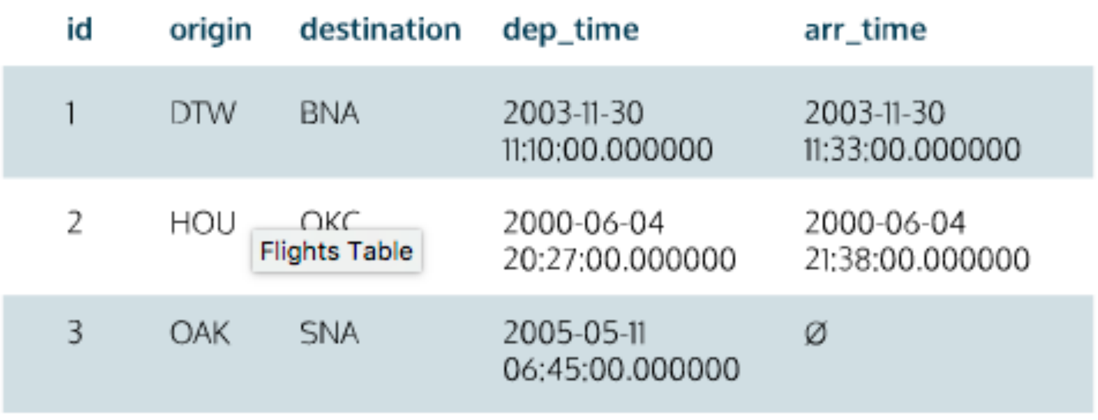
SQL Table Transformation Agregate Functions

Aggregate functions compute a single result from a set of multiple input values. You can think of aggregate data as data collected from multiple rows at a time. In this lesson, we'll continue learning about aggregate functions by focusing on conditionals, sums, and combining aggregate functions.
Conditional Aggregates are aggregate functions that compute a result set based on a given set of conditions.
If you are new to aggregate functions, we recommend you complete this course first.

The count function is an aggregate function, since it aggregates data from multiple rows.
Count the number of rows in the flights table, representing the total number of flights contained in the table

While working with databases, it's common to have empty or unknown "cells" in data tables. For example, in our first flights table, we keep track of when flights arrive in a column called arr_time:



id	origin	destination	dep_time	arr_time
1	DTW	BNA	2003-11-30 11:10:00.000000	2003-11-30 11:33:00.000000
2	HOU	OKC	2000-06-04 20:27:00.000000	2000-06-04 21:38:00.000000
3	OAK	SNA	2005-05-11 06:45:00.000000	Ø

Flights Table

In this table, the third flight was canceled and therefore did not have an arrive time. We'll leave this cell empty, which is denoted as NULL.
What do we do when we need to test whether a value is or is not null? We use the special keywords **IS NULL** or **IS NOT NULL** in the **WHERE** clause (= NULL does not work).

Count the number of rows from the flights table, where arr_time is not null and the destination is ATL.

SELECT **count(*) FROM** flights **WHERE** arr_time **IS NOT NULL** and destination = 'ATL';

Almost every programming language has a way to represent "if, then, else", or conditional logic. In SQL, we represent this logic with the CASE statement, as follows:

```
SELECT
   CASE
      WHEN elevation < 500 THEN 'Low'
       WHEN elevation BETWEEN 500 AND 1999 THEN 'Medium'
       WHEN elevation >= 2000 THEN 'High'
       ELSE 'Unknown'
   END AS elevation_tier
   , COUNT(*)
FROM airports
GROUP BY 1; // specifies the coloms to use
```

In the above statement, END is required to terminate the statement, but ELSE is optional. If ELSE is not included, the result will be NULL. Also notice the shorthand method of referencing **columns to use in GROUP BY**, so we don't have to rewrite the entire Case Statement.

Sometimes you want to look at an entire result set, but want to implement conditions on certain aggregates.
For instance, maybe you want to identify the total amount of airports as well as the total amount of airports with high elevation in the same result set. We can accomplish this by putting a **CASE WHEN** statement in the aggregate.

```
SELECT    state,
   COUNT(CASE WHEN elevation >= 2000 THEN 1 ELSE NULL END) as
count_high_elevation_aiports
FROM airports
GROUP BY state;
```

Using the same pattern, write a query to count the number of low elevation airports by state where low elevation is defined as less than 1000 ft.
Be sure to alias the counted airports as count_low_elevation_airports.

```
SSELECT state,
   COUNT(CASE
      WHEN elevation < 1000 THEN 1 ELSE NULL END)
      as count_low_elevation_airports
FROM airports
GROUP BY state;
```

6.
We can do that same thing for other aggregates like SUM(). For instance, if we wanted to **sum the total flight distance and compare that to the sum of flight distance from a particular airline** (in this case, United Airlines) by origin airport, we could run the following query:

SELECT **origin, sum(**distance) as total_flight_distance, **sum(CASE WHEN** carrier = 'UA' **THEN d**istance **ELSE** 0 **END**) as total_united_flight_distance
FROM flights
GROUP BY origin;

Oftentimes we'd like to combine aggregates, to create **percentages or ratios.** In the instance of the last query, we might want to find out the **percent of flight distance that is from United by origin airport**. We can do this simply by using the mathematical operators we need in SQL:
SELECT **origin,**
100.0*(sum(CASE WHEN carrier = 'UN' **THEN** distance **ELSE** 0 **END)/ sum(**distance)) as percentage_flight_distance_from_united FROM flights
GROUP BY origin;

7.

Find **the percentage of high elevation airports** (elevation >= 2000) **by state from the airports table**.
In the query, alias the percentage column as percentage_high_elevation_airports.

SELECT state, **100.0*(SUM(CASE WHEN elevation >= 2000 THEN 1 ELSE 0 END))/COUNT(*)** AS percentage_high_elevation_airports FROM airports
GROUP BY state;

- **Conditional Aggregates** are aggregate functions the compute a result set based on a given set of conditions.
- NULL can be used to denote an empty field value
- CASE statements allow for custom classification of data
- CASE statements can be used inside aggregates (like SUM() and COUNT()) to provide filtered measures