Accessing Data

Apache Spark™ and Databricks® have numerous ways to access your data.

In this lesson you

- Create a table from an existing file
- Create a table by uploading a data file from your local machine
- Mount an S3 bucket to DBFS
- Create tables for Databricks data sets to use throughout the course

The example below creates a table from the **ip-geocode.parquet** file (if it doesn't exist).

For Parquet files, you need to specify only one option: the path to the file.

A Parquet "file" is actually a collection of files stored in a single directory. The Parquet format offers features making it the ideal choice
Apache Parquet.

You can create a table from an existing DBFS file with a simple SQL `CREATE TABLE` statement. If you don't select a database, the database
us to delete these tables later.

Cmd 7

```
%sql CREATE DATABASE IF NOT EXISTS junk;

USE junk;

CREATE TABLE IF NOT EXISTS IPGeocode
USING parquet
OPTIONS (
  path "dbfs:/mnt/training/ip-geocode.parquet"
)
```

Cmd 6

Create a table from an existing file

DBFS (the Databricks File System) is the built-in, S3-backed, alternative to HDFS (the Hadoop Distributed File System).

Creating a table from an existing file in DBFS allows you to access the file as if it were a Spark table. It does **not** copy any data.

Cmd 8

The example below creates a table from the **ip-geocode.parquet** file (if it doesn't exist).

For Parquet files, you need to specify only one option: the path to the file.

A Parquet "file" is actually a collection of files stored in a single directory. The Parquet format offers features making it the ideal choice for storing "big data" on distributed file systems. For more information, see Apache Parquet.

You can create a table from an existing DBFS file with a simple SQL `CREATE TABLE` statement. If you don't select a database, the database called "default" is used. Here, we'll use a database called "junk", to remind us to delete these tables later.

Cmd 9

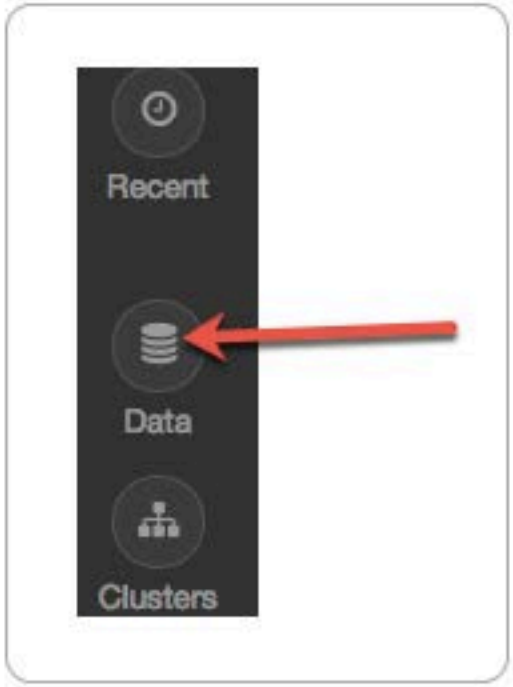
```
%sql
CREATE DATABASE IF NOT EXISTS junk;

USE junk;

CREATE TABLE IF NOT EXISTS IPGeocode
  USING parquet
  OPTIONS (
    path "dbfs:/mnt/training/ip-geocode.parquet"
  )
```

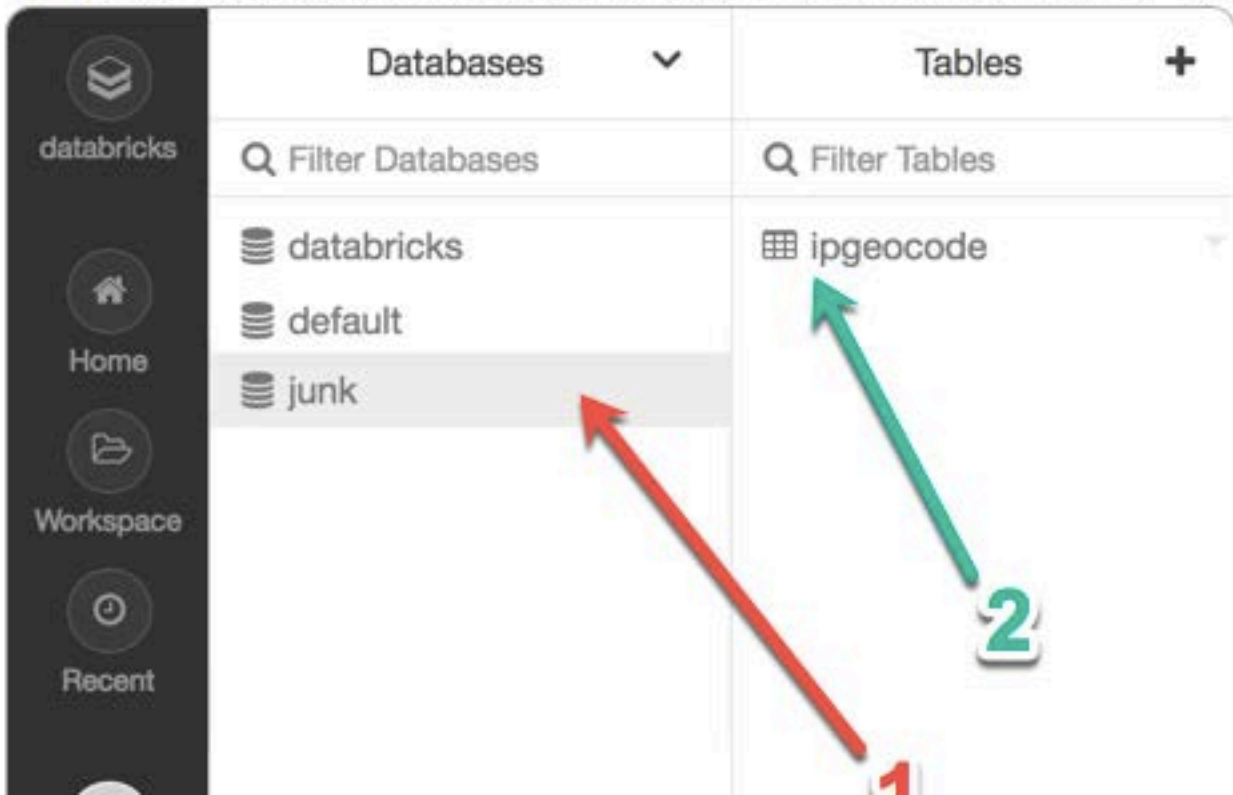
Now the table has been defined. You can see it in Databricks.

1. Click the **Data** icon on the left sidebar



2. Select the database **junk**.
3. Select the table **ipgeocode**.

   ⚠️ Right-click and open in a new tab, so you don't lose your place in this notebook



You see the schema of the table, along with a sample of its data.

ipgeocode

Refresh

Cluster

Spark 2.2

Schema:

col_name	data_type	comment
startingIP	decimal(38,0)	null
endingIP	decimal(38,0)	null
countryCode2	string	null
countryCode3	string	null
country	string	null
stateProvince	string	null
city	string	null
latitude	double	null
longitude	double	null

Sample Data:

startingIP	endingIP	countryCode2	countryCode3	country	stateProvince	city	latitude	longi
84549888	84551679	GE	GEO	Georgia	T'bilisi	Tbilisi	41.69411	44.83
92643328	92643583	GE	GEO	Georgia	T'bilisi	Tbilisi	41.69411	44.83
93848064	93848575	GE	GEO	Georgia	Ajaria	Bat'umi	41.64159	41.63
93853952	93854207	GE	GEO	Georgia	Imereti	K'ut'aisi	42.24961	42.69

Using A Personal Database

Any tables created or droped will be done so in the `junk` database.

However, every user of this system, if running this same code, will be altering the same tables.

In cases such as this one, it is often better to use a "personal" database.

For this reason, we will switch back to your personal database now.

We need to use the Spark programming API here only because we are unable to parameterize a `%sql` cell with the database we setup for you (as represtend by `databaseName`).

Cmd 13

```
# Programatically exectue a similar SQL command as above
spark.sql(f"USE {databaseName}")
```

Cmd 14

File formats other than Parquet

You can also create a table from other file formats.

One common format is CSV (comma-separated-values) for which you can specify:

- The file's delimiter, the default is "**,**"
- Whether the file has a header or not, the default is **false**
- Whether or not to infer the schema, the default is **false**

Cmd 16

In order to know which options to use, look at the first couple of lines of the file.

Take a look at the head of the file **/mnt/training/bikeSharing/data-001/day.csv.**

Cmd 17

```
%fs head /mnt/training/bikeSharing/data-001/day.csv --maxBytes=492
```

```
[Truncated to first 492 bytes]
instant,dteday,season,yr,mnth,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,casual,registered,cnt
1,2011-01-01,1,0,1,0,6,0,2,0.344167,0.363625,0.805833,0.160446,331,654,985
2,2011-01-02,1,0,1,0,0,0,2,0.363478,0.353739,0.696087,0.248539,131,670,801
3,2011-01-03,1,0,1,0,1,1,1,0.196364,0.189405,0.437273,0.248309,120,1229,1349
4,2011-01-04,1,0,1,0,2,1,1,0.2,0.212122,0.590435,0.160296,108,1454,1562
5,2011-01-05,1,0,1,0,3,1,1,0.226957,0.22927,0.436957,0.1869,82,1518,1600
```

Command took 0.33 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 10:46:22 PM on test-cluster

Cmd 18

Cmd 18

Spark can create a table from that CSV file, as well.

As you can see above:

- There is a header
- The file is comma separated (the default)
- Let Spark infer what the schema is

Cmd 19

```
%sql
CREATE TABLE IF NOT EXISTS BikeSharingDay
  USING csv
  OPTIONS (
    path "/mnt/training/bikeSharing/data-001/day.csv",
    inferSchema "true",
    header "true"
  )
```

OK

Command took 0.06 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 10:46:50 PM on test-cluster

Now the table is defined, view its contents with a simple select statement.

Cmd 21

```
%sql
SELECT * FROM BikeSharingDay
```

(1) Spark Jobs

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01T00:00:00.000+0000	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	2011-01-02T00:00:00.000+0000	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	2011-01-03T00:00:00.000+0000	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	2011-01-04T00:00:00.000+0000	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
5	2011-01-05T00:00:00.000+0000	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

Command took 0.60 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 10:47:05 PM on test-cluster

Cmd 22

Next, drop the table.

This does not delete the file from which the table was created. Rather, it simply removes the table definition from Spark.

Cmd 23

```
%sql
DROP TABLE BikeSharingDay
```

OK

Command took 0.54 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 10:47:57 PM on test-cluster

Cmd 24

Upload a local file as a table

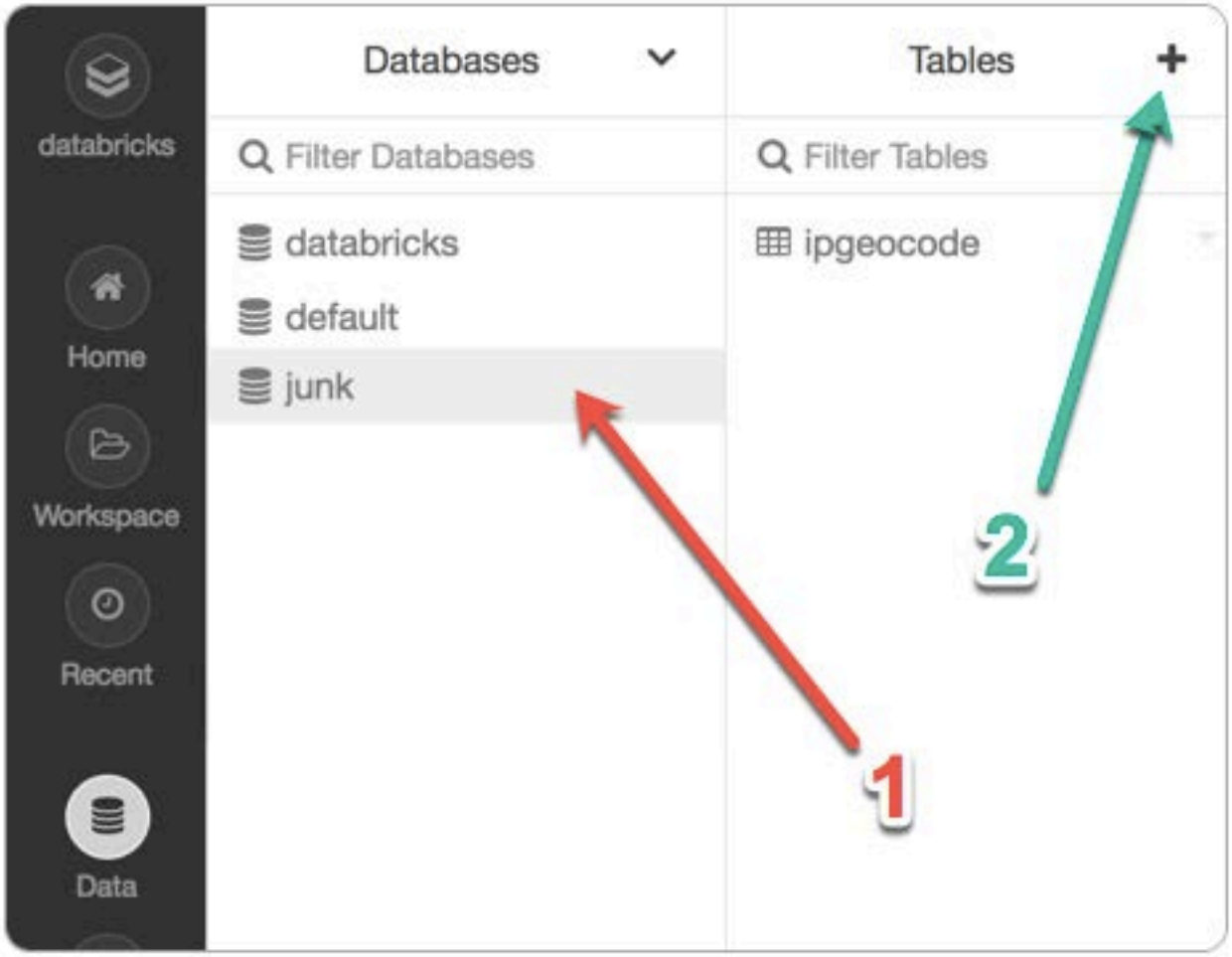
The last two examples use files already loaded on the "server."

Databricks also supports creating tables by uploading files.

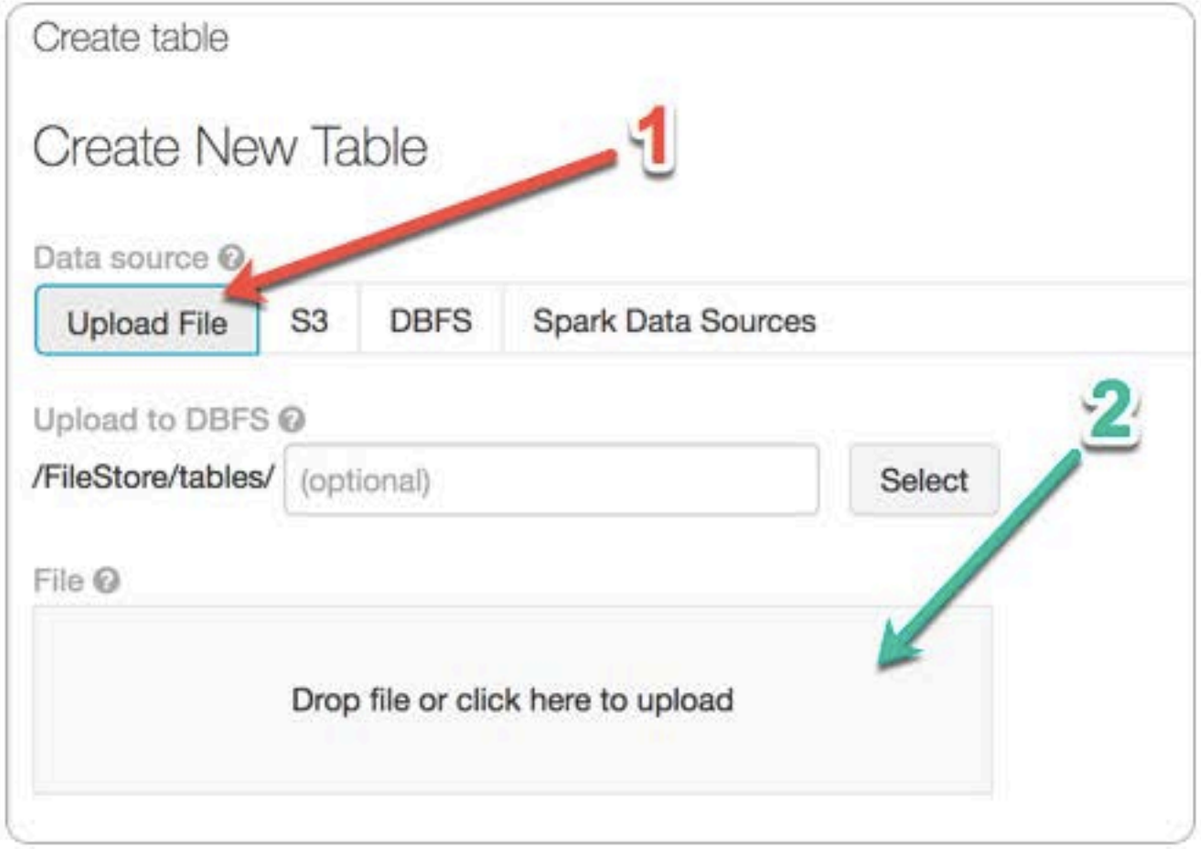
Next, download the following file to your local machine: state-income.csv

Select **Data** from the sidebar, and click on the **junk** database.

This time, select the + icon to create a new table.

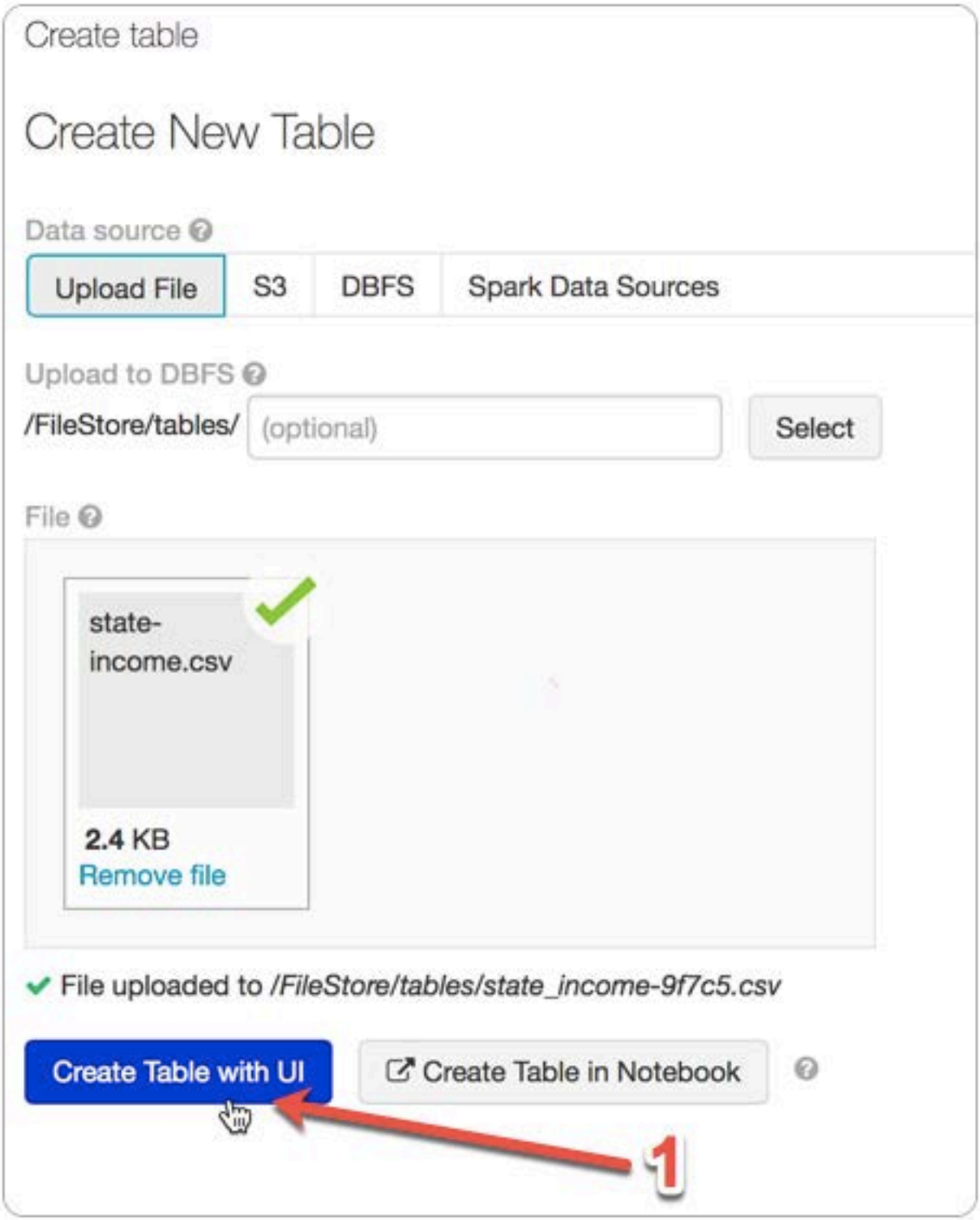


Ensure that **Upload File** is selected. Then, select the **state-income.csv** file from your machine, or drag-and-drop the file to initiate the upload.



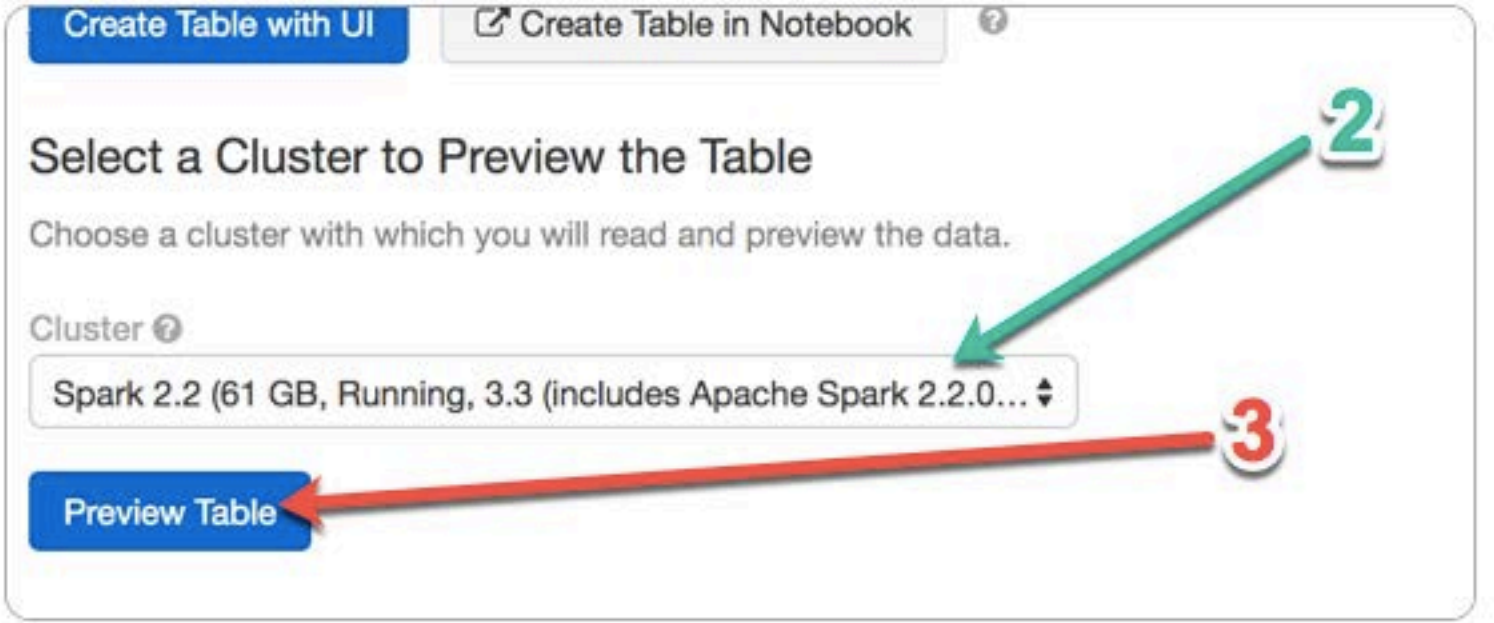
Once the file is uploaded, create the actual table:

1. Click the **Create Table with UI** button



2. In the drop-down dialog, select a cluster

3. Click the **Preview Table** button



4. Another dialog will drop down. Choose the **junk** database
5. Select the **First row is header** checkbox
6. Click the **Create Table** button

Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name

state_income_9f7c5_csv

Create in Database

junk

File Type

CSV

Column Delimiter

,

First row is header

Create Table

Table Preview

Rank	State	2014	2010
STRING	STRING	STRING	STRING
10	Alaska	$60287	$69860
3	California	$67458	$67034
11	Delaware	$57954	$58548
1	Maryland	$70004	$69272
2	New Jersey	$69825	$68342
4	Connecticut	$65753	$66953

4

5

6

Once Databricks finishes processing the file, you'll see another table preview.

⚠️ Databricks tries to choose a table name that doesn't clash with tables created by other users. However, a name clash is still possible. If the table already exists, you'll see an error like the following:

If that happens, just type in a different table name, and try again.

Cmd 29

Next, drop the table to ensure other users don't have a name conflict when uploading their tables.

Cmd 30

```
%sql
DROP TABLE IF EXISTS state_income
```

OK

Command took 0.08 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 10:55:40 PM on test-cluster

Cmd 31

How to mount an S3 bucket to DBFS

Amazon Web Services (AWS) provides cloud file storage in the form of the Amazon Simple Storage Service (S3). Files are stored in "buckets." If you have an S3 account, you can create a bucket, store data files in that bucket, and mount the bucket as a DBFS directory.

Once the bucket is mounted as a DBFS directory, you can access it without exposing your S3 keys.

Take a look at the buckets already mounted to your DBFS:

Cmd 34

```
%fs mounts
```

mountPoint	source	encryptionType
/mnt/training	s3a://databricks-corp-training/common	
/	DatabricksRoot	sse-s3
/databricks-datasets	databricks-datasets	sse-s3
/databricks-results	databricks-results	sse-s3

Command took 1.37 seconds -- by huseyinyilmaz01@gmail.com at 4/2/2020, 11:53:08 PM on test-cluster

Cmd 35

Mount your own bucket to a new mount point. To do so, use the `dbutils.fs.mount(..)` function.

Below, mount a Databricks S3 bucket (using a read-only access and secret key pair), access one of the files in the bucket as a DBFS path, then unmount the bucket.

⚠️ The mount point **must** start with `/mnt/`.

Cmd 36

Create the mount point with `%fs mount`.

⚠️ If the directory was already mounted, you would receive the following error:

> Directory already mounted: /mnt/temp-training

In this case, use a different mount point such as `temp-training-2`, and ensure you update all three references below.

Cmd 37

```
%fs mount s3a://AKIAJBRYNXGHORDHZB4A:a0BzE1bSegfydr3%2FGE3LSPM6uIV5A4hOUfpH8aFF@databricks-corp-training/common /mnt/temp-training
```

▸ (1) Spark Jobs

```
res5: Boolean = true
```

Command took 25.51 seconds -- by huseyinyilmaz01@gmail.com at 4/3/2020, 12:05:10 AM on test-cluster

List the contents of the directory you just mounted:

Cmd 39

```
%fs ls /mnt/temp-training
```

path	name	size
dbfs:/mnt/temp-training/301/	301/	0
dbfs:/mnt/temp-training/Chicago-Crimes-2018.csv	Chicago-Crimes-2018.csv	5201668
dbfs:/mnt/temp-training/City-Data.delta/	City-Data.delta/	0
dbfs:/mnt/temp-training/City-Data.parquet/	City-Data.parquet/	0
dbfs:/mnt/temp-training/EDGAR-Log-20170329/	EDGAR-Log-20170329/	0
dbfs:/mnt/temp-training/StatLib/	StatLib/	0
dbfs:/mnt/temp-training/UbiqLog4UCI/	UbiqLog4UCI/	0
dbfs:/mnt/temp-training/_META/	_META/	0
dbfs:/mnt/temp-training/adventure-works/	adventure-works/	0

Command took 0.63 seconds -- by huseyinyilmaz01@gmail.com at 4/3/2020, 12:06:09 AM on test-cluster

Take a peek at the head of the file `auto-mpg.csv` :

Cmd 41

```
%fs head /mnt/temp-training/auto-mpg.csv
```

```
18,8,307,130,3504,12,70,1,chevrolet chevelle malibu
15,8,350,165,3693,11.5,70,1,buick skylark 320
18,8,318,150,3436,11,70,1,plymouth satellite
16,8,304,150,3433,12,70,1,amc rebel sst
17,8,302,140,3449,10.5,70,1,ford torino
15,8,429,198,4341,10,70,1,ford galaxie 500
14,8,454,220,4354,9,70,1,chevrolet impala
14,8,440,215,4312,8.5,70,1,plymouth fury iii
14,8,455,225,4425,10,70,1,pontiac catalina
15,8,390,190,3850,8.5,70,1,amc ambassador dpl
15,8,383,170,3563,10,70,1,dodge challenger se
14,8,340,160,3609,8,70,1,plymouth 'cuda 340
15,8,400,150,3761,9.5,70,1,chevrolet monte carlo
14,8,455,225,3086,10,70,1,buick estate wagon (sw)
24,4,113,95,2372,15,70,3,toyota corona mark ii
22,6,198,95,2833,15.5,70,1,plymouth duster
18,6,199,97,2774,15.5,70,1,amc hornet
21,6,200,85,2587,16,70,1,ford maverick
27,4,97,88,2130,14.5,70,3,datsun pl510
26,4,97,46,1835,20.5,70,2,volkswagen 1131 deluxe sedan
25,4,110,87,2672,17.5,70,2,peugeot 504
```

Now you are done, unmount the directory.

Cmd 43

```
# %fs unmount /mnt/temp-training
```

Command took 0.02 seconds -- by huseyinyilmaz01@gmail.com at 4/3/2020, 12:06:37 AM on test-cluster

Cmd 44

Create your own access keys in AWS for your own S3 buckets and mount them the same way.

This allows access to your S3 data directly from Databricks distributed file system (DBFS).

Once mounted, you can delete the single cell that contained your keys or even the entire notebook protecting your keys from unscrupulous actors.

Cmd 45

Summary

Databricks allows you to:

- Create tables from existing data
- Create tables from uploaded files
- Mount your own S3 buckets

Review Questions

Q: How can you see which tables have been created?
A: Go to the **Data** section using the button-bar to the left.

Q: What is Amazon S3?
A: Amazon S3 stands for Simple Storage Service. It provides cloud-optimized storage of large data files that easily scales with your storage needs.

Q: What is DBFS?
A: DBFS stands for Databricks File System. DBFS provides for the cloud what the Hadoop File System (HDFS) provides for local spark deployments. DBFS uses Amazon S3 and makes it easy to access files by name.

Q: Which is more efficient to query, a parquet file or a CSV file?
A: Parquet files are highly optimized binary formats for storing tables. The overhead is less than required to parse a CSV file. Parquet is the big data analogue to CSV as it is optimized, distributed, and more fault tolerant than CSV files.

Q: How can you create a new table?
A: Create new tables by either:

- Uploading a new file using the Data tab on the left.
- Mounting an existing file from DBFS.

Q: What is the SQL syntax for defining a table in Spark from an existing parquet file in DBFS?
A: `CREATE TABLE IF NOT EXISTS IPGeocode USING parquet OPTIONS ( path "dbfs:/mnt/training/ip-geocode.parquet" )`