databricks8.1 Using Delta

Using Delta

Moovio, the fitness tracker company, is in the process of migrating non-Delta workloads to Delta Lake. You have access to the files that hold current data that you can experiment with while learning about how to work with Delta Lake. Creating Delta tables is as easy as issuing the command, `USING DELTA`. Get started by reading through the following cells and run the corresponding queries to create and modify Delta tables.

Getting started

Run the cell below to set up your classroom environment.

```
%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 table

To start, we will create a table of the raw data we've been provided. We're using only a small sample of the available data, so this set is limited to 5 devices over the course of one month. The raw files are in the `.json` format.

```
DROP TABLE IF EXISTS health_tracker_data_2020_01;

CREATE TABLE health_tracker_data_2020_01
USING json
OPTIONS (
  path
"dbfs:/mnt/training/healthcare/tracker/raw.json/health_tracker_data_2020_1.json
",
  inferSchema "true"
  );
```

OK

Preview data

Before we do anything else, let's quickly inspect the data by viewing a sample.

```
SELECT * FROM health_tracker_data_2020_01 TABLESAMPLE (5 ROWS)
```

	month	value
1	2020-01	▸ {"device_id": 0, "heartrate": 101.3720506847, "name": "Deborah Powell", "time
2	2020-01	▸ {"device_id": 0, "heartrate": 98.7361247811, "name": "Deborah Powell", "time"
3	2020-01	▸ {"device_id": 0, "heartrate": 99.9724260825, "name": "Deborah Powell", "time"
4	2020-01	▸ {"device_id": 0, "heartrate": 99.8718710286, "name": "Deborah Powell", "time"
5	2020-01	▸ {"device_id": 0, "heartrate": 98.3444848503, "name": "Deborah Powell", "time"

Showing all 5 rows.

Create Delta table

This example, so far, is of a Bronze level table. We can display the raw data, but it is not easily queryable. Our next step is to create a cleaned Silver table. This table may flow into several business aggregate Gold level tables later on. In this step, we'll focus on creating an easily queryable table that includes most or all of the data, with all columns accurately typed and object properties unpacked into individual columns.

Recall that a Delta table consists of three things:

1. The Delta files (in object storage)
2. The Delta Transaction Log (https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html) saved with the Delta files in object storage.
3. The Delta table registered in the Metastore

```
CREATE OR REPLACE TABLE health_tracker_silver
USING DELTA
PARTITIONED BY (p_device_id)
LOCATION "/health_tracker/silver" AS (
SELECT
  value.name,
  value.heartrate,
  CAST(FROM_UNIXTIME(value.time) AS timestamp) AS time,
  CAST(FROM_UNIXTIME(value.time) AS DATE) AS dte,
  value.device_id p_device_id
FROM
  health_tracker_data_2020_01
)
```

Query returned no results

Great! You have created your first Delta table! Run the `DESCRIBE DETAIL` command to view table details. You can see that the table format is `delta` and it is stored in the location you specified.

```
DESCRIBE DETAIL health_tracker_silver
```

	format	id	name
1			

Showing all 1 rows.

Read in new data

Recall that we created that table with just one month of data. Now let's see how we can add new data to that table.

Run the cell below to read in the new raw file.

```
DROP TABLE IF EXISTS health_tracker_data_2020_02;

CREATE TABLE health_tracker_data_2020_02
USING json
OPTIONS (
  path
"dbfs:/mnt/training/healthcare/tracker/raw.json/health_tracker_data_2020_2.json
",
  inferSchema "true"
  );
```

OK

Append files

We can append the next month of of records to the existing table using the `INSERT INTO` command. We will transform the new data to match the existing schema.

```
INSERT INTO
  health_tracker_silver
SELECT
  value.name,
  value.heartrate,
  CAST(FROM_UNIXTIME(value.time) AS timestamp) AS time,
  CAST(FROM_UNIXTIME(value.time) AS DATE) AS dte,
  value.device_id p_device_id
FROM
  health_tracker_data_2020_02
```

	num_affected_rows	num_inserted_rows
1	3408	3408

Showing all 1 rows.

Time Travel: Count records in the previous table

Let's count the records to verify that the append went as expected. First, we can write a query to show the count before we appended new records. Delta Lake can query an earlier version of a Delta table using a feature known as time travel (https://docs.databricks.com/delta/quick-start.html#query-an-earlier-version-of-the-table-time-travel).

We demonstrate querying the data as of version 0, which is the initial conversion of the table from Parquet.

```
5 devices * 24 hours * 31 days = 3720 records
```

```
SELECT COUNT(*) FROM health_tracker_silver VERSION AS OF 0
```

	count(1)
1	3720

Showing all 1 rows.

Count records in our current table

Now, let's count the records to see if our new data was appended as expected. Note that this data is from February 2020, which had 29 days because 2020 was a leap year. We are still working with 5 devices, with heartrate readings occurring once an hour.

```
5 devices * 24 hours * 29 days + 3720 = 7200 records
```

```
SELECT COUNT(*) FROM health_tracker_silver
```

	count(1)
1	7128

Showing all 1 rows.

Find missing records by device

Let's see if we can identify which device(s) are missing records.

The absence of records from the last few days of the month shows a phenomenon that may often occur in a production data pipeline: **late-arriving data**. This can create problems in some of the other data storage and management models we talked about. If our analytics runs on stale or incomplete data, we may draw

```
SELECT p_device_id, COUNT(*) FROM health_tracker_silver GROUP BY p_device_id
```

	p_device_id	count(1)
1	0	1440
2	1	1440
3	3	1440
4	2	1440
5	4	1368

Showing all 5 rows.

Plot Records

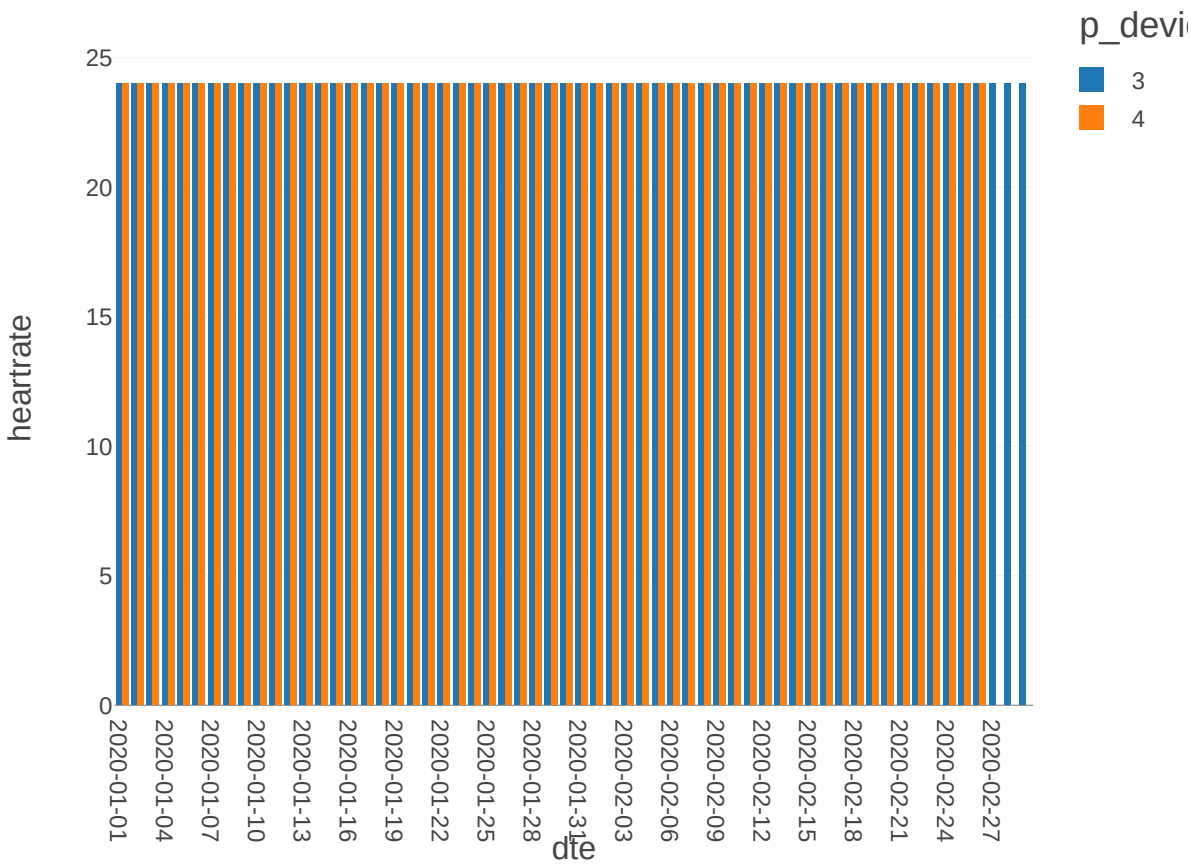
We can run a query and use visualization tools to find out more about which dates or times are missing. For this query, it may be helpful to compare two devices, even though we're showing only one is missing data. Run the cell below to query the table. Then, click on the chart icon to plot the data.

To set up your graph:

- Click `Plot Options`
- Drag `dte` into the Keys dialog

- Drag `p_device_id` into the Series Groupings dialog
- Drag `heartrate` into the values dialog
- Choose `COUNT` as your Aggregation type (the dropdown in the lower left corner)
- Select "Bar Chart" as your display type.

```
SELECT * FROM health_tracker_silver WHERE p_device_id IN (3,4)
```



Find Broken Readings

It's always useful to check for errant readings. Think about this scenario. Is there any reading would seem impossible?

Since this is heartrate date, we should expect that everyone who is using the tracker has a heartbeat. Let's check the data to see if we've got any data that might point to a faulty reading.

```
CREATE OR REPLACE TEMPORARY VIEW broken_readings
AS (
  SELECT COUNT(*) as broken_readings_count, dte FROM health_tracker_silver
  WHERE heartrate < 0
  GROUP BY dte
  ORDER BY dte
)
```

OK

View broken readings

Run the cell and then create a a visualization that will help us get a sense of how many broken readings exist and how they are spread across the data.

To visualize this view:

- Run the cell
- Click the chart icon
- Choose 'dte' as the Key and `broken_readings_count` as Values.

 You should notice that most days have at least one broken reading and that some

```
SELECT * FROM broken_readings;
```

	broken_readings_count	dte
1	1	2020-01-01
2	3	2020-01-02
3	3	2020-01-05
4	3	2020-01-06
5	1	2020-01-08
6	1	2020-01-12
7	1	2020-01-15

Showing all 35 rows.

Clean-up

Run the next cell to clean up your classroom enviroment

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

Great job! You're officially working with Delta Lake! In the next reading, you'll continue your work to repair the broken data and missing values we discovered here.