databricks6.4 Lab - Sharing Insights

Lab 3 - Sharing Insights

Module 6 Assignment

In this lab, we will explore a small mock data set from a group of data centers. You'll see that is is similar to the data you have been working with, but it contains a few new columns and it is structured slightly differently to test your skills with hierarchical data manipulation.

⭐ In this assignment you will:

- Apply higher-order functions to array data
- Apply advanced aggregation and summary techniques to process data
- Present data in an interactive dashboard or static file

As you work through the following tasks, you will be prompted to enter selected answers in Coursera. Find the quiz associated with this lab to enter your answers.

Run the cell below to prepare this workspace for the lab.

```
%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
```

Exercise 1: Create a table

Summary: Create a table.

Use this path to access the data:
`/mnt/training/iot-devices/data-centers/energy.json`

Steps to complete:

- Write a `CREATE TABLE` statement for the data located at the endpoint listed above
- Use json as the file format

```
CREATE TABLE IF NOT EXISTS energyRaw
USING
json
OPTIONS (
  path "/mnt/training/iot-devices/data-centers/energy.json",
  inferschema "true"
)
```

OK

Exercise 2: Sample the table

Summary: Sample the table to get a closer look at a few rows

Steps to complete:

- Write a query that allows you to see a few rows of the data

```
SELECT * FROM energyRaw
ORDER BY
RAND() LIMIT 6;
```

	battery_level	co2_level	device_id	device_type	signal	t
1	[5, 7, 6]	[1280, 1131, 1292]	22	sensor-igauge	[17, 17, 17]	2
2	[8, 9, 9]	[1381, 1434, 1401]	29	sensor-istick	[24, 20, 20]	1
3	[6, 7, 7]	[1613, 1589, 1529]	4	sensor-istick	[19, 21, 19]	2
4	[5, 5, 4]	[1337, 1092, 1307]	33	sensor-inest	[24, 24, 22]	2

Showing all 6 rows.

Exercise 3: Create view

Summary: Create a temporary view that displays the timestamp column as a timestamp.

Steps to complete:

- Create a temporary view named `DCDevices`
- Convert the `timestamp` column to a timestamp type. Refer to the Datetime patterns (https://spark.apache.org/docs/latest/sql-ref-datetime-pattern.html#) documentation for the formatting information.
- (Optional) Rename columns to use camelCase

```
DESCRIBE energyRaw;
```

	col_name	data_type	comment
1	battery_level	array<bigint>	null
2	co2_level	array<bigint>	null
3	device_id	bigint	null
4	device_type	string	null
5	signal	array<bigint>	null
6	temps	array<bigint>	null
7	timestamp	string	null

Showing all 7 rows.

```
DROP VIEW IF EXISTS DCDevices;
CREATE
OR REPLACE TEMPORARY VIEW DCDevices AS
SELECT battery_level batteryLevel,
  co2_level co2Level,
  device_id deviceId,
  device_type deviceType,
  signal,
  temps,
  to_date(split(timestamp, ' ')[0], "yyyy/MM/dd")
  timestamp
FROM energyRaw;
```

OK

Exercise 4: Flag records with defective batteries

Summary: When a battery is malfunctioning, it can report negative battery levels. Create a new boolean column `needService` that shows whether a device needs service.

Steps to complete:

- Write a query that shows which devices have malfunctioning batteries
- Include columns `batteryLevel`, `deviceId`, and `needService`
- Order the results by `deviceId`, and then `batteryLevel`
- **Answer the corresponding question in Coursera**

```
SELECT batteryLevel,
  deviceId,
  EXISTS(batteryLevel, i -> i < 0) needService
FROM DCDevices
WHERE
EXISTS(batteryLevel, i -> i < 0)=true
Order by deviceId, batteryLevel;
```

	batteryLevel	deviceId	needService
1	[-4, -1, -1]	0	true
2	[-4, -1, -1]	0	true
3	[-4, -1, 0]	0	true
4	[-3, -3, -4]	0	true
5	[-3, -2, -2]	0	true
6	[-3, -2, -2]	0	true
7	[-3, -2, -2]	0	true

Truncated results, showing first 1000 rows.

```
SELECT deviceId,
  deviceType,
  FILTER(co2Level, iterator -> iterator > 1400) highCO2,
  timestamp time
FROM DCDevices
WHERE
EXISTS(co2Level, iterator -> iterator > 1400) = true
Order by deviceId, highCO2;
```

	deviceId	deviceType	highCO2	time
1	0	sensor-ipad	▸ [1401]	2019-06-29
2	0	sensor-istick	▸ [1401]	2019-08-16
3	0	sensor-ipad	▸ [1401]	2019-06-01
4	0	sensor-istick	▸ [1401]	2019-08-02
5	0	sensor-istick	▸ [1401]	2019-07-01
6	0	sensor-ipad	▸ [1401]	2019-08-12
7	0	sensor-igauge	▸ [1401]	2019-06-21

Truncated results, showing first 1000 rows.

Exercise 6: Create a partitioned table

Summary: Create a new table partitioned by `deviceId`

Steps to complete:

- Include all columns
- Create the table using Parquet
- Rename the partitioned column `p_deviceId`
- Run a `SELECT *` to view your table.

Answer the corresponding question in Coursera

```
SELECT * FROM DCDevices;
```

	batteryLevel	co2Level	deviceId	deviceType	signal	te
1	▸ [3, 3, 2]	▸ [1343, 1595, 1405]	0	sensor-istick	▸ [24, 24, 25]	▸ 2
2	▸ [1, 1, 2]	▸ [1213, 1346, 1247]	1	sensor-inest	▸ [22, 24, 24]	▸ 3

3	[3, 1, 3]	[1261, 1216, 1258]	2	sensor-ipad	[25, 24, 28]	4
4	[4, 3, 4]	[1270, 1257, 1228]	3	sensor-igauge	[24, 22, 24]	1

Truncated results, showing first 1000 rows.

```
%fs rm -r dbfs:/user/hive/warehouse/table1
```

```
res3: Boolean = true
```

```
DROP TABLE IF EXISTS Table1;
CREATE TABLE IF NOT EXISTS Table1
USING parquet
--WITH TempTable
PARTITIONED BY (p_deviceId)
AS (
  SELECT
  batteryLevel,
  co2Level,
  deviceId AS p_deviceId,
  deviceType,
  signal,
  temps,
  timestamp
FROM DCDevices
);
```

```
SELECT * FROM Table1;
```

	batteryLevel	co2Level	deviceType	signal	temps	ti
1	[1, 1, 1]	[1446, 1421, 1420]	sensor-igauge	[14, 14, 14]	[33, 26, 27, 25]	2(
2	[6, 7, 6]	[1173, 1202, 1402]	sensor-istick	[27, 26, 26]	[28, 30, 35, 34]	2(
3	[2, 0, 2]	[1189, 1121, 1071]	sensor-igauge	[15, 14, 14]	[20, 24, 25, 22]	2(
4	[4, 6, 5]	[1148, 1253, 1192]	sensor-inest	[22, 21, 22]	[32, 25, 28, 25]	2(

Truncated results, showing first 1000 rows.

Exercise 7: Visualize average temperatures

```
DROP VIEW IF EXISTS View1;
CREATE OR REPLACE TEMPORARY VIEW View1
AS
SELECT
batteryLevel,
  co2Level,
  p_deviceId,
  deviceType,
  signal,
  temps,
  REDUCE(temps, 0, (c, acc) -> c + acc, acc ->(acc div size(temps))) as
avgTemps,
  timestamp
FROM
  (
  SELECT
  batteryLevel,
  co2Level,
  p_deviceId,
  deviceType,
  signal,
  TRANSFORM(temps, t -> int(t)) temps,
  timestamp
  FROM Table1
  )
;
SELECT * FROM View1;
```

	batteryLevel	co2Level	p_deviceId	deviceType	signal	t
1	▸ [1, 1, 1]	▸ [1446, 1421, 1420]	0	sensor-igauge	▸ [14, 14, 14]	2
2	▸ [6, 7, 6]	▸ [1173, 1202, 1402]	0	sensor-istick	▸ [27, 26, 26]	3
3	▸ [2, 0, 2]	▸ [1189, 1121, 1071]	0	sensor-igauge	▸ [15, 14, 14]	2
4	▸ [4, 6, 5]	▸ [1148, 1253, 1192]	0	sensor-inest	▸ [22, 21, 22]	2

Truncated results, showing first 1000 rows.

Exercise 8: Create a widget

```
DROP TABLE IF EXISTS Table2;
CREATE TABLE Table2
AS
SELECT
batteryLevel,
  co2Level,
  p_deviceId,
  deviceType,
  signal,
  temps,
  REDUCE(temps, 0, (c, acc) -> c + acc, acc ->(acc div size(temps))) as
avgTemps,
  timestamp
FROM
  (
  SELECT
  batteryLevel,
  co2Level,
  CAST(p_deviceId AS STRING),
  deviceType,
  signal,
  TRANSFORM(temps, t -> int(t)) temps,
  timestamp
  FROM Table1
  )
;
SELECT * FROM Table2;
```

	batteryLevel	co2Level	p_deviceId	deviceType	signal	t
1	▸ [8, 10, 11]	▸ [1541, 1167, 1344]	11	sensor-inest	▸ [17, 19, 18]	2
2	▸ [13, 13, 11]	▸ [1102, 1430, 1233]	11	sensor-ipad	▸ [31, 31, 31]	1
3	▸ [3, 2, 4]	▸ [1324, 1185, 1234]	11	sensor-igauge	▸ [22, 22, 21]	2
4	▸ [6, 5, 8]	▸ [1256, 1376, 1263]	11	sensor-ipad	▸ [23, 21, 23]	2

Truncated results, showing first 1000 rows.

```
DESCRIBE Table2
```

	col_name	data_type	comment
1	batteryLevel	array<bigint>	

2	co2Level	array<bigint>	
3	p_deviceId	string	
4	deviceType	string	
5	signal	array<bigint>	
6	temps	array<int>	
7	avgTemps	bigint	

Showing all 11 rows.

```
CREATE WIDGET DROPDOWN selectedDeviceId DEFAULT "0" CHOICES
SELECT
  DISTINCT p_deviceId
FROM
Table2
;
```

OK

Exercise 9: Use the widget in a query

```
SELECT
  p_deviceId,
  REDUCE(temps, 0, (c, acc) -> c + acc, acc ->(acc div size(temps))) as
avgTemps
FROM
  Table2
WHERE p_deviceId = getArgument("selectedDeviceId")
--GROUP BY p_deviceId
;
```

	p_deviceId	avgTemps
1	0	27
2	0	31
3	0	22
4	0	27
5	0	40
6	0	28
7	0	24

Truncated results, showing first 1000 rows.

```
REMOVE WIDGET selectedDeviceId
```

```
Error in SQL statement: InputWidgetNotDefined: No input widget named selecte
dDeviceId is defined
```

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