**Create a Table**

In the previous lesson, we created a table for you to start querying. In this lesson, you will create the table by reading directly from the data source and specifying a **schema**. A schema describes the structure of your data. It contains column names and the type of data in each column. All tables must have an associated schema; if you do not explicitly define one, Spark may be able to infer it.

In the cell below, we define the schema as we create the table. This data has the following schem

**DROP TABLE IF EXISTS movieRatings;**

CREATE TABLE movieRatings (

userId INT,

movieId INT,

rating FLOAT,

timeRecorded INT

) USING csv OPTIONS (

PATH "/mnt/training/movies/20m/ratings.csv",

header "true"

);

**DROP TABLE IF EXISTS People10M;**

CREATE TABLE People10M

USING parquet

OPTIONS (

path "/mnt/training/dataframes/people-10m.parquet",

header "true");

We can view the schema for this table by using the DESCRIBE function. The **schema** is a list that defines the columns in a table and the datatypes within those columns.

DESCRIBE People10M;

## Create temporary views

Next, we create two temporary views so that the actual join will be easy to read/write.

**CREATE OR REPLACE TEMPORARY VIEW SSADistinctNames AS**

SELECT DISTINCT firstName AS ssaFirstName

FROM SSANames;

**CREATE OR REPLACE TEMPORARY VIEW PeopleDistinctNames AS**

SELECT DISTINCT firstName

FROM People10M

## Perform join

Now, we can use the view names to **join** the two data sets. If you are new to using SQL, you may want to learn more about the different types of joins you can perform. This [wikipedia article](https://en.wikipedia.org/wiki/Join_(SQL" \t "_blank) offers complete explanations, with pictures and sample SQL code.

By default, the join type shown here is INNER. That means the results will contain the intersection of the two sets, and any names that are not in both sets will not appear. Note, becuase it is default, we did not specify the join type.

SELECT firstName

FROM PeopleDistinctNames

JOIN SSADistinctNames ON firstName = ssaFirstName

## Cast as timestamp

We use the CAST() function to show the timestamp as a human-readable time and date.

SELECT

rating,

CAST(timeRecorded as timestamp)

FROM

movieRatings;

## Create temporary view

We will create a temporary view that we can easily refer to the data we want to include in our visualization. For this data, we can investigate whether there are any patterns in the ratings when grouped by month. To do that, we use the ROUND() and AVG() functions to calculate the average rating and limit it to 3 decimal places. Then, extract the month from the timeRecorded column after casting it as a timestamp. The AVG() is calculated over the course of a month, as specified in the GROUP BY clause.

CREATE

OR REPLACE TEMPORARY VIEW ratingsByMonth AS

SELECT

ROUND(AVG(rating), 3) AS avgRating,

month(CAST(timeRecorded as timestamp)) AS month

FROM

movieRatings

GROUP BY

month;