statement is optional, and provides a way to capture values not specified in the WHEN/THEN statements.

- The CASE statement always goes in the SELECT clause
- CASE must include the following components: WHEN, THEN, and END. ELSE is an optional component.
- You can make any conditional statement using any conditional operator (like WHERE) between WHEN and THEN. This includes stringing together multiple conditional statements using AND and OR.
- You can include multiple WHEN statements, as well as an ELSE statement to deal with any unaddressed conditions.

18. We would like to understand 3 different branches of customers based on the amount associated with their purchases. The top branch includes anyone with a Lifetime Value (total sales of all orders) greater than 200,000 usd. The second branch is between 200,000 and 100,000 usd. The lowest branch 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, SUM(total_amt_usd) total_spent,
     CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
          WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
          ELSE 'low'
     END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY a.name
```

```
ORDER BY 2 DESC
LIMIT 5;
```

name	total_spent	customer_level
EOG Resources	382873.3	top
Mosaic	345618.6	top
IBM	326819.5	top
General Dynamics	300694.8	top
Republic Services	293861.1	top

19. Restrict the results of the preivous question to the orders occurred only in 2016 and 2017.

```
SELECT a.name, SUM(total_amt_usd) total_spent,
     CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
          WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
          ELSE 'low'
      END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
WHERE occurred_at > '2016-01-01' and occurred_at < '2018-01-01'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

name	total_spent	customer_level
Pacific Life	255319.2	top
Mosaic	172180.0	middle
CHS	163471.8	middle
Core-Mark Holding	148105.9	middle
Disney	129157.4	middle

20. **YOUR TURN** 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, COUNT(*) num_ords,
     CASE WHEN COUNT(*) > 200 THEN 'top'
          ELSE 'not'
     END AS sales_rep_level
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 s.name
ORDER BY 2 DESC
LIMIT 5;
```

name	num_ords	sales_rep_level
Earlie Schleusner	335	top
Vernita Plump	299	top
Tia Amato	267	top
Georgianna Chisholm	256	top
Moon Torian	250	top

21. The previous question 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.

```
SELECT s.name, COUNT(*), SUM(o.total_amt_usd) total_spent,
     CASE WHEN COUNT(*) > 200 OR SUM(o.total_amt_usd) > 750000 THEN 'top'
          WHEN COUNT(*) > 150 OR SUM(o.total_amt_usd) > 500000 THEN 'middle'
          ELSE 'low'
      END AS sales_rep_level
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 s.name
ORDER BY 3 DESC
LIMIT 5;
```

name	count	total_spent	sales_rep_level
Earlie Schleusner	335	1098137.7	top
Tia Amato	267	1010690.6	top
Vernita Plump	299	934212.9	top
Georgianna Chisholm	256	886244.1	top
Arica Stoltzfus	186	810353.3	top