Reporting with SQL Part 3 Notes

Team Treehouse

June 5, 2020

1 Counting Results

- **Syntax 1:** SELECT COUNT(*column name*) FROM *table name*;
  - Counts all non-null values
- **Syntax 2:** SELECT COUNT(*) FROM *table name*;
  - counts all rows in a table
- **Syntax 2:** SELECT COUNT(DISTINCT *column name*) FROM *table*;
  - Counts all items with distinct value in a column

Example:

```
1    SELECT COUNT(DISTINCT category) FROM products;
2
3
4    SELECT COUNT(*) FROM customers ORDER BY id DESC LIMIT 1;
5
```

2 Exercise 1

- Solution included in *exercise_1.sql*

3 Counting Groups of Rows

- **Syntax:** SELECT COUNT(*column name*) FROM *table name* GROUP BY *column name with common value*;
- is almost like using keyword distinct
  - SELECT COUNT(DISTINCT *column name*) FROM *table*;
- but, group by allows to add additional columns

Exxample:

```
1    SELECT category, COUNT(*) AS product_count FROM products GROUP BY
     category;
2
```

```
1 -- SELECT <column> FROM <table> GROUP BY <column>;
2
3 SELECT category, COUNT(*) AS product_count FROM products GROUP BY category;
4
```

Reset Run

category	product_count
Books	20
Clothing	6
Electronics	3

4 Exercise 2

- Solution included in *exercise_2.sql*

5 Getting the Grand Total

- SUM
  - **Syntax:** SELECT SUM(*numeric column*) FROM *table name*;

    Example:

```
    SELECT SUM(cost) AS total_spend, user_id FROM orders GROUP BY
   user_id;

```

```
-- SUM(<column>)

SELECT SUM(cost) AS total_spend, user_id FROM orders GROUP BY user_id;

```

Reset Run

total_spend	user_id
885.5000000000003	1
776.6000000000004	2
1456.7700000000002	5
917.8100000000002	10
237.93000000000006	11
30.97	12
1244.5700000000004	13

- SUM with GROUP BY and WHERE
  - Not possible, but there is an alternative, HAVING
  - **Syntax:** SELECT SUM(*numeric column name*) AS *alias* FROM *table name* GROUP BY *another column name* HAVING *alias operator value*;

    Example:

```
    SELECT SUM(cost) AS total_spend, user_id FROM orders
        GROUP BY user_id
        HAVING total_spend > 250
        ORDER BY total_spend DESC;

```

```
-- SUM(<column>)

SELECT SUM(cost) AS total_spend, user_id FROM orders
                                        GROUP BY user_id
                                        HAVING total_spend > 250
                                        ORDER BY total_spend DESC;
```

Reset Run

total_spend	user_id
1928.6700000000003	20
1456.7700000000002	5
1324.5300000000004	21
1244.5700000000004	13
1027.71	14
917.8100000000002	10
885.5000000000003	1
776.6000000000004	2

6 Exercise 3

- Solution included in *exercise_3.sql*

7 Calculating Averages

- **Syntax:** SELECT AVG(¡numeric column¿) FROM ¡table¿;
- **Syntax (with Group By):** SELECT AVG(¡numeric column¿) FROM ¡table¿ GROUP BY ¡other column¿;

Example:

```
1    SELECT AVG(cost) AS "average", user_id FROM orders GROUP BY
     user_id;
2
```

8 Exercise 4

- Solution included in *exercise_4.sql*

9 Getting Maximum and Minimum Values

- MAX
  - **Syntax 1:** SELECT MAX(*numeric column name*) FROM *table name*;
  - **Syntax 2:** SELECT MAX(*numeric column name*) FROM *table name* GROUP BY *other column name*;
  - Grabs the maximum value in a given column of a table
- MIN
  - SELECT MIN(*numeric column name*) FROM *table name*;
  - SELECT MIN(*numeric column name*) FROM *table name* GROUP BY *other column name*;

Example:

```
1    SELECT AVG(cost) AS average, MAX(cost) AS Maximum, MIN(cost) AS
   Minimum, user_id,
2         FROM orders GROUP BY user_id;
3
```

```
1 -- MAX(<numeric column>) MIN(<numeric column>)
2
3 SELECT AVG(cost) AS average, MAX(cost) AS Maximum, MIN(cost) AS Minimum, user_id
4          FROM orders GROUP BY user_id;
5
```

Reset Run

average	Maximum	Minimum	user_id
17.710000000000008	172.99	2.99	1
19.41500000000001	167.99	2.99	2
63.33782608695653	343.99	5.99	5
48.30578947368422	299.99	3.99	10
33.99000000000001	171.99	7.99	11
10.323333333333332	15.99	2.99	12
28.943488372093032	296.99	3.99	13