databricks3.3 Data Visualization

Data Visualization with Databricks

In this lesson, you'll learn how to create and share data visualizations in Databricks.

By the end of the lesson, you will be able to:

- Create a table with a specified schema
- Cast a column as a timestamp and extract day, month, or year
- Use in-notebook visualizations to see your data

Step 1: Read through and run all the cells in this notebook.
Step 2: View the corresponding video to see instructions for visualizing and sharing data.

Getting Started

Run the following cell to connect your workspace to the appropriate data source.

```
%run ../Includes/Classroom-Setup
```

Mounting course-specific datasets to **/mnt/training**...
Datasets are already mounted to **/mnt/training** from **s3a://databricks-corp-training/common**

```
res1: Boolean = false
```

```
res2: Boolean = false
```

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 schema:

Column Name	Type
userId	INT
movieId	INT
rating	FLOAT
timeRecorded	INT

Notice that it is defined right after the `CREATE TABLE` statement with the name of each column followed by the datatypes within the column. The whole group of columns is surround by parentheses and each individual column is spereated by a comma.

```
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"
);
```

OK

Preview the data

This table contains a little more than 20 million records of movies ratings submitted by users. Note that the timestamp is an integer value recorded in UTC time.

```
SELECT
  *
FROM
  movieRatings;
```

	userId	movieId	rating	timeRecorded
1	1	2	3.5	1112486027
2	1	29	3.5	1112484676
3	1	32	3.5	1112484819
4	1	47	3.5	1112484727
5	1	50	3.5	1112484580
6	1	112	3.5	1094785740
7	1	151	4	1094785734

Truncated results, showing first 1000 rows.

```
SELECT
  count(*) as Total_Registers,
  count(Distinct movieid) as Distinct_movies_Total
FROM
  movieRatings;
```

	Total_Registers	Distinct_movies_Total
1	20000263	26744

Showing all 1 rows.

```
SELECT
  movieid,
  count(*)
FROM
  movieRatings
GROUP BY movieid
ORDER BY count(*) DESC;
```

	movieid	count(1)
1	296	67310
2		

3	318	63366
4	593	63299
5	480	59715
6	260	54502
7	110	53769

Truncated results, showing first 1000 rows.

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;
```

	rating	timeRecorded
1	3.5	2005-04-02T23:53:47.000+0000
2	3.5	2005-04-02T23:31:16.000+0000
3	3.5	2005-04-02T23:33:39.000+0000
4	3.5	2005-04-02T23:32:07.000+0000
5	3.5	2005-04-02T23:29:40.000+0000
6	3.5	2004-09-10T03:09:00.000+0000
7	4	2004-09-10T03:08:54.000+0000

Truncated results, showing first 1000 rows.

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.

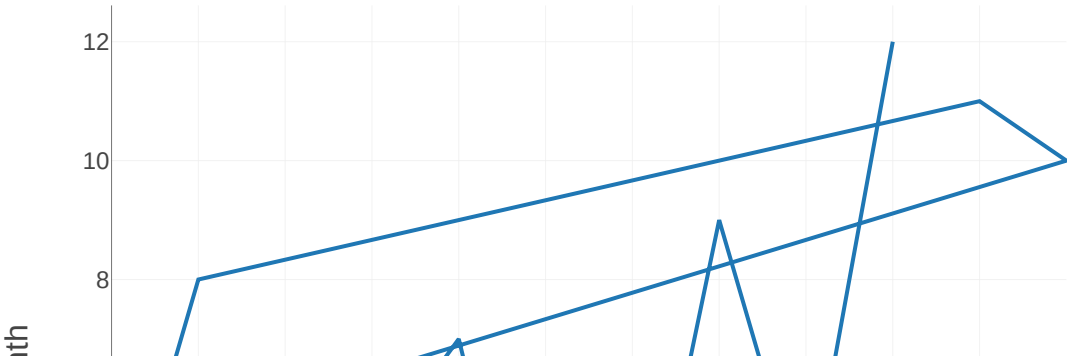
```
CREATE
OR REPLACE TEMPORARY VIEW ratingsByMonth AS
SELECT
  ROUND(AVG(rating), 3) AS avgRating,
  month(CAST(timeRecorded as timestamp)) AS month
FROM
  movieRatings
GROUP BY
  month;
```

OK

Visualize the data

Run the next cell to see the view we just defined ordered by `avgRating` from least to greatest. The results will appear as a table. In the next section, you will receive video instruction that will show you how to display this table as a chart.

```
SELECT
  *
FROM
  ratingsByMonth
ORDER BY
  avgRating;
```




```
%run ../Includes/Classroom-Cleanup
```

Citation

Access original data and backround information here:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872 (http://dx.doi.org/10.1145/2827872)