databricksAssignment

Engineering Data Pipelines

Module 3 Assignment

In this assignment you:

- Create a table with persistent data and a specified schema
- Populate table with specific entries
- Change partition number to compare query speeds

For each **bold** question, input its answer in Coursera.

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

Data mounted to /mnt/davis ...

OK

Create a table whose data will remain after you drop the table and after the cluster shuts down. Name this table `newTable` and specify the location to be at `/tmp/newTableLoc`

Set up the table to have the following schema:

```
`Address` STRING,
`City` STRING,
`Battalion` STRING,
`Box` STRING,
```

Run the following cell first to remove any files stored at `/tmp/newTableLoc` before creating our table. Be sure to first re-run that cell each time you create `newTable`.

```
%python
# removes files stored at '/tmp/newTableLoc'
dbutils.fs.rm("/tmp/newTableLoc", True)
```

```
Out[16]: False
```

```
-- TODO
CREATE TABLE IF NOT EXISTS newTable (
`Address` STRING,
`City` STRING,
`Battalion` STRING,
`Box` STRING)
USING csv
OPTIONS (
path '/tmp/newTableLoc')
```

OK

Check that the data type of each column is what we want.

Question 1

What type of table is `newTable`? "EXTERNAL" or "MANAGED"?

```
DESCRIBE EXTENDED newTable
--EXTERNAL
```

	col_name	data_type
1	Address	string
2	City	string

3	Battalion	string
4	Box	string
5		
6	# Detailed Table Information	
7	Database	databricks
8	Table	newtable

Showing all 19 rows.

Run the following cell to read in the data stored at `/mnt/davis/fire-calls/fire-calls-truncated.json`. Check that the columns of the data are of the correct types (not all strings).

```
CREATE OR REPLACE TEMPORARY VIEW  fireCallsJSON (
  `ALS Unit` boolean,
  `Address` string,
  `Available DtTm` string,
  `Battalion` string,
  `Box` string,
  `Call Date` string,
  `Call Final Disposition` string,
  `Call Number` long,
  `Call Type` string,
  `Call Type Group` string,
  `City` string,
  `Dispatch DtTm` string,
  `Entry DtTm` string,
  `Final Priority` long,
  `Fire Prevention District` string,
  `Hospital DtTm` string,
  `Incident Number` long,
  `Location` string,
  `Neighborhooods - Analysis Boundaries` string,
  `Number of Alarms` long,
  `On Scene DtTm` string,
  `Original Priority` string,
  `Priority` string,
  `Received DtTm` string,
  `Response DtTm` string,
  `RowID` string,
  `Station Area` string,
  `Supervisor District` string,
  `Transport DtTm` string,
  `Unit ID` string,
  `Unit Type` string,
  `Unit sequence in call dispatch` long,
  `Watch Date` string,
  `Zipcode of Incident` long
)
USING JSON
OPTIONS (
    path "/mnt/davis/fire-calls/fire-calls-truncated.json"
);

DESCRIBE fireCallsJSON
```

	col_name	data_type	comment
1	ALS Unit	boolean	null
2	Address	string	null
3	Available DtTm	string	null
4	Battalion	string	null

5	Box	string	null
6	Call Date	string	null
7	Call Final Disposition	string	null
8	Call Number	bigint	null

Showing all 34 rows.

Take a look at the table to make sure it looks correct.

```
%fs ls /mnt/davis/fire-calls/fire-calls-truncated.json
```

	path	name	size
1	dbfs:/mnt/davis/fire-calls/fire-calls-truncated.json	fire-calls-truncated.json	221798942

Showing all 1 rows.

```
SELECT * FROM fireCallsJSON
```

	ALS Unit	Address	Available DtTm
1	false	4TH ST/CHANNEL ST	04/12/2000 09:45:28 PM
2	false	1800 Block of IRVING ST	04/12/2000 09:49:52 PM
3	false	0 Block of SOUTH VAN NESS AVE	04/12/2000 11:42:43 PM
4	true	CLAYTON ST/PARNASSUS AV	04/13/2000 12:33:18 AM
5	true	500 Block of 38TH AVE	04/13/2000 02:40:25 AM
6	true	200 Block of MADRID ST	04/13/2000 09:26:54 AM
7	false	2800 Block of BROADWAY	04/13/2000 09:39:36 AM
8	true	2500 Block of OCEAN AVE	04/13/2000 01:16:20 PM

Showing the first 1000 rows.

Now let's populate `newTable` with some of the rows from the `fireCallsJSON` table you just loaded. We only want to include fire calls whose `Final Priority` is `3`.

```
INSERT INTO newTable
SELECT Address, City, Battalion, Box
FROM fireCallsJSON
WHERE `Final Priority` = 3
```

OK

```
SELECT `Final Priority`, Battalion FROM fireCallsJSON
WHERE `Final Priority` = 3
```

	Final Priority	Battalion
1	3	B03
2	3	B05
3	3	B07
4	3	B09
5	3	B04
6	3	B04
7	3	B03
8	3	B01

Showing the first 1000 rows.

Question 2

How many rows are in `newTable`?

```
SELECT COUNT(*) FROM newTable
```

	count(1)
1	191039

Showing all 1 rows.

Sort the rows of `newTable` by ascending `Battalion`.

Question 3

What is the "Battalion" of the first entry in the sorted table?

```
SELECT * FROM newTable
SORT BY Battalion
```

	Address	City	Battalion	Box
1	300 Block of GRANT AVE	SF	B01	1315
2	900 Block of GRANT AVE	SF	B01	1312
3	300 Block of COLUMBUS AVE	SF	B01	1311
4	300 Block of GEARY ST	SF	B01	1411
5	900 Block of NORTH POINT ST	SF	B01	1623
6	1300 Block of SACRAMENTO ST	SF	B01	1465
7	600 Block of UNION ST	SF	B01	1334
8	500 Block of GEARY ST	SF	B01	1462

Showing the first 1000 rows.

Let's see how this table is stored in our file system.

Note: You should have specified the location of the table to be `/tmp/newTableLoc` when you created it.

```
%fs ls dbfs:/tmp/newTableLoc
```

First run the following cell to check how many partitions are in this table. Did the number of partitions match the number of files our data was stored as?

Let's try increasing the number of partitions to 256. Create this as a new table and call it `newTablePartitioned`

```
-- TODO
CREATE TABLE IF NOT EXISTS newTablePartitioned
AS
SELECT /*+ REPARTITION(256) */ *
FROM newTable
```

OK

Now let's take a look at how this new table is stored.

```
DESCRIBE EXTENDED newTablePartitioned
```

	col_name	data_type
1	Address	string
2	City	string
3	Battalion	string
4	Box	string
5		
6	# Detailed Table Information	
7	Database	databricks
8	Table	newtablepartitioned

Showing all 19 rows.

Copy the location of the `newTablePartitioned` from the table above and take a look at the files stored at that location. Now how many parts is our data stored?

```
%fs ls dbfs:/user/hive/warehouse/databricks.db/newtablepartitioned
```

Now sort the rows of `newTablePartitioned` by ascending `Battalion` and compare how long this query takes.

Question 4

Was this query faster or slower on the table with increased partitions?

```
-- It is much slower, taking around 6.82sec comparing to the previous one
which took 0.74sec
SELECT * FROM newTablePartitioned SORT BY `Battalion`
```

	Address	City	Battalion	Box
1	POWELL ST/SUTTER ST	SF	B01	1362
2	1200 Block of PACIFIC AVE	SF	B01	1511
3	500 Block of POST ST	SF	B01	1451
4	300 Block of CALIFORNIA ST	SF	B01	1162

5	600 Block of MONTGOMERY ST	SF	B01	1233
6	2500 Block of MASON ST	SF	B01	1425
7	500 Block of SACRAMENTO ST	SF	B01	1166
8	2200 Block of STOCKTON ST	SF	B01	1341

Showing the first 1000 rows.

Run the following cell to see where the data of the original `newTable` is stored.

```
%fs ls dbfs:/tmp/newTableLoc
```

	path
1	dbfs:/tmp/newTableLoc/_SUCCESS
2	dbfs:/tmp/newTableLoc/_committed_4946010204335988767
3	dbfs:/tmp/newTableLoc/_started_4946010204335988767
4	dbfs:/tmp/newTableLoc/part-00000-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
5	dbfs:/tmp/newTableLoc/part-00001-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
6	dbfs:/tmp/newTableLoc/part-00002-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
7	dbfs:/tmp/newTableLoc/part-00003-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
8	dbfs:/tmp/newTableLoc/part-00004-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c

Showing all 11 rows.

Now drop the table `newTable`.

```
DROP TABLE newTable;

-- The following line should error!
-- SELECT * FROM newTable;
```

OK

Question 5

Does the data stored within the table still exist at the original location (`dbfs:/tmp/newTableLoc`) after you dropped the table? (Answer "yes" or "no")

```
%fs ls dbfs:/tmp/newTableLoc
--Yes
```

	path
1	dbfs:/tmp/newTableLoc/_SUCCESS
2	dbfs:/tmp/newTableLoc/_committed_4946010204335988767
3	dbfs:/tmp/newTableLoc/_started_4946010204335988767
4	dbfs:/tmp/newTableLoc/part-00000-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
5	dbfs:/tmp/newTableLoc/part-00001-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
6	dbfs:/tmp/newTableLoc/part-00002-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
7	dbfs:/tmp/newTableLoc/part-00003-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c
8	dbfs:/tmp/newTableLoc/part-00004-tid-4946010204335988767-4abf2916-cb52-426b-9d26-e4c

Showing all 11 rows.