Aggregate Functions

DSC 301: Lecture 6

February 12, 2021

Lecture Objectives

- Review `DISTINCT` clause
- Aggregate Functions
- `GROUP BY` and `HAVING`
- Introduce Store database (and Products table)

`DISTINCT` clause

Recall the `DISTINCT` clause is used to elminate duplicates in a column of the data. For example, the following statement returns $108,714$ records containing obvious duplicates in the `carrier` field.

```
SELECT carrier FROM Flights;
```

However, if we wish to list the unique airline carriers, we need to use the `DISTINCT` keyword, as in:

```
SELECT DISTINCT(carrier) FROM Flights;
```

Often we will use the `DISTINCT` keyword with the aggregate function `COUNT`.

Aggregate Functions

Aggregate functions, (also called *column functions*), summarize data. Aggregate functions allow us to calculate averages, totals, or find the highest or lowest value in a given column. In addition, aggregate functions provide a means to count the occurrence of a field in the database. Queries containing one or more aggregate functions are often called *summary queries*. The **aggregate**

functions are: `COUNT()`, `MIN()`, `MAX()`, `SUM()`, and `AVG()`. **Note**: by default, all values are included in aggregate calculations (including duplicates). Use `DISTINCT` to remove duplicates in these calculations.

COUNT & COUNT(*)

- `COUNT`: Counts the number of non-null records in a given column
  - `COUNT([ALL | DISTINCT] )`
    - Default is `ALL`
  - Data type is unimportant (counts non-null records) in column
  - Duplicate rows are counted unless `DISTINCT` is applied.
  - `COUNT(*)` counts all records INCLUDING `NULL` values

```
# Basic Structure of COUNT statements
SELECT COUNT(<attribute>) FROM <TableX>;
SELECT COUNT(*) FROM <TableX>;
```

Example 1. *Count the number of destinations.*

```
SELECT COUNT(DISTINCT dest) FROM Flights;
```

Example 2. *Count the number of destinations that American Airlines flies from Seattle.*

```
SELECT COUNT(DISTINCT dest) FROM Flights WHERE carrier = 'AA';
```

Minimum

- `MIN`: Smallest value in column (data type: numeric, string, date)
  - Resulting data type can be numeric, date, or string

    ```
    SELECT MIN(distance) FROM Flights;
    ```

  - For example, if column contains string, min will result in lowest value in a sort sequence.

```
SELECT MIN(carrier) FROM Flights;
```

Example 3. *Find the shortest flight. First note that in this example the first statement produces the result* 0 *(shortest flight ever!). Therefore, add a* `WHERE` *clause to make sure we get flights that actually leave the airport by providing the condition that* `air_time > 0`*. Secondly, note this returns ONLY the time of the shortest flight, NOT the actual flight number, or destination, or carrier, etc. If you want to show those as well, there are a few ways to do so. Perhaps the best is using subqueries, but we have not covered them yet. Another way is to make use of several clauses, some of which are not covered until later in this lecture. We will revisit this example*

```
SELECT MIN(air_time) as AirTime FROM Flights;
SELECT MIN(air_time) as AirTime FROM Flights WHERE air_time > 0;
```

Maximum

- `MAX`: Largest value in column
  - Resulting data type can be numeric, date, or string
  - Note: Cannot be used on right-hand side of comparison
    - That is, the following is invalid syntax

```
SELECT fid FROM Flights where air_time = MAX(air_time);
```

  - Must be used in subquery[1] if used in `WHERE` clause as a condition
  - Same rules apply to `MIN`

Sum

- `SUM`: Total of column (or expression), under given condition

```
# Total milage flown 2014
SELECT SUM(distance) FROM Flights;
```

- Must result in a numeric value

---

[1] Also called **nested query**.

Average

- `AVG()` - the average of all non-null values, default is `ALL`.

```
SELECT AVG(dep_delay) FROM Flights;
```

- May be used with the keyword `DISTINCT`. Note the difference between the next query results and the previous. The results are vastly different. Make sure you select the correct options based on your needs.

```
SELECT AVG(DISTINCT dep_delay) FROM Flights;
```

- Must result in a numeric value

NOTE: A `SELECT` clause that contains an aggregate function can only contain aggregate functions except if there is a *literal* or a `GROUP BY` clause is used. Here is an example with a literal. We will see examples with `GROUP BY` in the next section.

```
SELECT 'Total Records' AS '', count(*) AS '' from Flights;
```

GROUP BY

- `GROUP BY` clause groups the records based on one or more columns (or expressions).
- If an aggregate function is in the `SELECT` clause, then the aggregate is calculated for each group specified by the `GROUP BY` clause. In this case, we may place columns in the `SELECT` clause as mentioned above.

```
SELECT carrier, avg(dep_delay) FROM Flights group by carrier;
```

- `GROUP BY` sorts columns in ascending order (by default). Place `DESC` after column names in `GROUP BY` clause.

Having

`HAVING` is to `GROUP BY` as `WHERE` is to `SELECT`. In particular, the `WHERE` clause places conditions on selected columns, whereas the `HAVING` clause places conditions on groups created by the `GROUP BY` clause.

- `HAVING` clause can only refer to a column included in the `SELECT` clause.
- `HAVING` clause can contain aggregate functions
  - `WHERE` clause cannot contain aggregate functions.

Example 4. *Find the average distance by destinations that start with the letter 'C'. Round up to next whole number of miles.*

```
SELECT
    dest, CEIL(AVG(distance))
FROM
    Flights
GROUP BY dest
HAVING dest like 'C%';
```

Example 5. *Find the longest flight and show its destination. This was mentioned by a student during lecture (and there are a couple problems on the homework where this will come in handy). The "trick" here is to make use of several clauses (perhaps just until we learn subqueries).*

```
SELECT
    dest as City, MAX(distance) AS Distance
FROM
    Flights
GROUP BY City
ORDER BY Distance DESC
LIMIT 1;
```

Store Database

A store database, with only one table (`Products`) was added to the server. The table contains approximately 13000 nature/gardening/pet type products. Eventually we will add more tables to this database including (`Customers, Orders, Manufacturers, ZipCodes, ...`).

Become familiar with this table by running several queries such as:

1. countries manufacture the products?
2. What is the difference between the `price` and `MSRP`? Literally and figuratively.
3. How many products are out of stock?

4. Are there any products that do not have image files?
5. etc.