SQL
Brandon Knight

Intermediate SQL

Keywords

- **SELECT** - Specifies which columns to retrieve (* for all columns)
- **FROM** - Specifies which table to query
- ; is the statement terminator that marks the end of a query.
- **LIMIT**
- **SUM** - Adds up all numeric values in a column, like the sum formula in Excel

```
SELECT SUM(distance) as total_distance
FROM trips
```

Sums the distance column in the trips table and assigns an alias “total_distance” to it

- **MIN** - Finds the minimum value in a column
- **MAX** - Finds the maximum value in a column
- **AVG** - Calculates the average value in a column

- **COUNT** - Counts how many non-null values exist in a specified column, regardless of data type.
    - COUNT(*) returns the number of all rows regardless of nulls

```
SELECT COUNT(order_id) AS order_count
FROM orders;
```

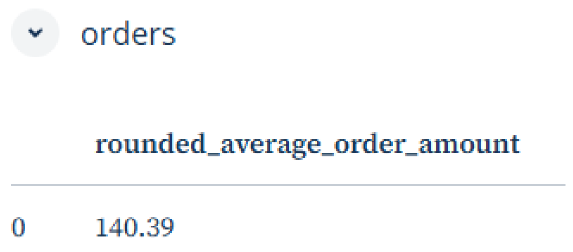
- **COUNT DISTINCT** - Counts how many unique non-null items are in a specified column, regardless of datatype.

```
SELECT COUNT(DISTINCT user_id) AS user_count
FROM orders;
```

Numeric Function

- ROUND → Just like in excel, any aggregate function can be rounded like so

```
    SELECT ROUND(AVG(amount),2) AS  rounded_average_order_amount
    FROM orders;
```



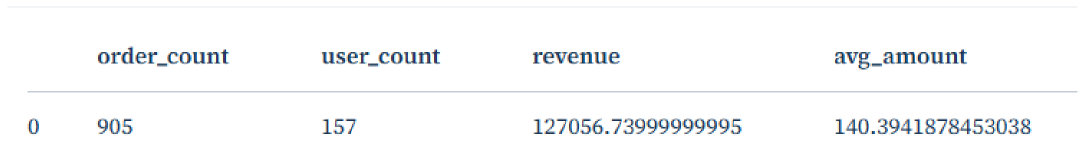
```
orders

     rounded_average_order_amount
0    140.39
```

SQL
Brandon Knight

Intermediate SQL

Running multiple summary values in one query:

```
SELECT COUNT(order_id)          AS order_count,
       COUNT(DISTINCT user_id) AS user_count,
       SUM(amount)              AS revenue,
       AVG(amount)              AS avg_amount
FROM orders;
```



	order_count	user_count	revenue	avg_amount
0	905	157	127056.73999999995	140.3941878453038

Concepts:

- Aggregate functions collapse data into single values, so when you use them in a query, **all columns in the SELECT must be aggregate functions (like SUM(), COUNT(), AVG())**
- SQL keywords:
- SQL functions:
    - String functions
    - Numeric functions

- Primary Key:
- Foreign Key:
- Data Aggregation: Is the operation of summarizing data by applying functions (like SUM, AVG, COUNT) to columns to produce single values.
- Aggregates work across rows (collapses rows) → once aggregated, you can't re-aggregate without grouping or subqueries

- Column aliases defined in the SELECT clause cannot be reused in the same SELECT clause. *SQL evaluates expressions **before** aliases exist.*

```
SELECT COUNT(order_id)                  AS total_orders,
       COUNT(DISTINCT customer_id)      AS unique_customers,
       total_orders / unique_customers AS avg_order_per_customer
FROM orders;
```

So something like this ^ is not possible because SQL does not yet know what the alias total_oders or unique_customers are.

How Databases organize data:

SQL
Brandon Knight

- Databases use a **hierarchical structure**. At the top level, databases contain multiple **schemas** (the blueprints of the database, called **datasets**).
- BigQuery specific hiearchecal structure = **Project → Dataset → Table**
    - Typically referenced as **project.dataset.table**