Databricks DBFS
Databricks File System

Source : databricks.com

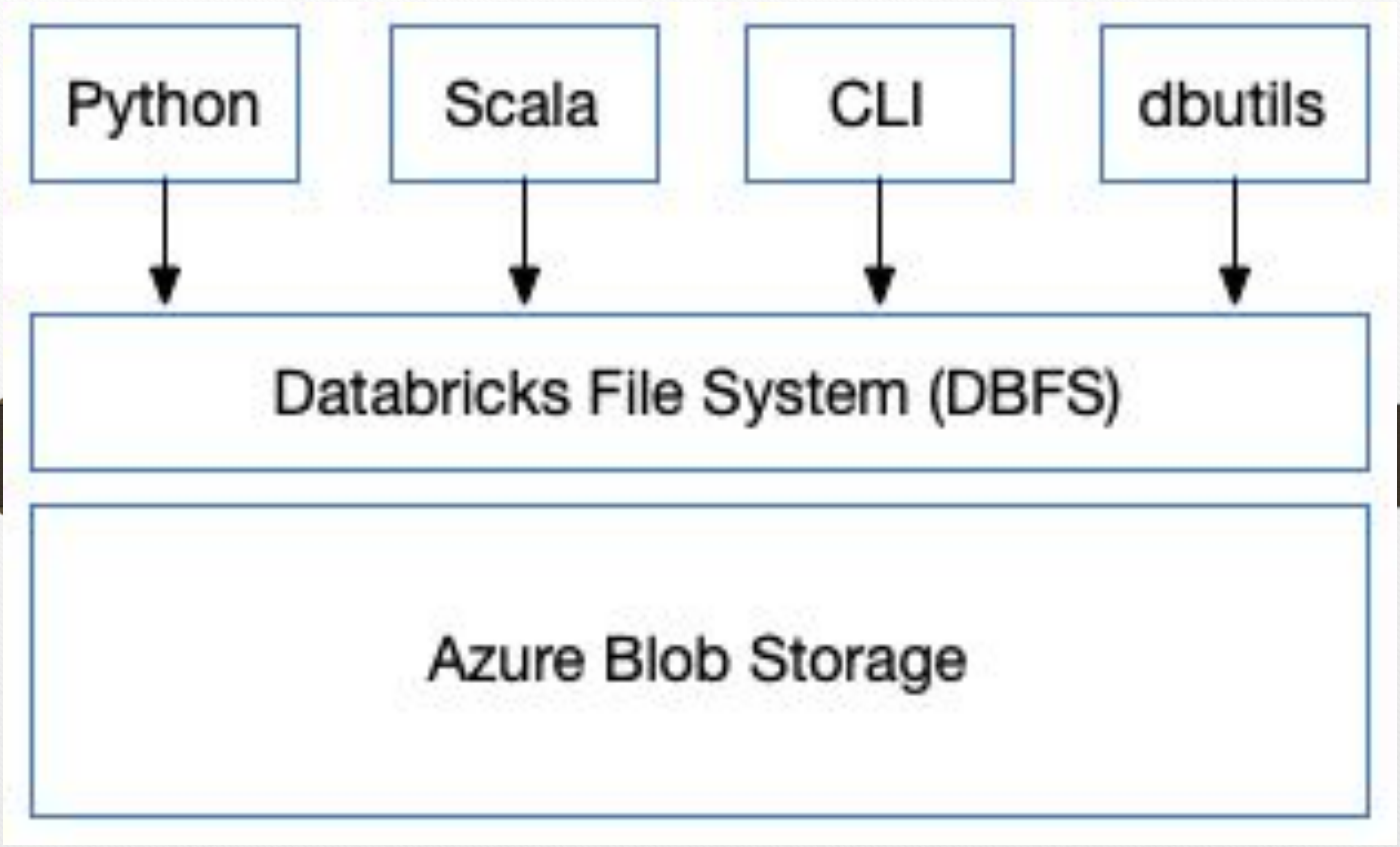
Databricks File System (DBFS)

Databricks File System (DBFS) is a distributed file system mounted into a Databricks workspace and available on Databricks clusters. DBFS is an abstraction on top of scalable object storage and offers the following benefits

- Allows you to mount storage objects so that you can seamlessly access data without requiring credentials.

- Allows you to interact with object storage using directory and file semantics instead of storage URLs.

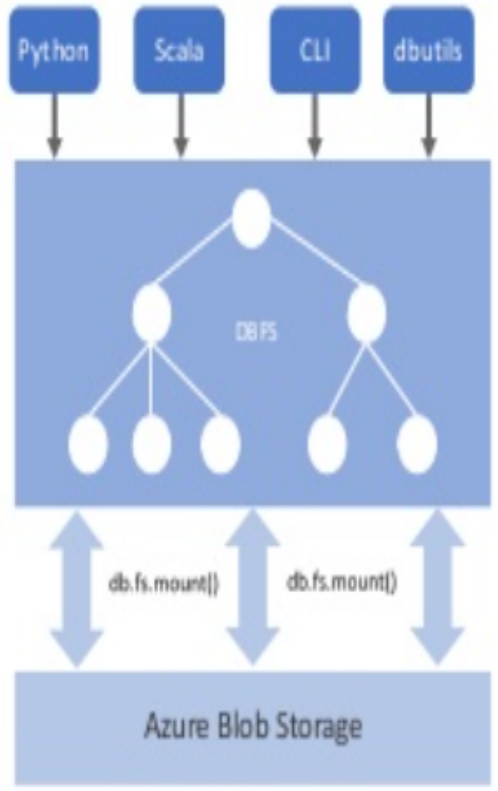
- Persists files to object storage, so you won't lose data after you terminate a cluster.

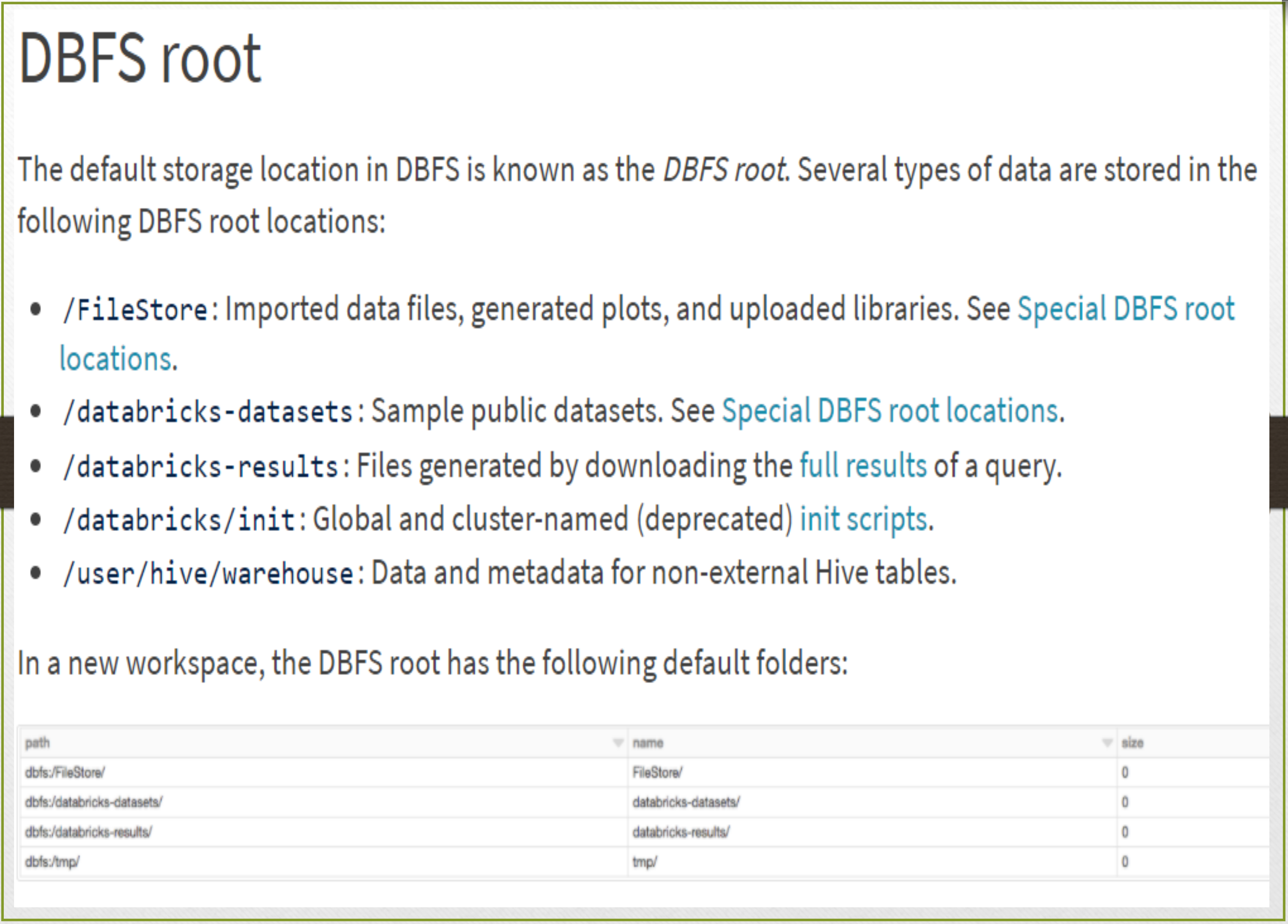

Python
Scala
CLI
dbutils
Databricks File System (DBFS)
Azure Blob Storage

DATABRICKS FILE SYSTEM (DBFS)

Is a distributed File System (DBFS) that is a layer over Azure Blob Storage

- Azure Storage buckets can be mounted in DBFS so that users can directly access them without specifying the storage keys
- DBFS mounts are created using *dbutils.fs.mount()*
- Azure Storage data can be cached locally on the SSD of the worker nodes
- Available in both Python and Scala and accessible via a DBFS CLI
- Data persist in Azure Blob Storage – is not lost even after cluster termination
- Comes pre-installed on Spark clusters in Databricks





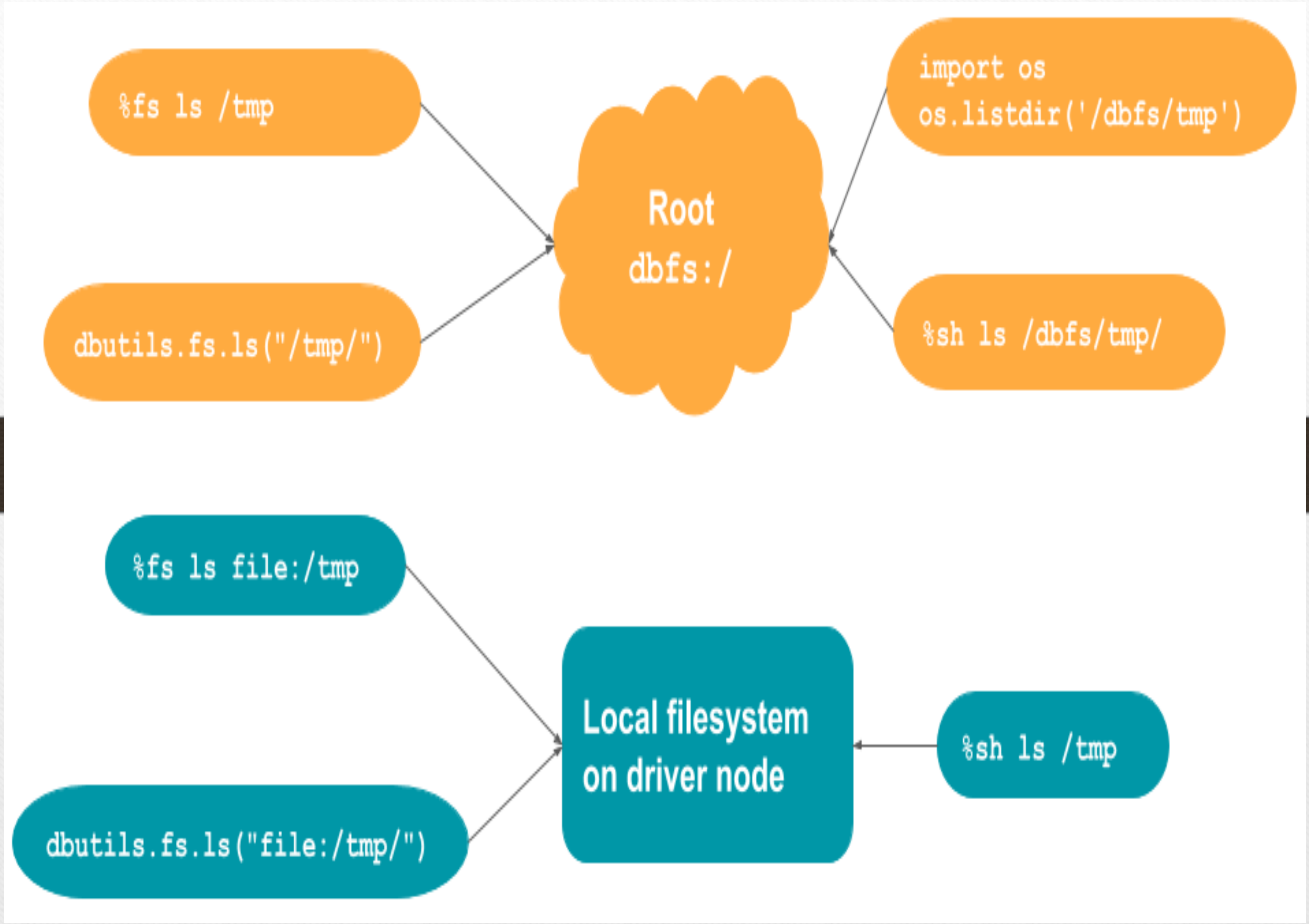
DBFS root

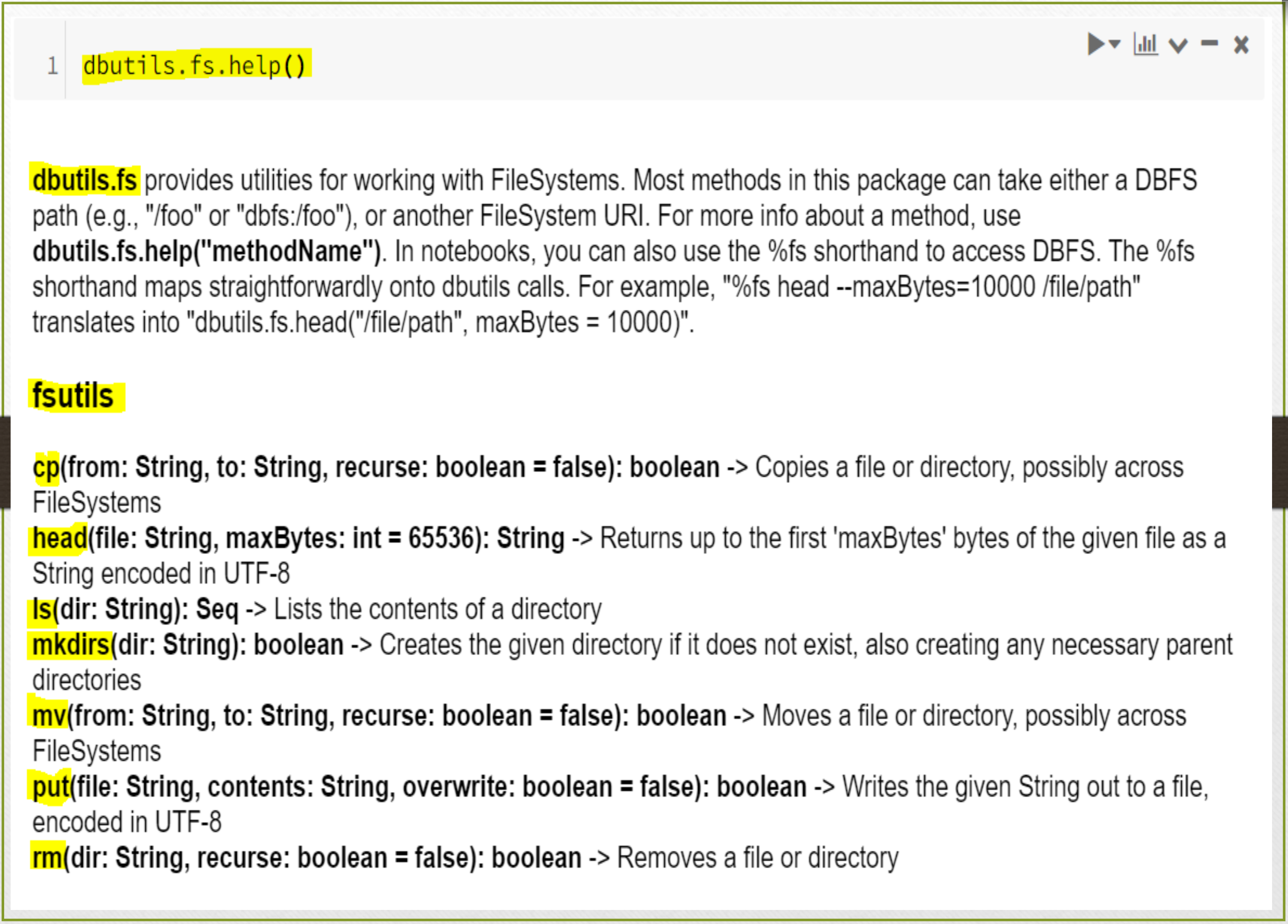
The default storage location in DBFS is known as the *DBFS root*. Several types of data are stored in the following DBFS root locations:

- `/FileStore`: Imported data files, generated plots, and uploaded libraries. See Special DBFS root locations.
- `/databricks-datasets`: Sample public datasets. See Special DBFS root locations.
- `/databricks-results`: Files generated by downloading the full results of a query.
- `/databricks/init`: Global and cluster-named (deprecated) init scripts.
- `/user/hive/warehouse`: Data and metadata for non-external Hive tables.

In a new workspace, the DBFS root has the following default folders:

path	name	size
dbfs:/FileStore/	FileStore/	0
dbfs:/databricks-datasets/	databricks-datasets/	0
dbfs:/databricks-results/	databricks-results/	0
dbfs:/tmp/	tmp/	0


%fs ls /tmp
dbutils.fs.ls("/tmp/")
Root
dbfs:/
import os
os.listdir('/dbfs/tmp')
%sh ls /dbfs/tmp/
%fs ls file:/tmp
dbutils.fs.ls("file:/tmp/")
Local filesystem on driver node
%sh ls /tmp

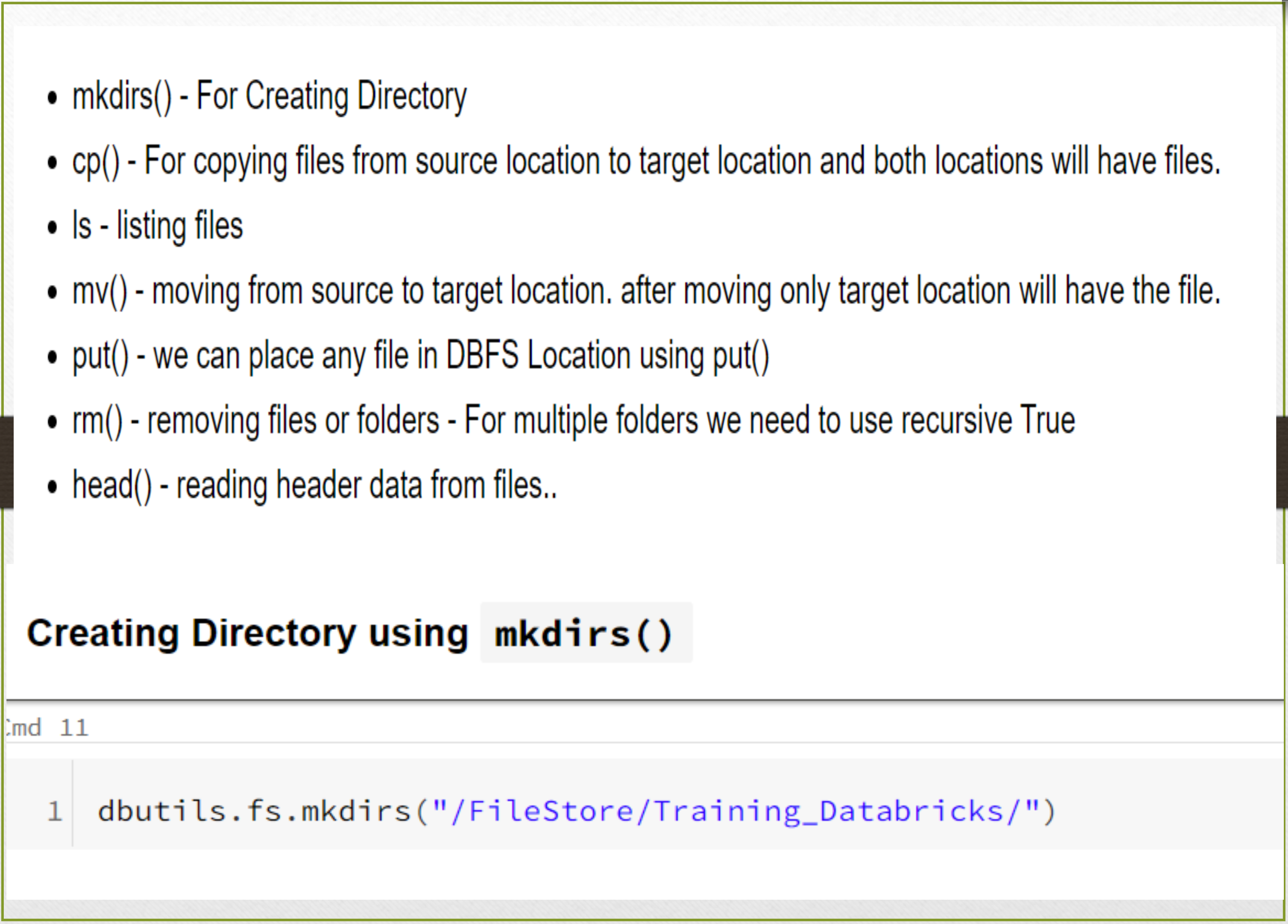


```
dbutils.fs.help()
```

dbutils.fs provides utilities for working with FileSystems. Most methods in this package can take either a DBFS path (e.g., "/foo" or "dbfs:/foo"), or another FileSystem URI. For more info about a method, use **dbutils.fs.help("methodName")**. In notebooks, you can also use the %fs shorthand to access DBFS. The %fs shorthand maps straightforwardly onto dbutils calls. For example, "%fs head --maxBytes=10000 /file/path" translates into "dbutils.fs.head("/file/path", maxBytes = 10000)".

fsutils

cp(from: String, to: String, recurse: boolean = false): boolean -> Copies a file or directory, possibly across FileSystems
head(file: String, maxBytes: int = 65536): String -> Returns up to the first 'maxBytes' bytes of the given file as a String encoded in UTF-8
ls(dir: String): Seq -> Lists the contents of a directory
mkdirs(dir: String): boolean -> Creates the given directory if it does not exist, also creating any necessary parent directories
mv(from: String, to: String, recurse: boolean = false): boolean -> Moves a file or directory, possibly across FileSystems
put(file: String, contents: String, overwrite: boolean = false): boolean -> Writes the given String out to a file, encoded in UTF-8
rm(dir: String, recurse: boolean = false): boolean -> Removes a file or directory

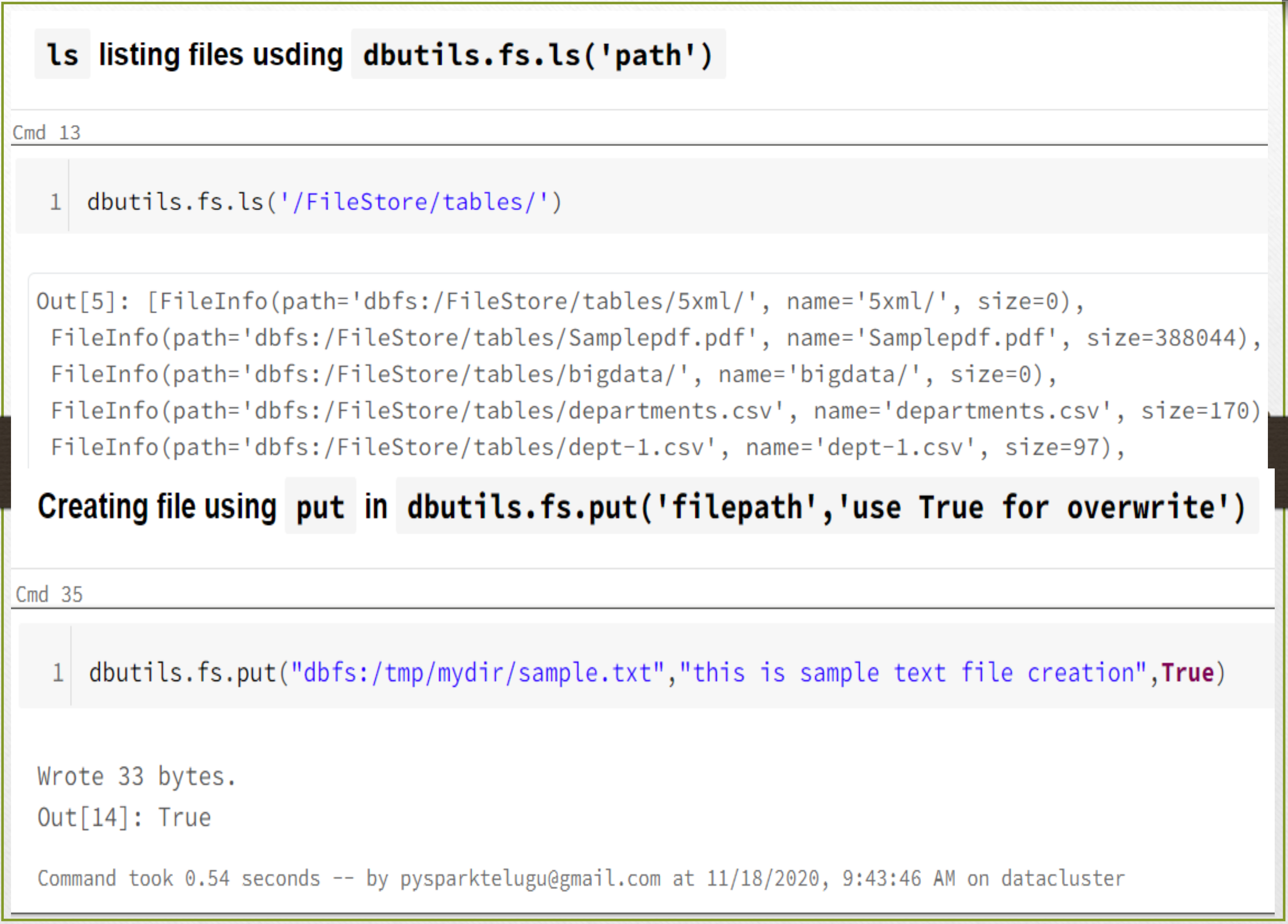


- mkdirs() - For Creating Directory
- cp() - For copying files from source location to target location and both locations will have files.
- ls - listing files
- mv() - moving from source to target location. after moving only target location will have the file.
- put() - we can place any file in DBFS Location using put()
- rm() - removing files or folders - For multiple folders we need to use recursive True
- head() - reading header data from files..

Creating Directory using `mkdirs()`

md 11

```
dbutils.fs.mkdirs("/FileStore/Training_Databricks/")
```



`ls` listing files usding `dbutils.fs.ls('path')`

Cmd 13

```
dbutils.fs.ls('/FileStore/tables/')
```

```
Out[5]: [FileInfo(path='dbfs:/FileStore/tables/5xml/', name='5xml/', size=0),
 FileInfo(path='dbfs:/FileStore/tables/Samplepdf.pdf', name='Samplepdf.pdf', size=388044),
 FileInfo(path='dbfs:/FileStore/tables/bigdata/', name='bigdata/', size=0),
 FileInfo(path='dbfs:/FileStore/tables/departments.csv', name='departments.csv', size=170)
 FileInfo(path='dbfs:/FileStore/tables/dept-1.csv', name='dept-1.csv', size=97),
```

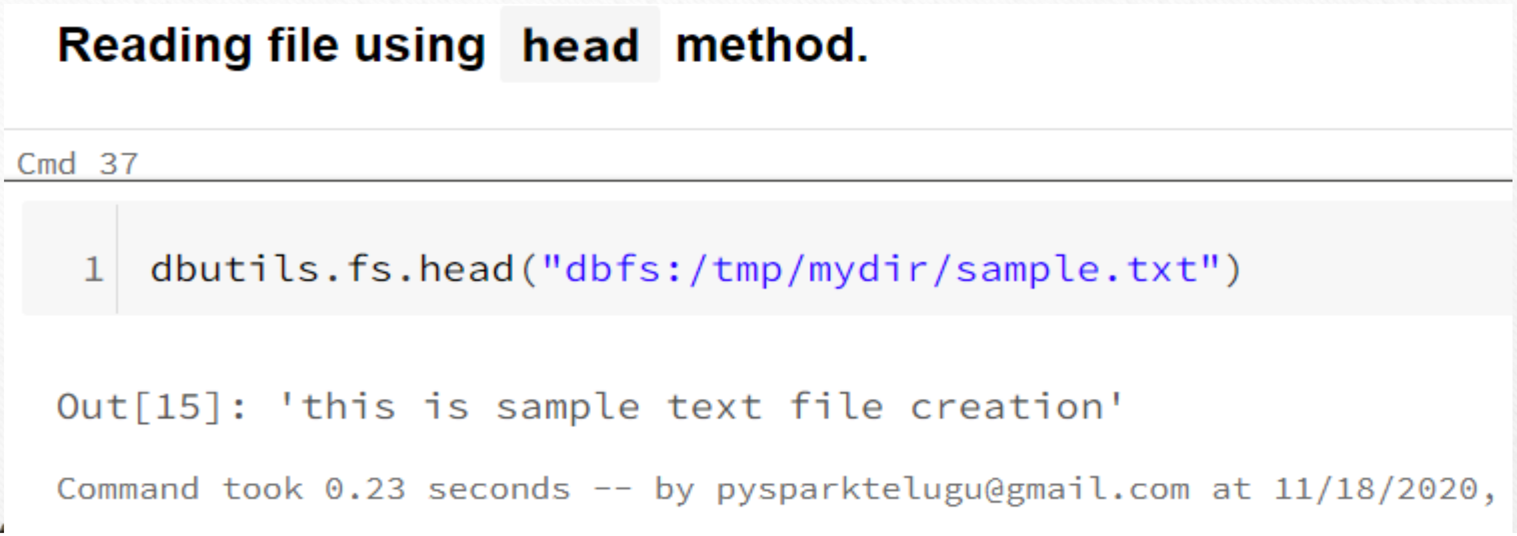
Creating file using `put` in `dbutils.fs.put('filepath','use True for overwrite')`

Cmd 35

```
dbutils.fs.put("dbfs:/tmp/mydir/sample.txt","this is sample text file creation",True)
```

```
Wrote 33 bytes.
Out[14]: True
```

Command took 0.54 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 9:43:46 AM on datacluster

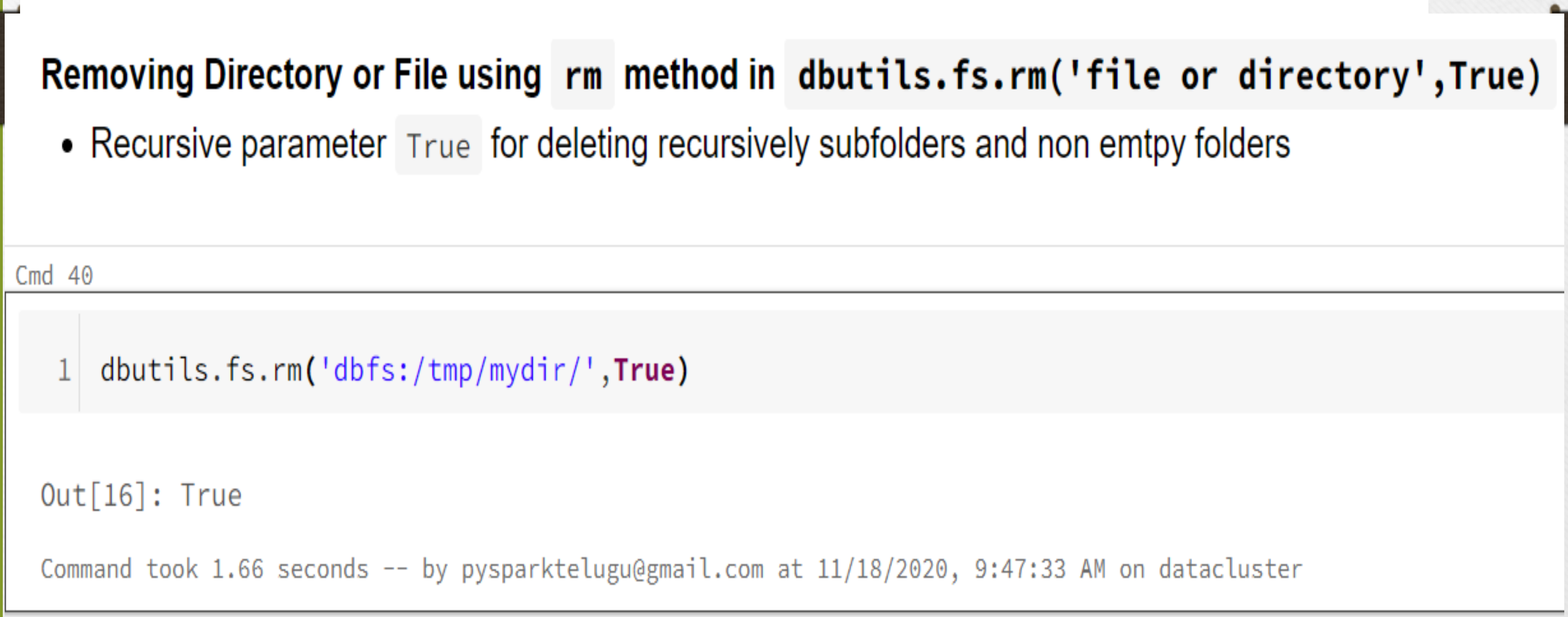


Reading file using `head` method.

Cmd 37

```
dbutils.fs.head("dbfs:/tmp/mydir/sample.txt")
```

```
Out[15]: 'this is sample text file creation'

Command took 0.23 seconds -- by pysparktelugu@gmail.com at 11/18/2020,
```



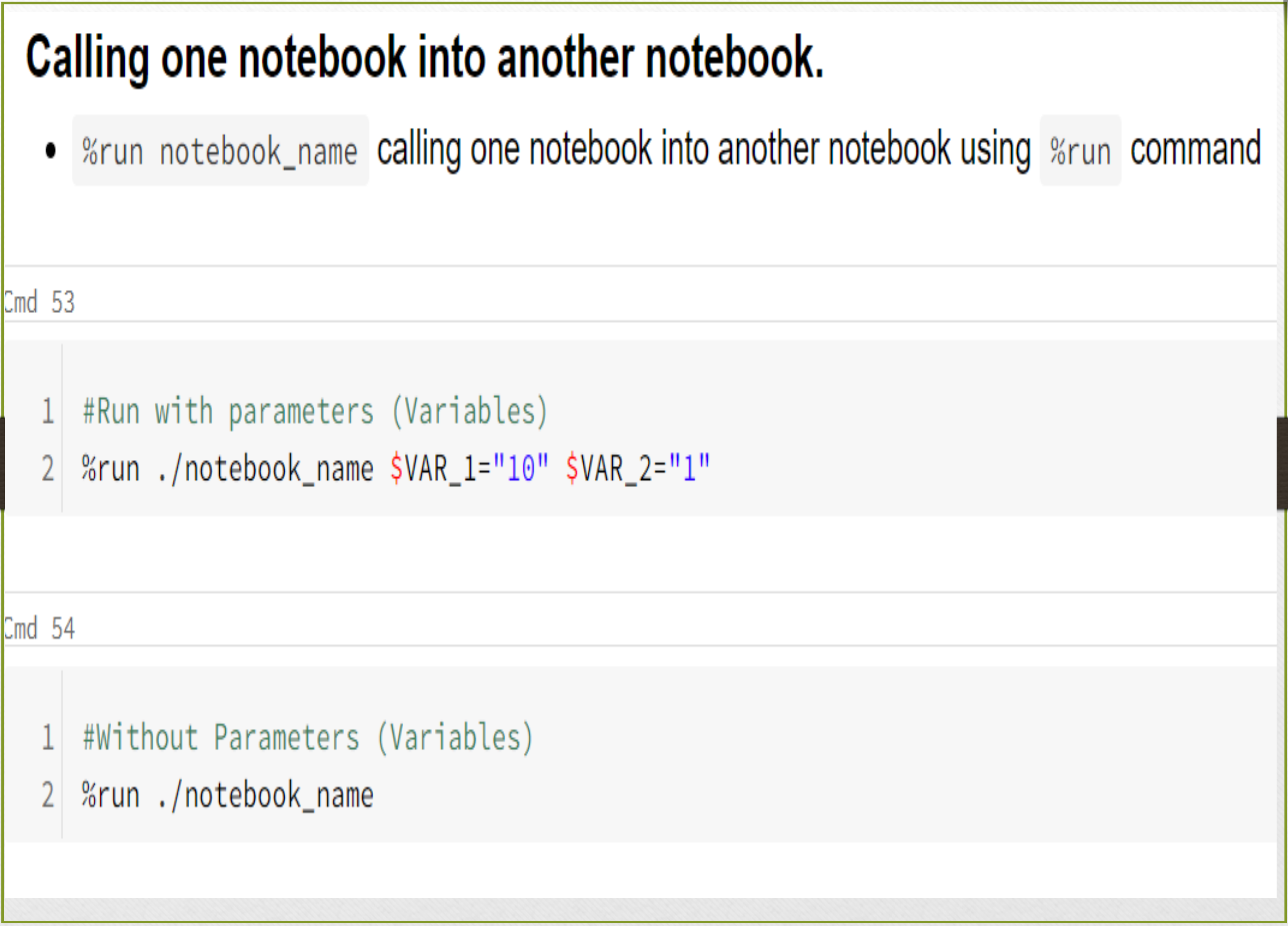
Removing Directory or File using `rm` method in `dbutils.fs.rm('file or directory',True)`

- Recursive parameter `True` for deleting recursively subfolders and non emtpy folders

Cmd 40

```
dbutils.fs.rm('dbfs:/tmp/mydir/',True)
```

```
Out[16]: True

Command took 1.66 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 9:47:33 AM on datacluster
```



Calling one notebook into another notebook.

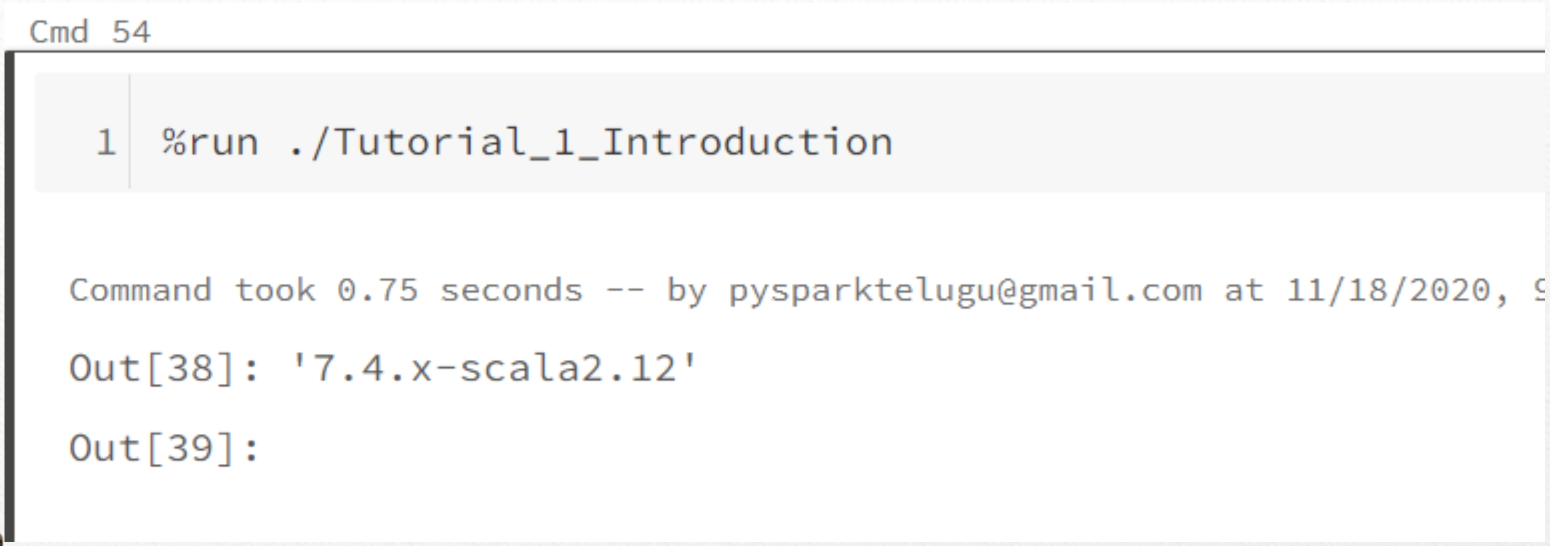
- `%run notebook_name` calling one notebook into another notebook using `%run` command

Cmd 53

```
#Run with parameters (Variables)
%run ./notebook_name $VAR_1="10" $VAR_2="1"
```

Cmd 54

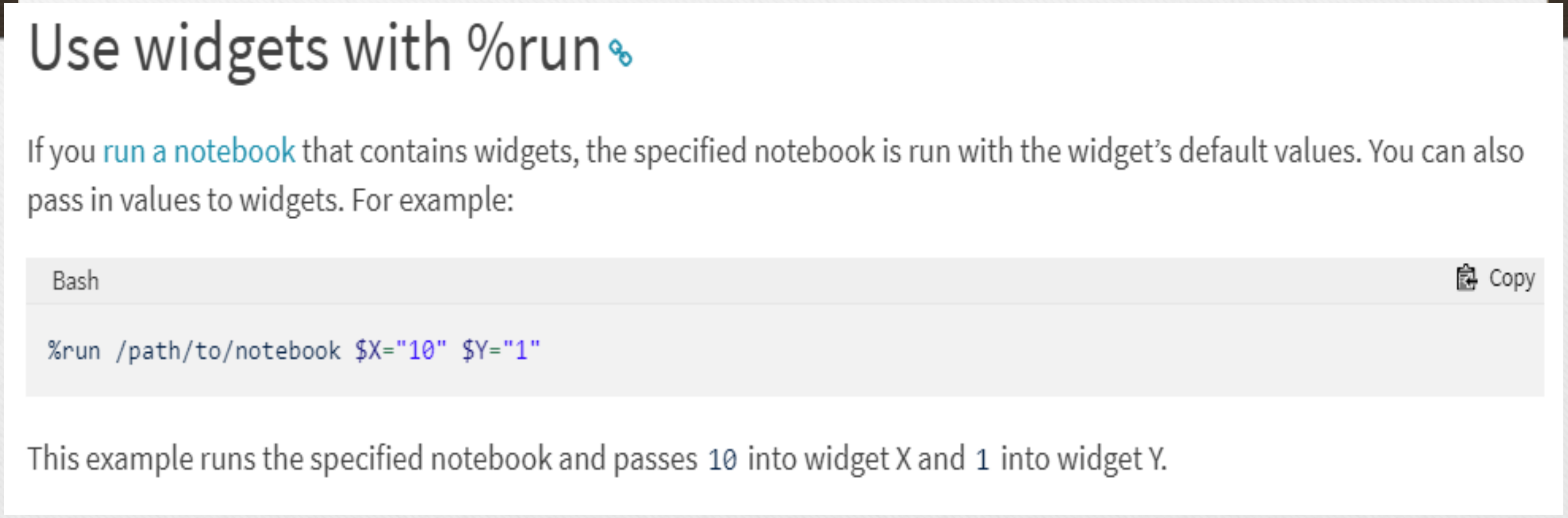
```
#Without Parameters (Variables)
%run ./notebook_name
```



Cmd 54

```
%run ./Tutorial_1_Introduction
```

```
Command took 0.75 seconds -- by pysparktelugu@gmail.com at 11/18/2020,

Out[38]: '7.4.x-scala2.12'

Out[39]:
```

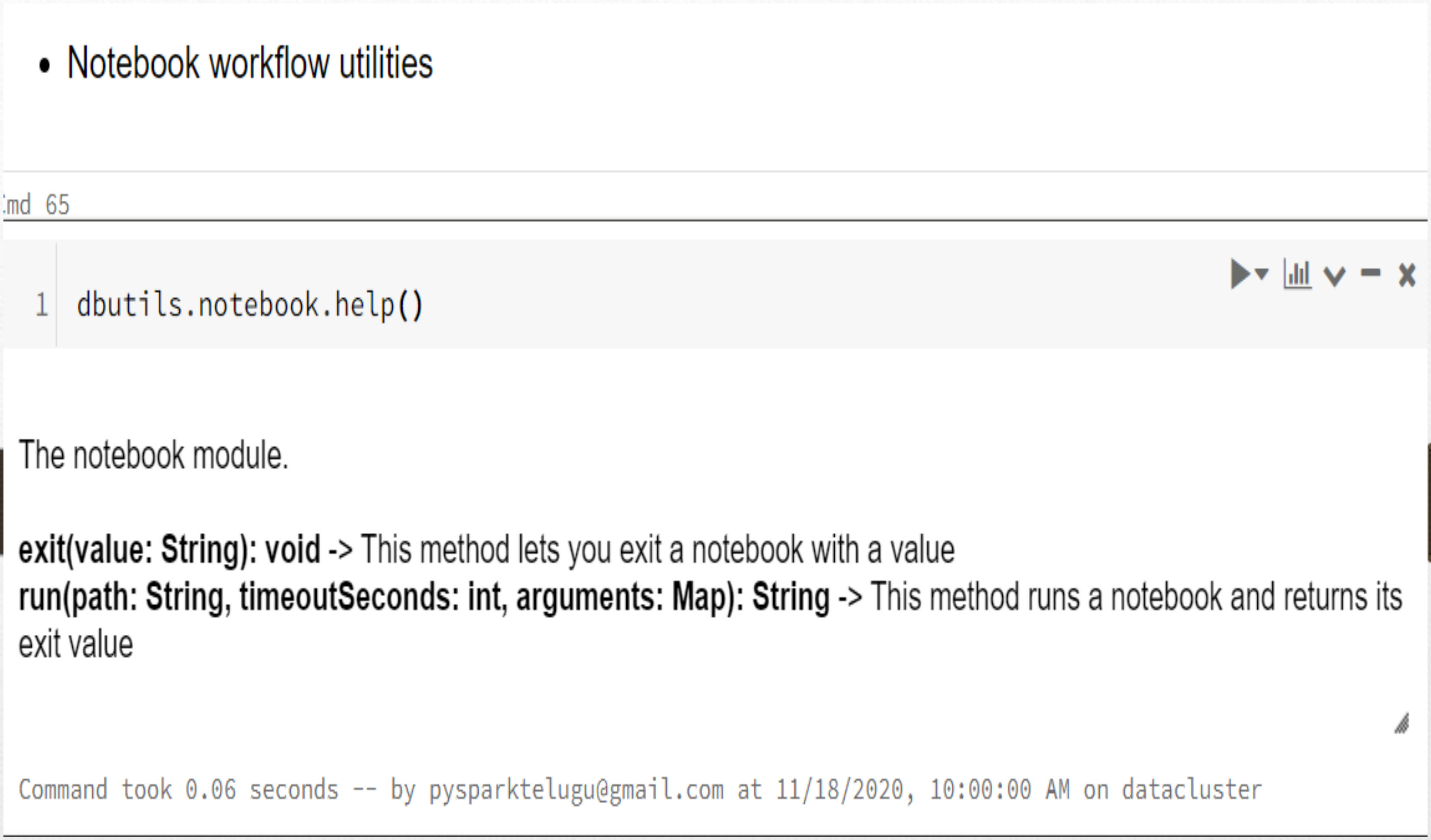


Use widgets with %run

If you run a notebook that contains widgets, the specified notebook is run with the widget's default values. You can also pass in values to widgets. For example:

```bash
%run /path/to/notebook $X="10" $Y="1"
```

This example runs the specified notebook and passes `10` into widget X and `1` into widget Y.


- Notebook workflow utilities

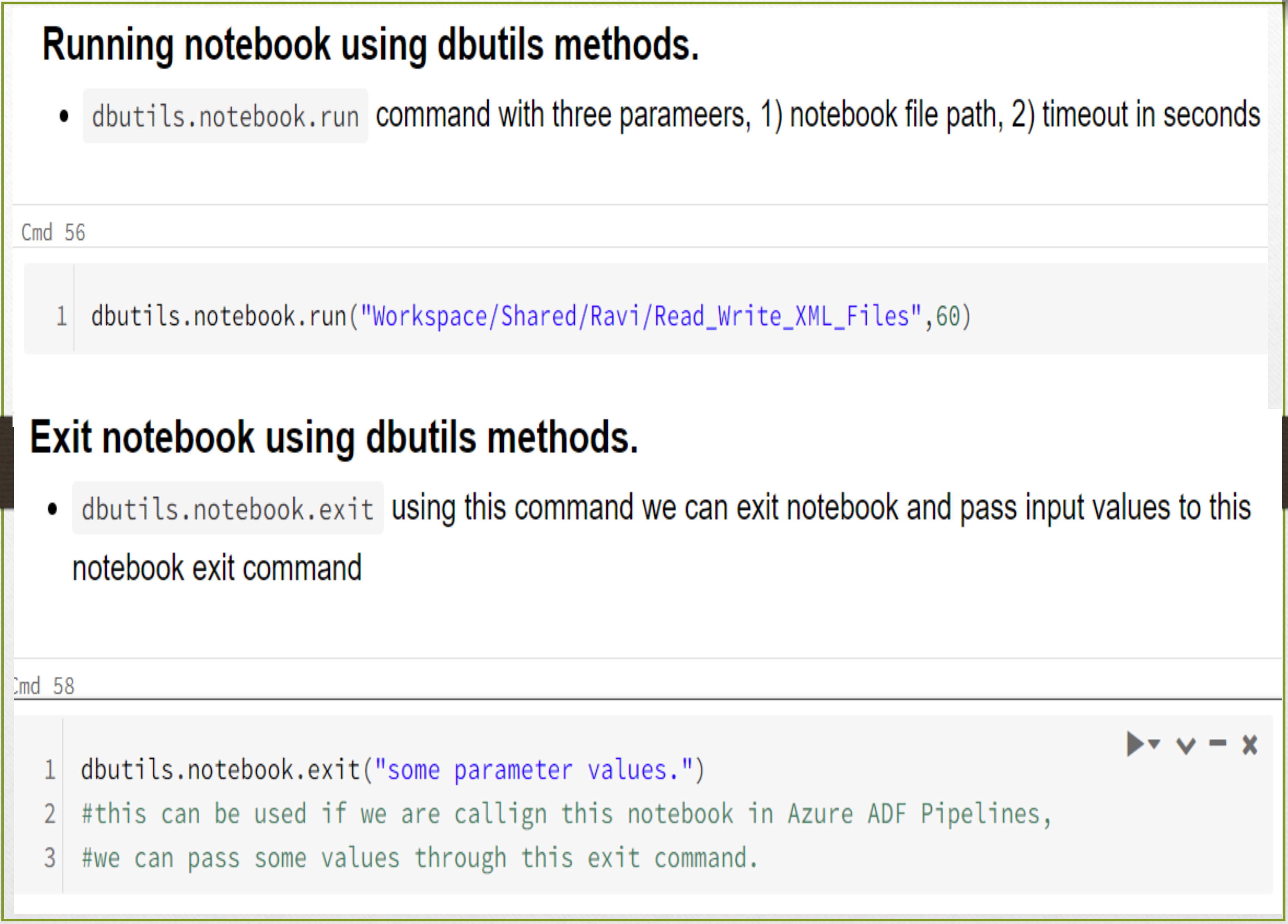
Cmd 65

```
dbutils.notebook.help()
```

The notebook module.

exit(value: String): void -> This method lets you exit a notebook with a value
run(path: String, timeoutSeconds: int, arguments: Map): String -> This method runs a notebook and returns its exit value

```
Command took 0.06 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 10:00:00 AM on datacluster
```



Running notebook using dbutils methods.

- `dbutils.notebook.run` command with three parameers, 1) notebook file path, 2) timeout in seconds

Cmd 56

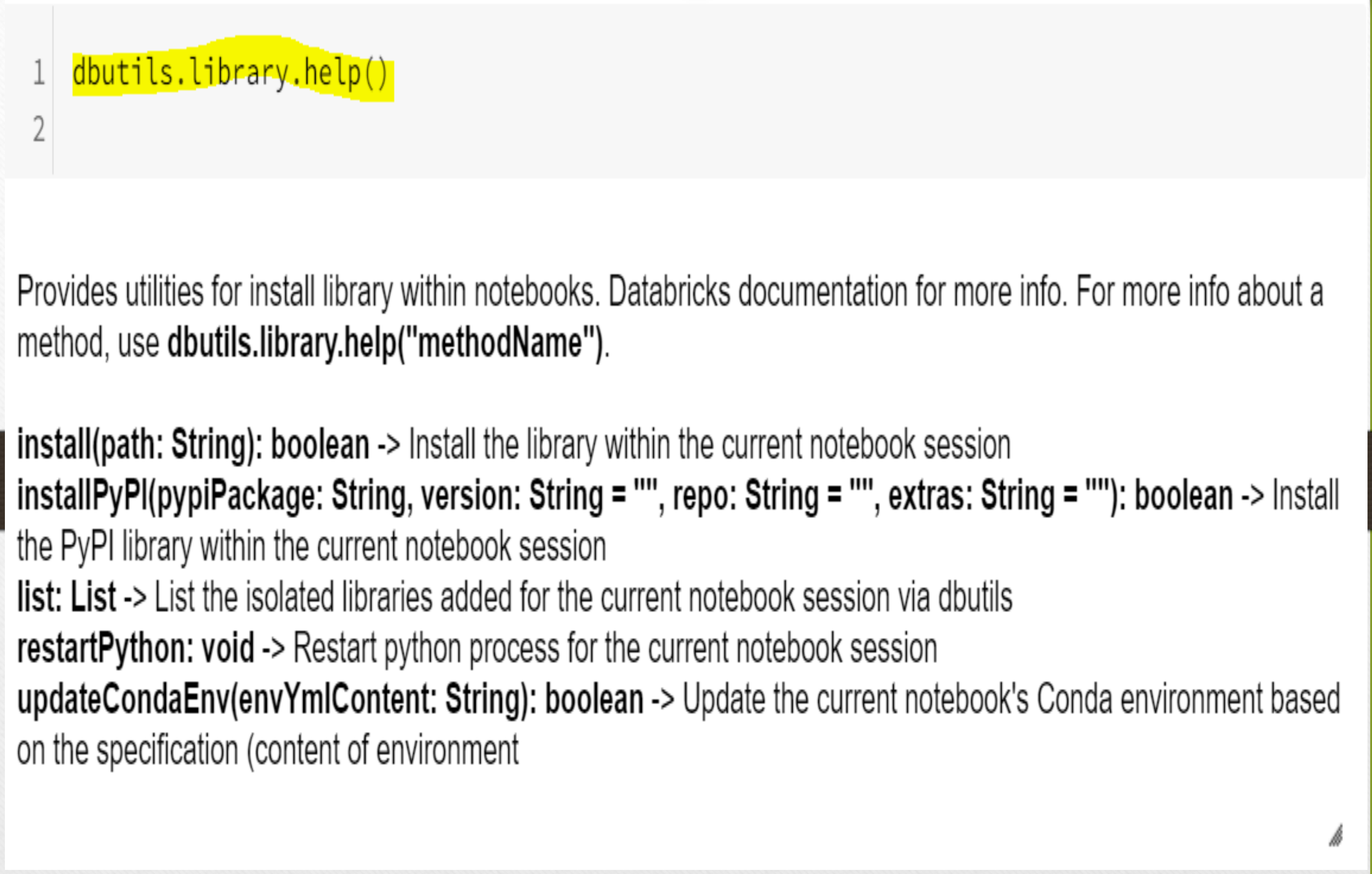
```
dbutils.notebook.run("Workspace/Shared/Ravi/Read_Write_XML_Files",60)
```

Exit notebook using dbutils methods.

- `dbutils.notebook.exit` using this command we can exit notebook and pass input values to this notebook exit command

Cmd 58

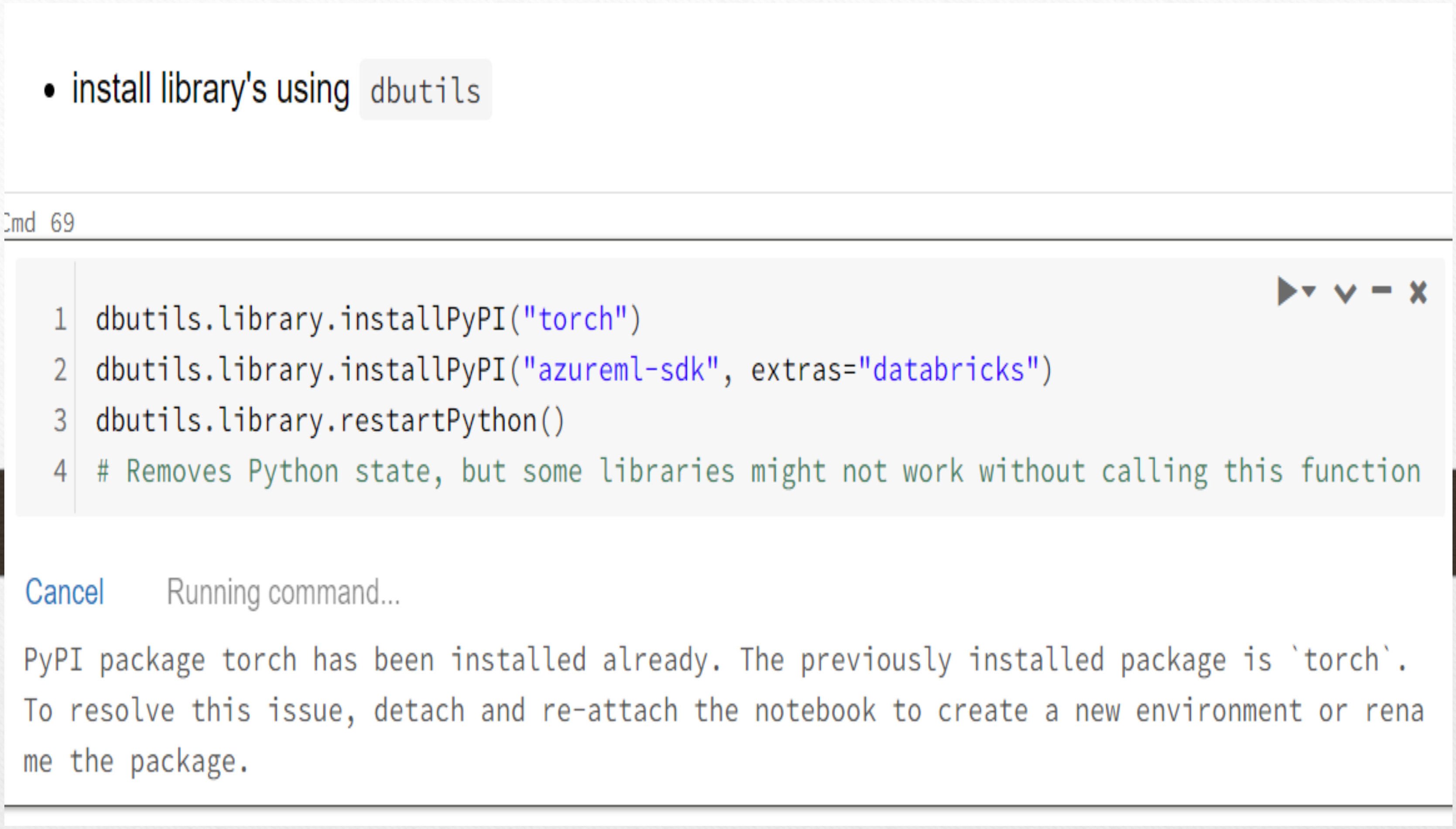
```
dbutils.notebook.exit("some parameter values.")
#this can be used if we are callign this notebook in Azure ADF Pipelines,
#we can pass some values through this exit command.
```



```
dbutils.library.help()
```

Provides utilities for install library within notebooks. Databricks documentation for more info. For more info about a method, use **dbutils.library.help("methodName")**.

install(path: String): boolean -> Install the library within the current notebook session
installPyPI(pypiPackage: String, version: String = "", repo: String = "", extras: String = ""): boolean -> Install the PyPI library within the current notebook session
list: List -> List the isolated libraries added for the current notebook session via dbutils
restartPython: void -> Restart python process for the current notebook session
updateCondaEnv(envYmlContent: String): boolean -> Update the current notebook's Conda environment based on the specification (content of environment



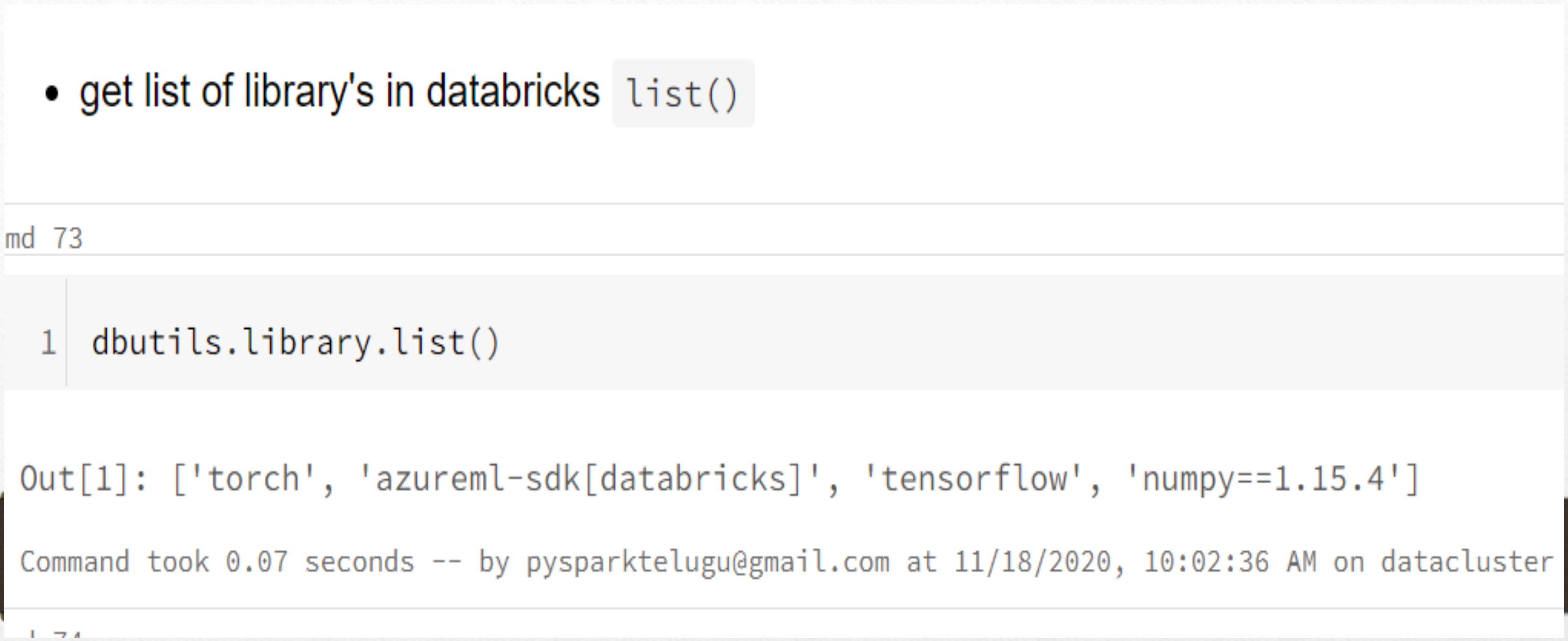
- install library's using `dbutils`

Cmd 69

```python
dbutils.library.installPyPI("torch")
dbutils.library.installPyPI("azureml-sdk", extras="databricks")
dbutils.library.restartPython()
# Removes Python state, but some libraries might not work without calling this function
```

Cancel Running command...

```
PyPI package torch has been installed already. The previously installed package is `torch`.
To resolve this issue, detach and re-attach the notebook to create a new environment or rena
me the package.
```



- get list of library's in databricks `list()`

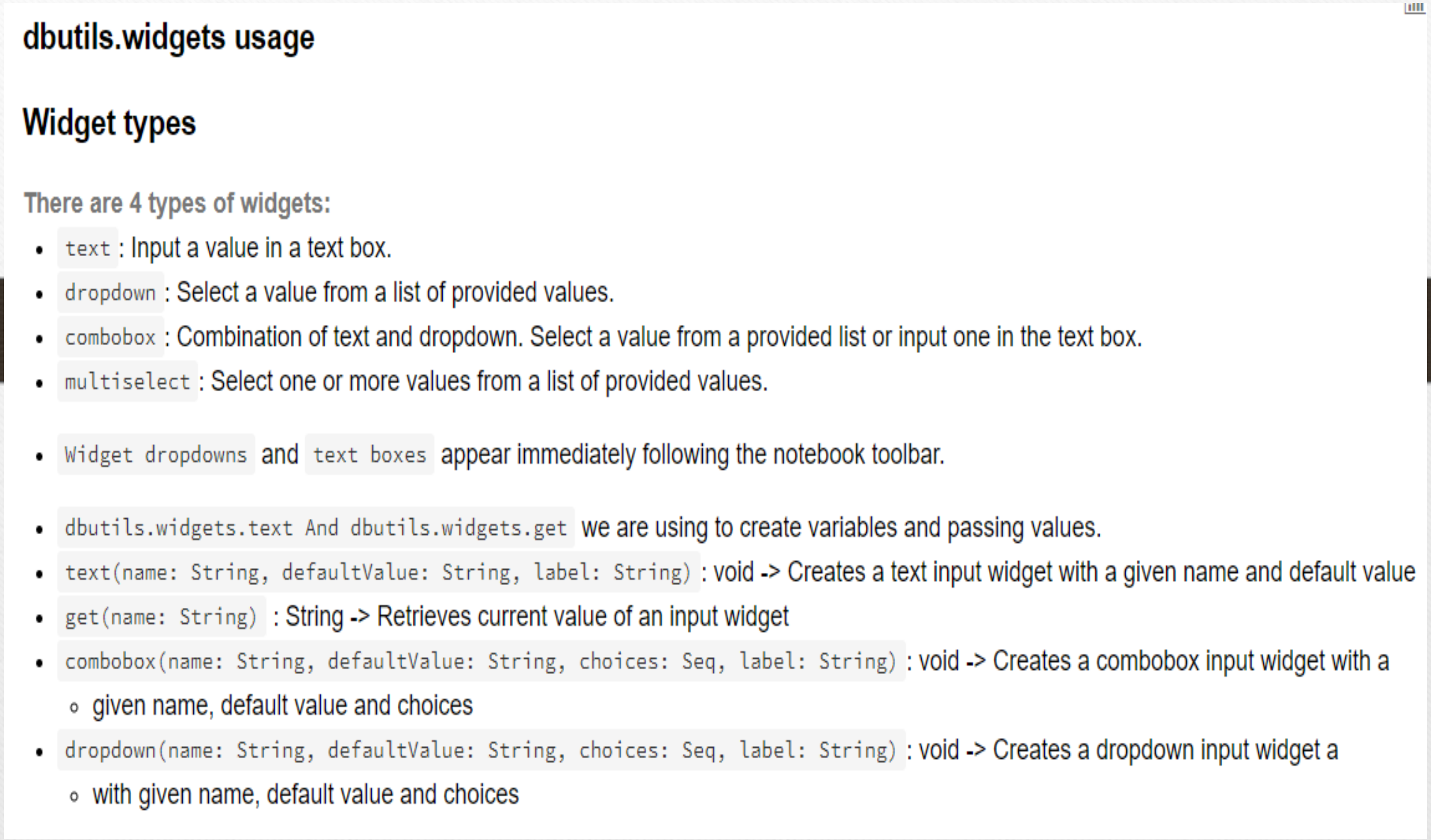
md 73

```
dbutils.library.list()
```

```
Out[1]: ['torch', 'azureml-sdk[databricks]', 'tensorflow', 'numpy==1.15.4']

Command took 0.07 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 10:02:36 AM on datacluster
```

Dbutils.widgets

Input widgets allow you to add parameters to your notebooks and dashboards. The widget API consists of calls to create various types of input widgets, remove them, and get bound values



dbutils.widgets usage

Widget types

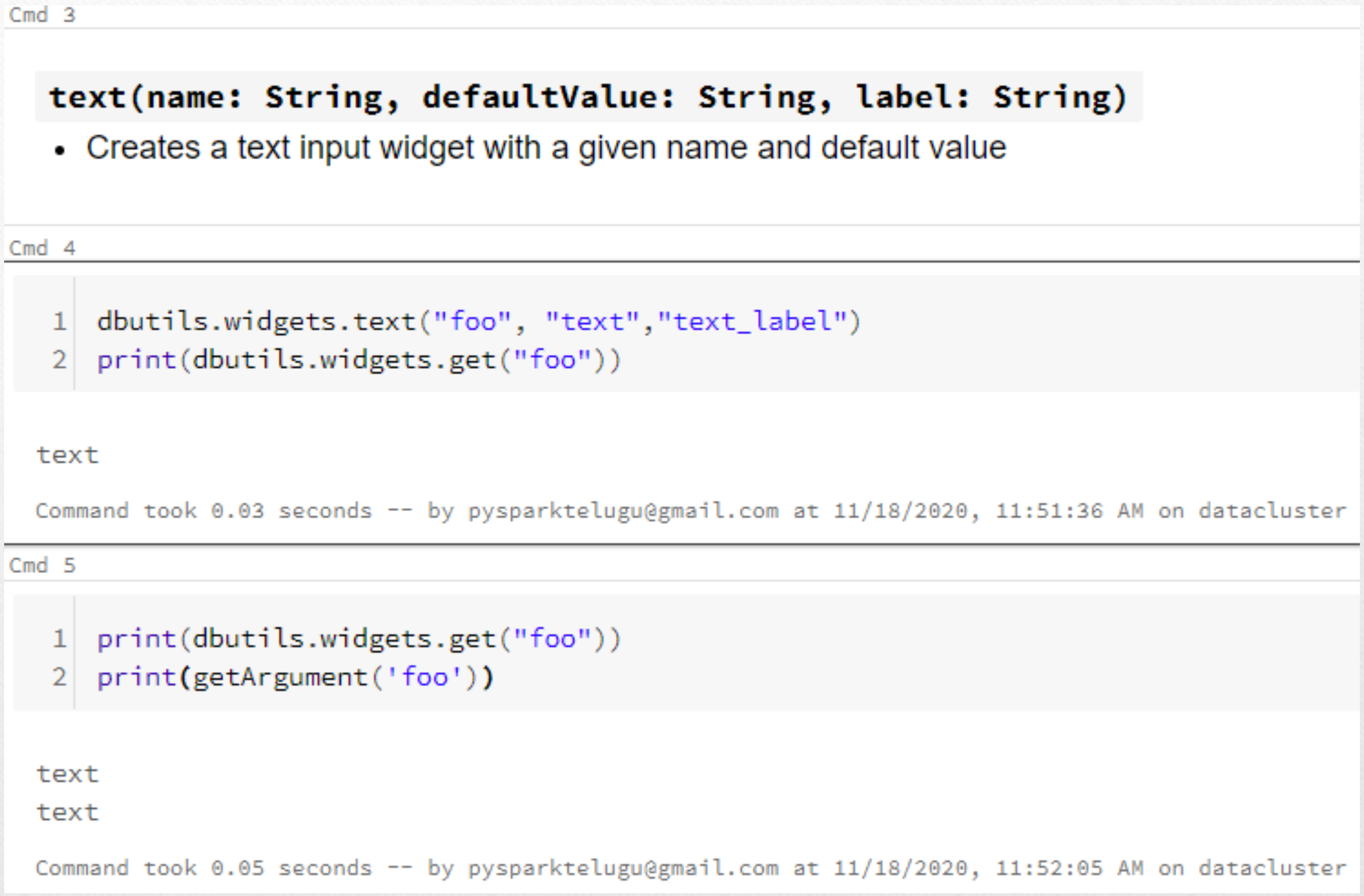
There are 4 types of widgets:

- `text` : Input a value in a text box.
- `dropdown` : Select a value from a list of provided values.
- `combobox` : Combination of text and dropdown. Select a value from a provided list or input one in the text box.
- `multiselect` : Select one or more values from a list of provided values.

- `Widget dropdowns` and `text boxes` appear immediately following the notebook toolbar.

- `dbutils.widgets.text And dbutils.widgets.get` we are using to create variables and passing values.
- `text(name: String, defaultValue: String, label: String)` : void -> Creates a text input widget with a given name and default value
- `get(name: String)` : String -> Retrieves current value of an input widget
- `combobox(name: String, defaultValue: String, choices: Seq, label: String)` : void -> Creates a combobox input widget with a
  - given name, default value and choices
- `dropdown(name: String, defaultValue: String, choices: Seq, label: String)` : void -> Creates a dropdown input widget a
  - with given name, default value and choices

Dbutils.widgets.text()



```
Cmd 3
```

`text(name: String, defaultValue: String, label: String)`

- Creates a text input widget with a given name and default value

```
Cmd 4
dbutils.widgets.text("foo", "text","text_label")
print(dbutils.widgets.get("foo"))
```

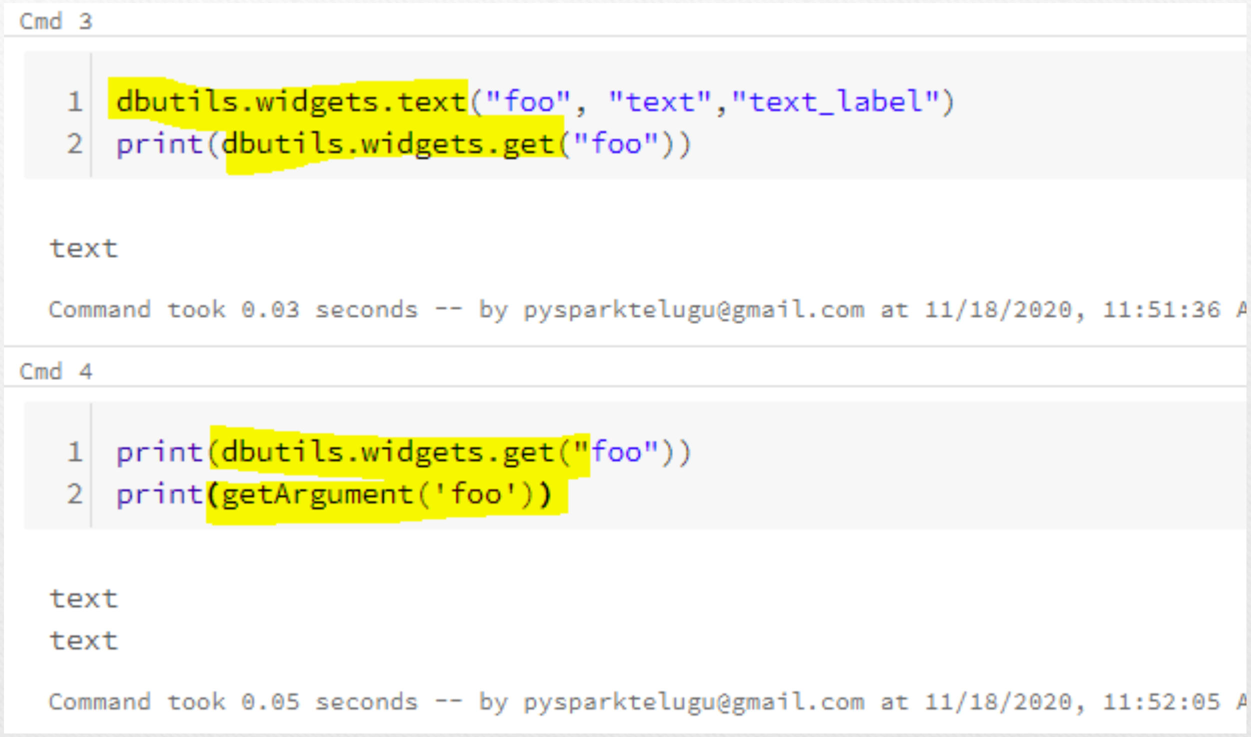
```
text

Command took 0.03 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 11:51:36 AM on datacluster
```

```
Cmd 5
print(dbutils.widgets.get("foo"))
print(getArgument('foo'))
```

```
text
text

Command took 0.05 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 11:52:05 AM on datacluster
```

Get or GetArgument for Getting values

You can access the current value of the widget with the call Using Get or GetArgument methods.



Cmd 3

```
dbutils.widgets.text("foo", "text","text_label")
print(dbutils.widgets.get("foo"))
```

```
text

Command took 0.03 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 11:51:36 A
```

Cmd 4

```
print(dbutils.widgets.get("foo"))
print(getArgument('foo'))
```

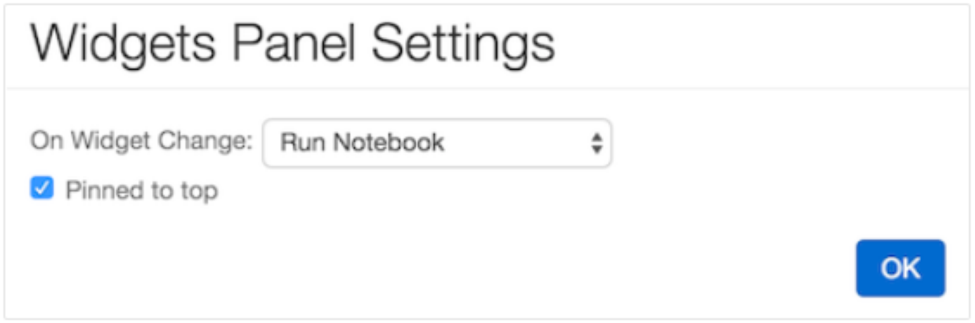
```
text
text

Command took 0.05 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 11:52:05 A
```

Configure widget settings

You can configure the behavior of widgets when a new value is selected and whether the widget panel is always pinned to the top of the notebook.

1. Click the ⚙ icon at the right end of the Widget panel.

2. In the pop-up Widget Panel Settings dialog box, choose the widget's execution behavior.


Widgets Panel Settings

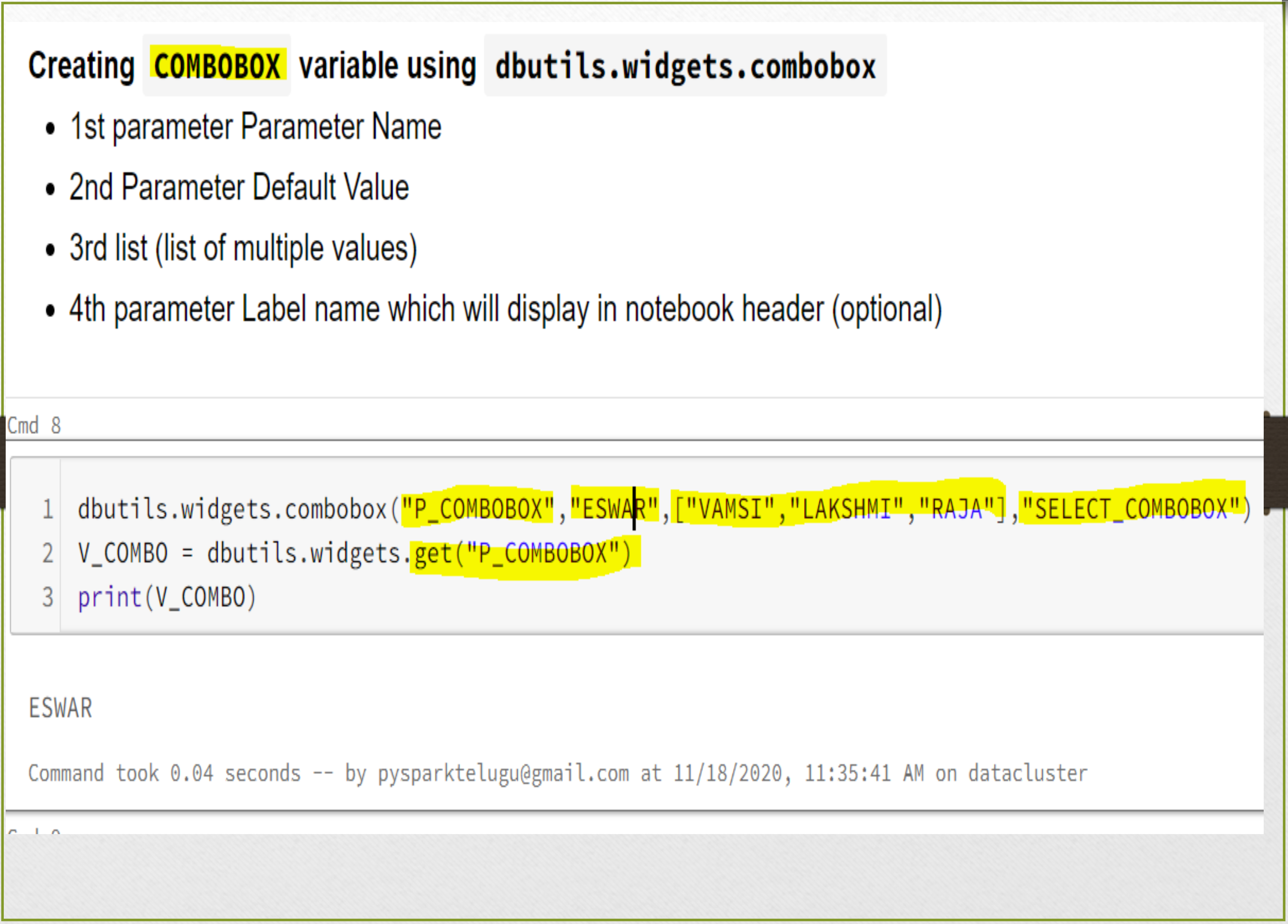
On Widget Change: Run Notebook

✓ Pinned to top

- **Run Notebook**: Every time a new value is selected, the entire notebook is rerun.
- **Run Accessed Commands**: Every time a new value is selected, only cells that retrieve the values for that particular widget are rerun. This is the default setting when you create a widget.

  > **Note**
  >
  > SQL cells are not rerun in this configuration.

- **Do Nothing**: Every time a new value is selected, nothing is rerun.



Creating COMBOBOX variable using `dbutils.widgets.combobox`

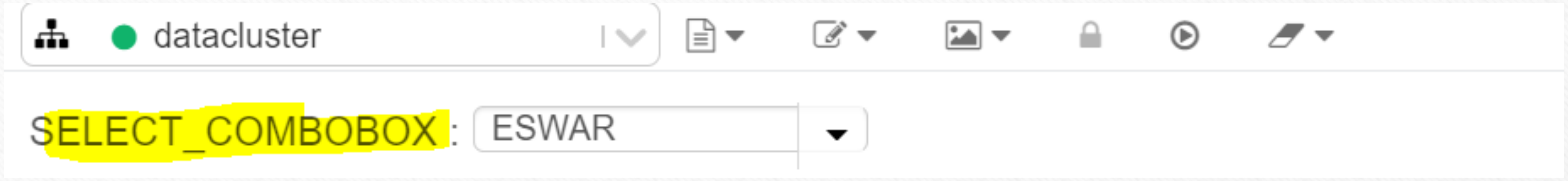
- 1st parameter Parameter Name
- 2nd Parameter Default Value
- 3rd list (list of multiple values)
- 4th parameter Label name which will display in notebook header (optional)

Cmd 8

```
dbutils.widgets.combobox("P_COMBOBOX","ESWAR",["VAMSI","LAKSHMI","RAJA"],"SELECT_COMBOBOX")
V_COMBO = dbutils.widgets.get("P_COMBOBOX")
print(V_COMBO)
```

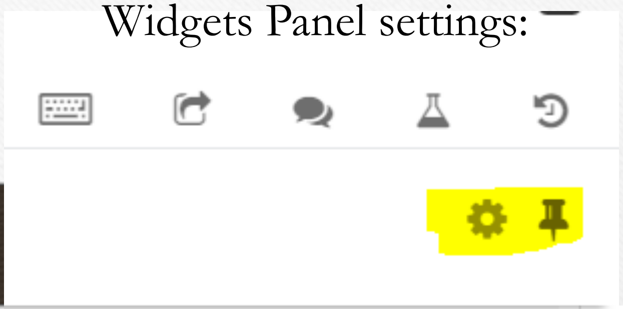
```
ESWAR
```

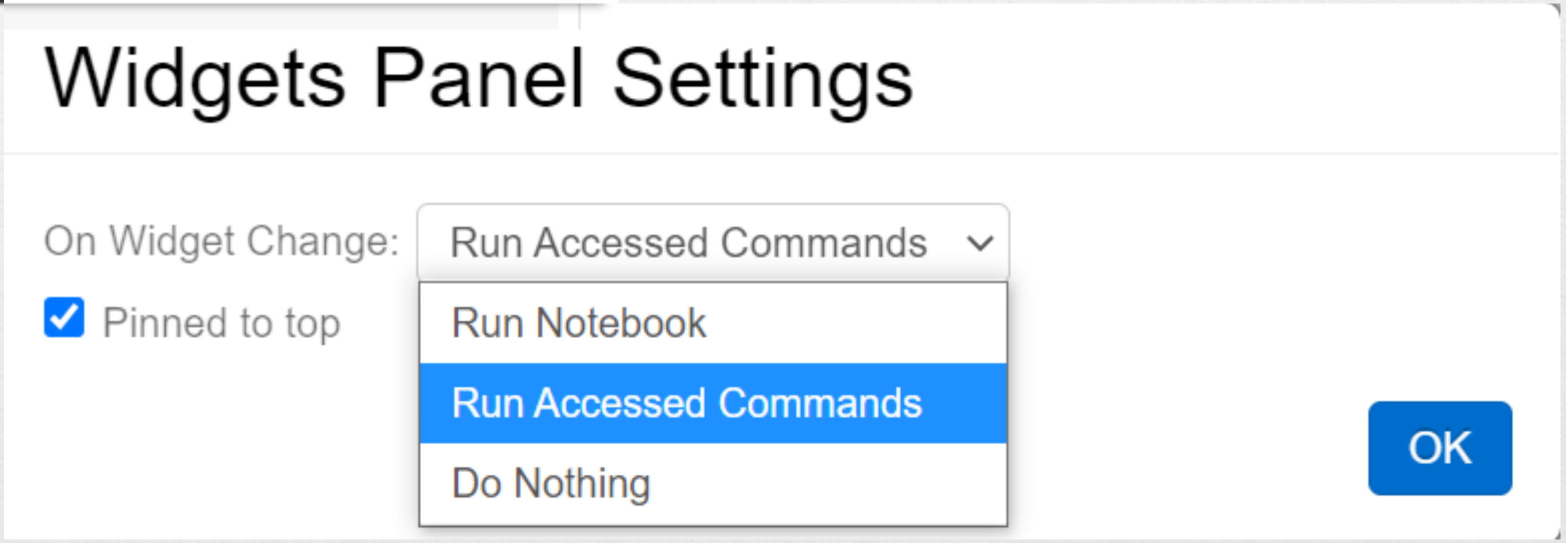
Command took 0.04 seconds -- by pysparktelugu@gmail.com at 11/18/2020, 11:35:41 AM on datacluster


datacluster

SELECT_COMBOBOX : ESWAR

Widgets Panel settings: –



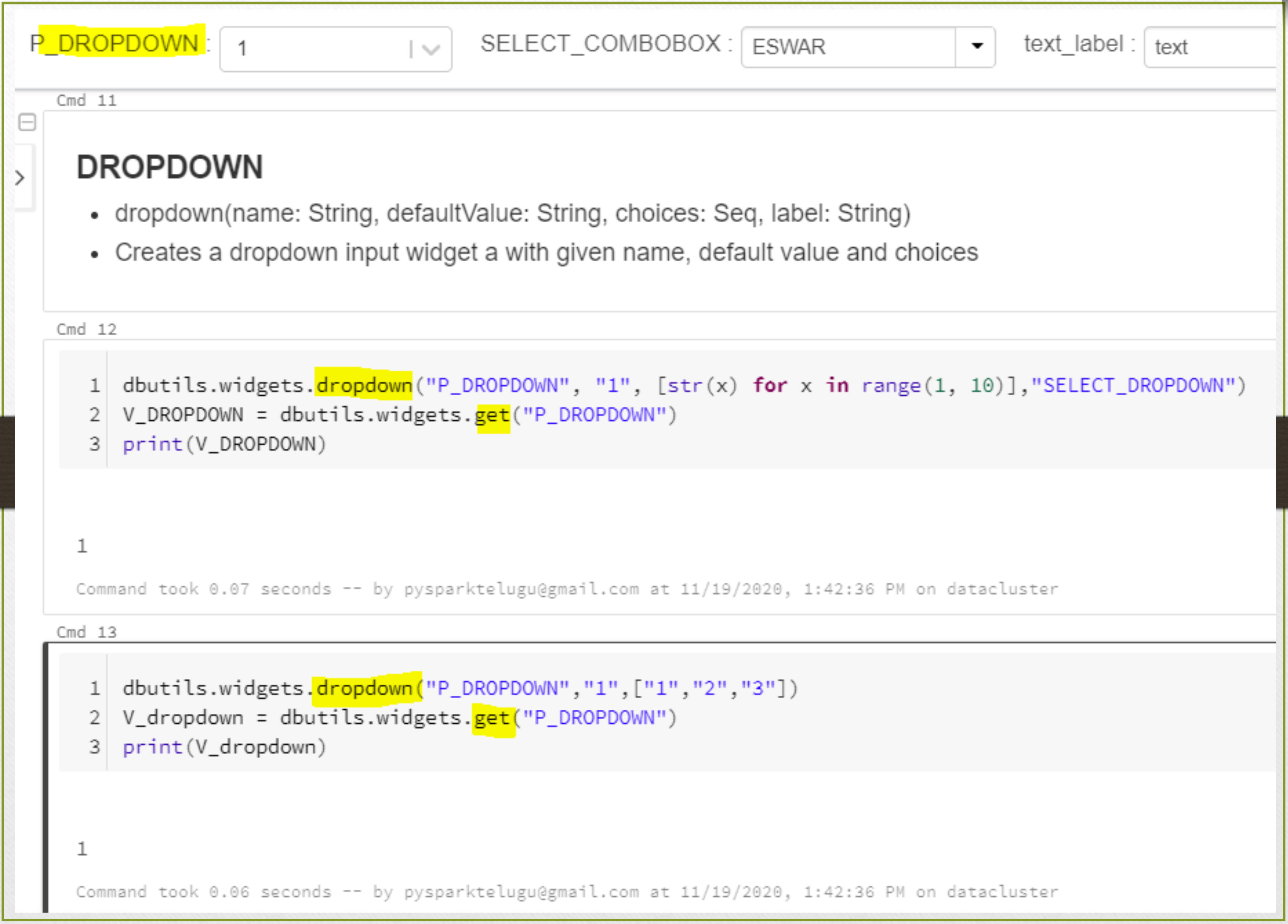

Widgets Panel Settings

On Widget Change: Run Accessed Commands

- Run Notebook
- Run Accessed Commands
- Do Nothing

☑ Pinned to top

OK


P_DROPDOWN : 1   SELECT_COMBOBOX : ESWAR   text_label : text

Cmd 11

DROPDOWN

- dropdown(name: String, defaultValue: String, choices: Seq, label: String)
- Creates a dropdown input widget a with given name, default value and choices

Cmd 12

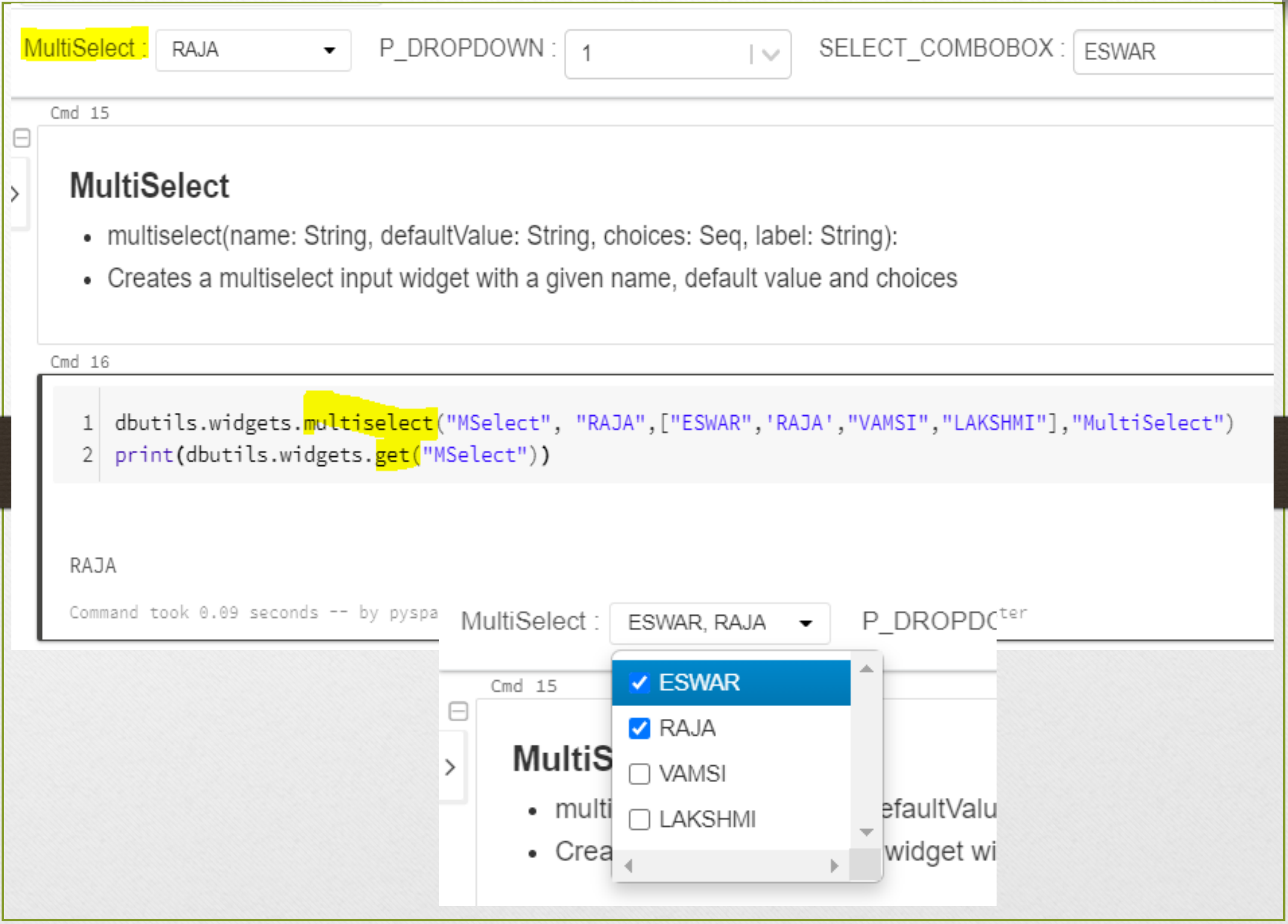
```
dbutils.widgets.dropdown("P_DROPDOWN", "1", [str(x) for x in range(1, 10)],"SELECT_DROPDOWN")
V_DROPDOWN = dbutils.widgets.get("P_DROPDOWN")
print(V_DROPDOWN)
```

```
1

Command took 0.07 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 1:42:36 PM on datacluster
```

Cmd 13

```
dbutils.widgets.dropdown("P_DROPDOWN","1",["1","2","3"])
V_dropdown = dbutils.widgets.get("P_DROPDOWN")
print(V_dropdown)
```

```
1

Command took 0.06 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 1:42:36 PM on datacluster
```



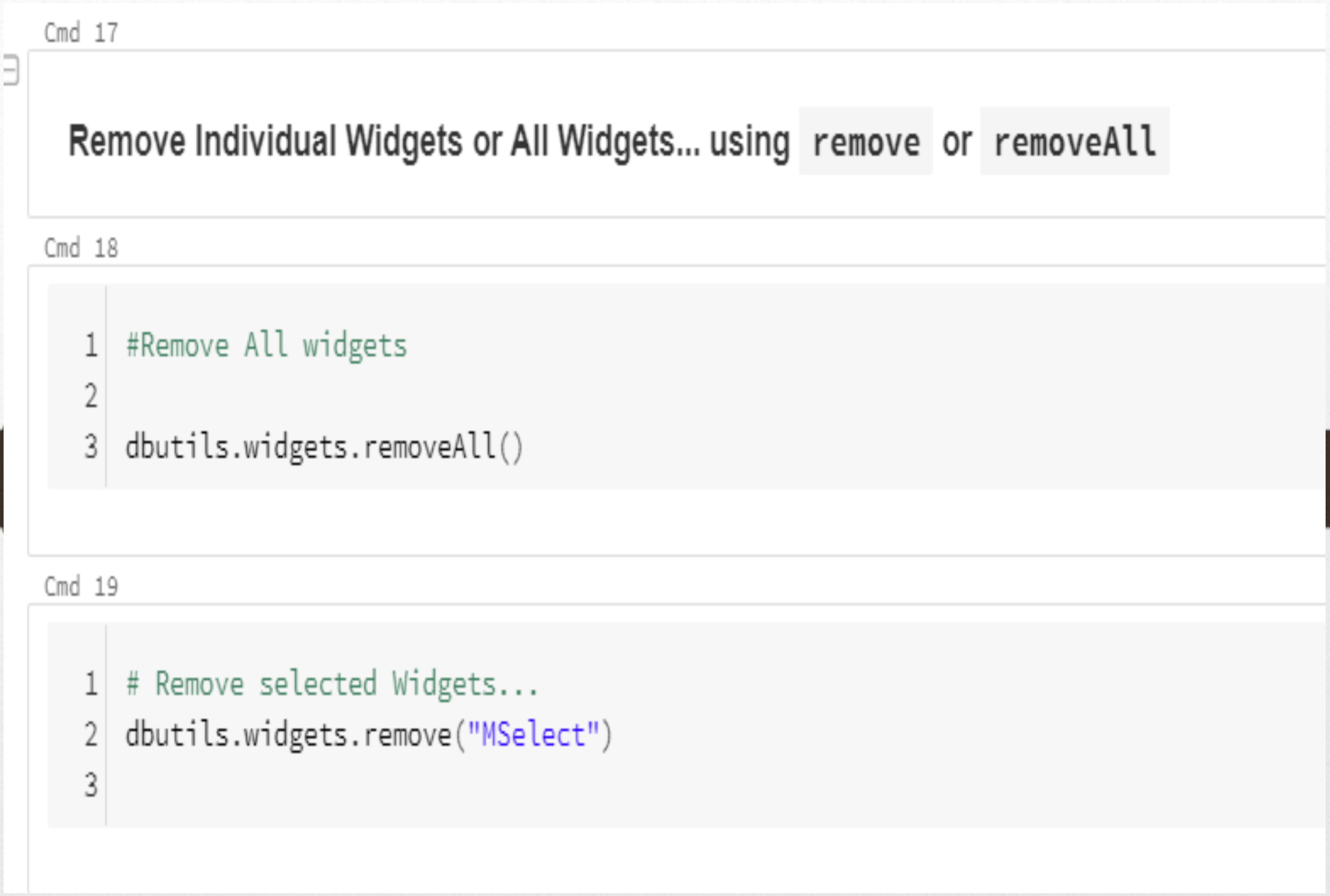
MultiSelect

- multiselect(name: String, defaultValue: String, choices: Seq, label: String):
- Creates a multiselect input widget with a given name, default value and choices

```
dbutils.widgets.multiselect("MSelect", "RAJA",["ESWAR",'RAJA',"VAMSI","LAKSHMI"],"MultiSelect")
print(dbutils.widgets.get("MSelect"))
```

```
RAJA
```

Command took 0.09 seconds -- by pyspa
MultiSelect : ESWAR, RAJA
P_DROPDC
✓ ESWAR
✓ RAJA
VAMSI
LAKSHMI
Cmd 15


Cmd 17

Remove Individual Widgets or All Widgets... using `remove` or `removeAll`

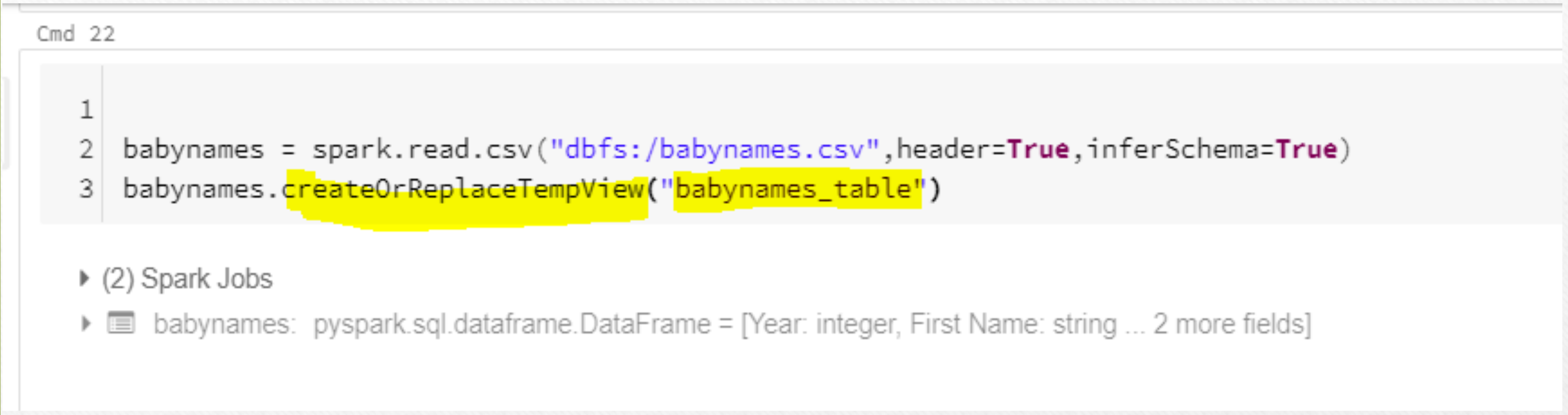
Cmd 18

```
#Remove All widgets

dbutils.widgets.removeAll()
```

Cmd 19

```
# Remove selected Widgets...
dbutils.widgets.remove("MSelect")

```

Creating Temporary table..

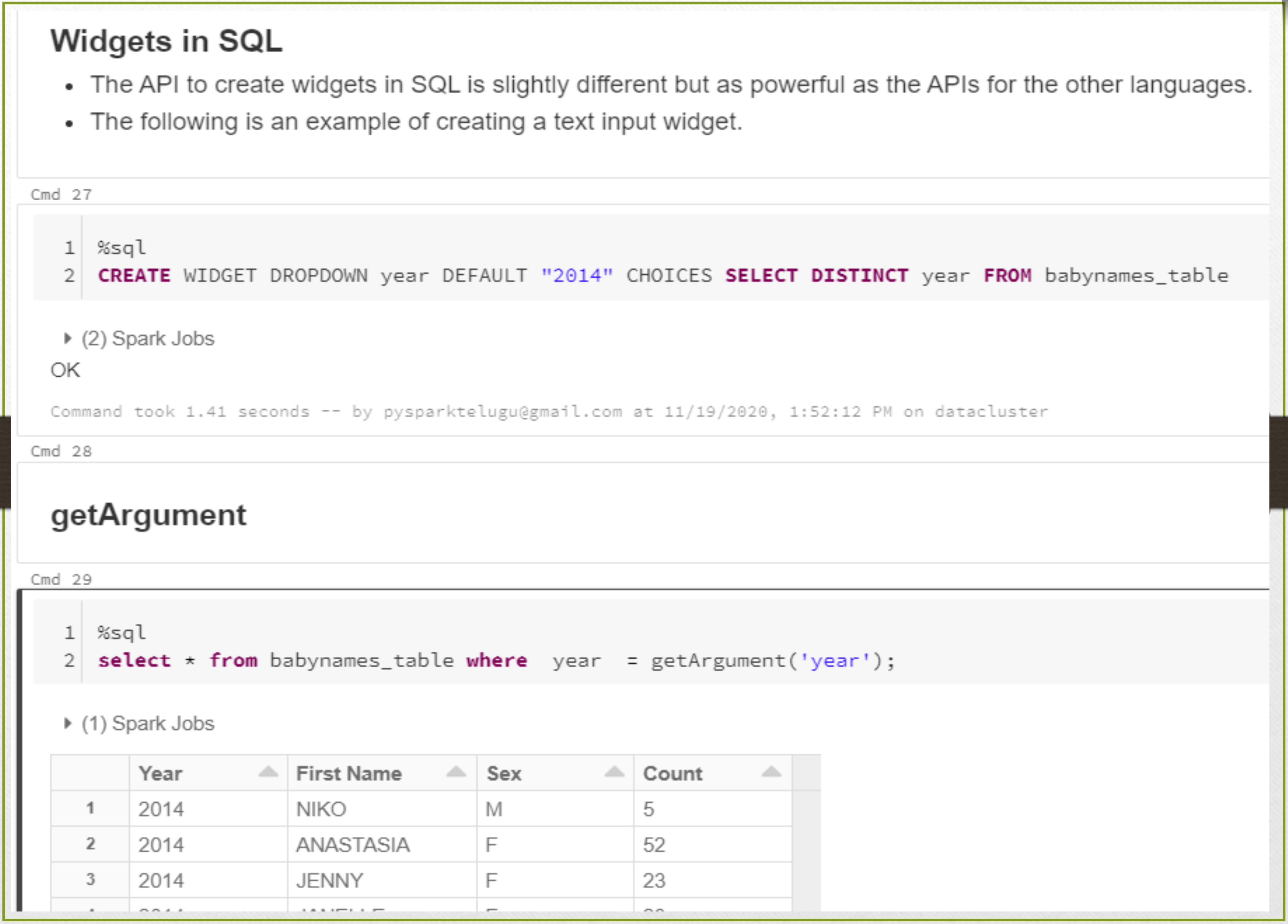


Cmd 22

```
babynames = spark.read.csv("dbfs:/babynames.csv",header=True,inferSchema=True)
babynames.createOrReplaceTempView("babynames_table")
```

▸ (2) Spark Jobs

▸ babynames: pyspark.sql.dataframe.DataFrame = [Year: integer, First Name: string ... 2 more fields]



Widgets in SQL

- The API to create widgets in SQL is slightly different but as powerful as the APIs for the other languages.
- The following is an example of creating a text input widget.

Cmd 27

```
%sql
CREATE WIDGET DROPDOWN year DEFAULT "2014" CHOICES SELECT DISTINCT year FROM babynames_table
```

▸ (2) Spark Jobs

```
OK

Command took 1.41 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 1:52:12 PM on datacluster
```

Cmd 28

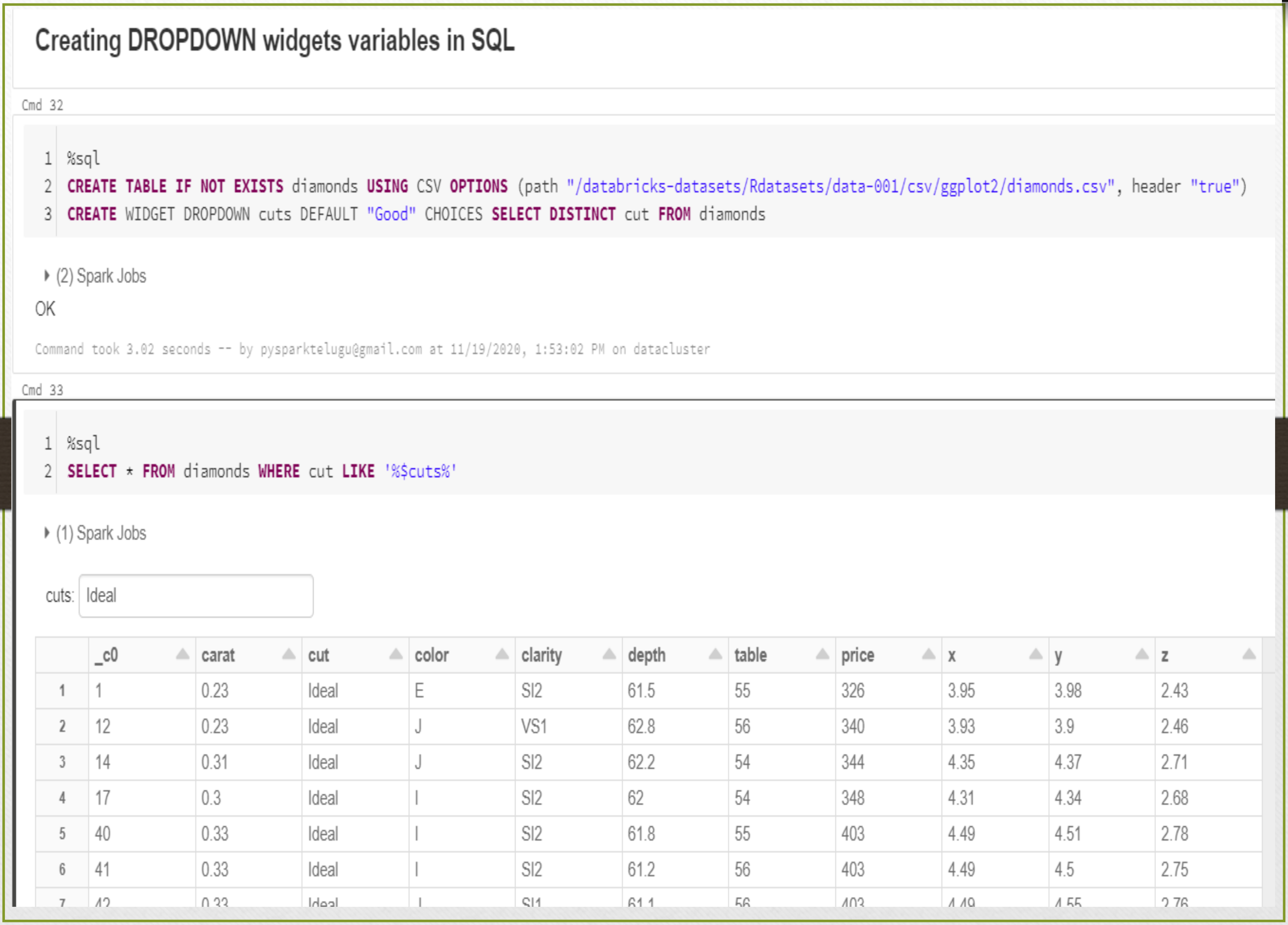
getArgument

Cmd 29

```
%sql
select * from babynames_table where  year  = getArgument('year');
```

▸ (1) Spark Jobs

	Year	First Name	Sex	Count
1	2014	NIKO	M	5
2	2014	ANASTASIA	F	52
3	2014	JENNY	F	23
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Creating DROPDOWN widgets variables in SQL

Cmd 32

```
%sql
CREATE TABLE IF NOT EXISTS diamonds USING CSV OPTIONS (path "/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header "true")
CREATE WIDGET DROPDOWN cuts DEFAULT "Good" CHOICES SELECT DISTINCT cut FROM diamonds
```

▸ (2) Spark Jobs

OK

Command took 3.02 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 1:53:02 PM on datacluster

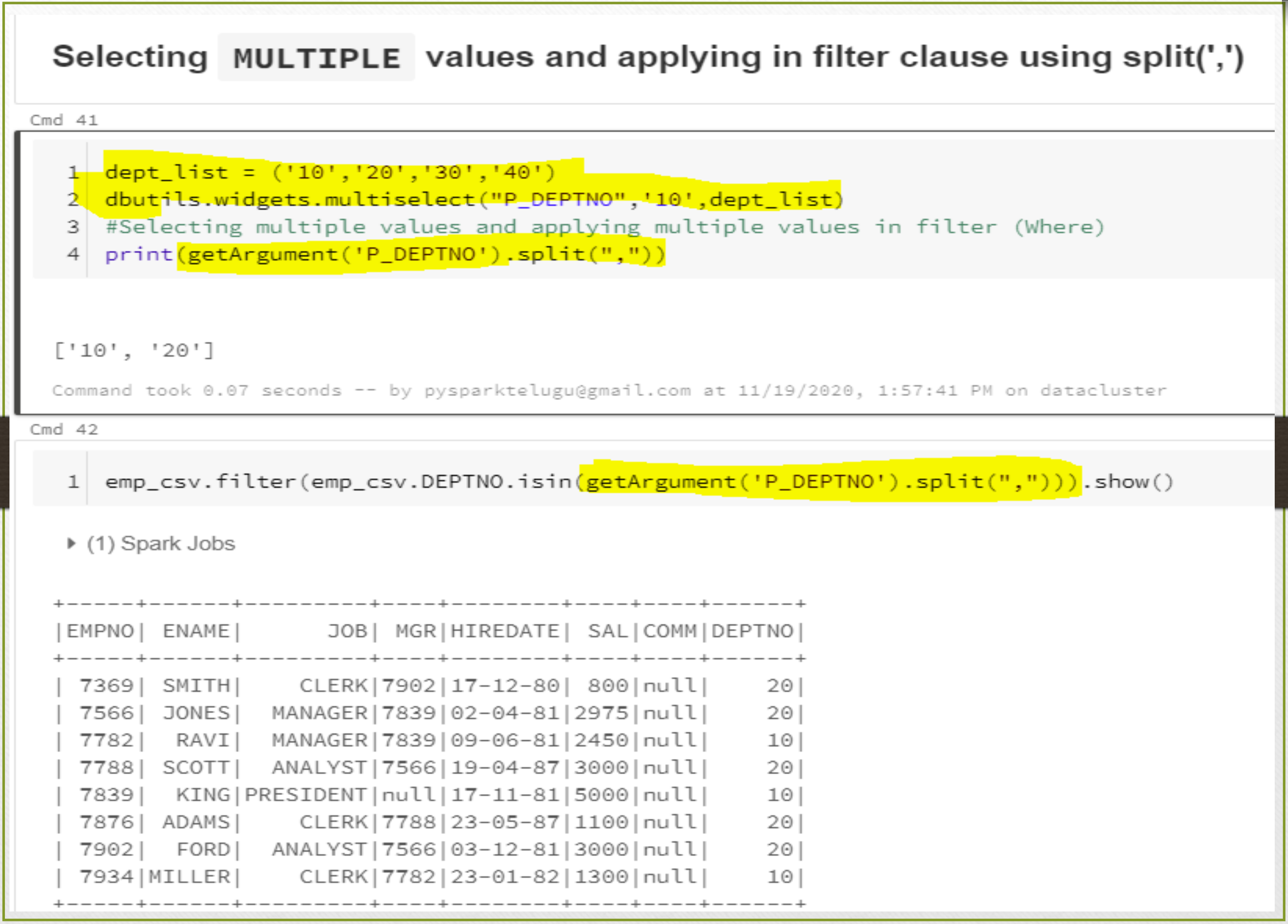
Cmd 33

```
%sql
SELECT * FROM diamonds WHERE cut LIKE '%$cuts%'
```

▸ (1) Spark Jobs

cuts: Ideal

	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
1	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	12	0.23	Ideal	J	VS1	62.8	56	340	3.93	3.9	2.46
3	14	0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71
4	17	0.3	Ideal	I	SI2	62	54	348	4.31	4.34	2.68
5	40	0.33	Ideal	I	SI2	61.8	55	403	4.49	4.51	2.78
6	41	0.33	Ideal	I	SI2	61.2	56	403	4.49	4.5	2.75
7	42	0.33	Ideal	I	SI1	61.1	56	403	4.49	4.55	2.76



Selecting `MULTIPLE` values and applying in filter clause using split(',')

Cmd 41

```
dept_list = ('10','20','30','40')
dbutils.widgets.multiselect("P_DEPTNO",'10',dept_list)
#Selecting multiple values and applying multiple values in filter (Where)
print(getArgument('P_DEPTNO').split(","))
```

```
['10', '20']
```

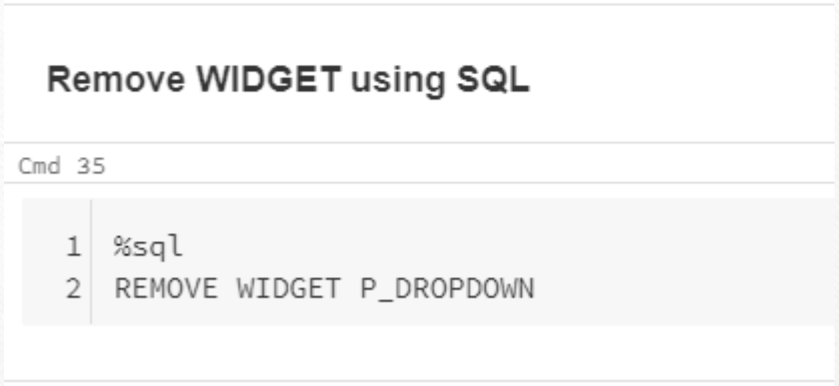
Command took 0.07 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 1:57:41 PM on datacluster

Cmd 42

```
emp_csv.filter(emp_csv.DEPTNO.isin(getArgument('P_DEPTNO').split(","))).show()
```

▸ (1) Spark Jobs

```
+-----+------+---------+----+--------+----+----+------+
|EMPNO| ENAME|      JOB| MGR|HIREDATE| SAL|COMM|DEPTNO|
+-----+------+---------+----+--------+----+----+------+
| 7369| SMITH|    CLERK|7902|17-12-80| 800|null|    20|
| 7566| JONES|  MANAGER|7839|02-04-81|2975|null|    20|
| 7782|  RAVI|  MANAGER|7839|09-06-81|2450|null|    10|
| 7788| SCOTT|  ANALYST|7566|19-04-87|3000|null|    20|
| 7839|  KING|PRESIDENT|null|17-11-81|5000|null|    10|
| 7876| ADAMS|    CLERK|7788|23-05-87|1100|null|    20|
| 7902|  FORD|  ANALYST|7566|03-12-81|3000|null|    20|
| 7934|MILLER|    CLERK|7782|23-01-82|1300|null|    10|
+-----+------+---------+----+--------+----+----+------+
```

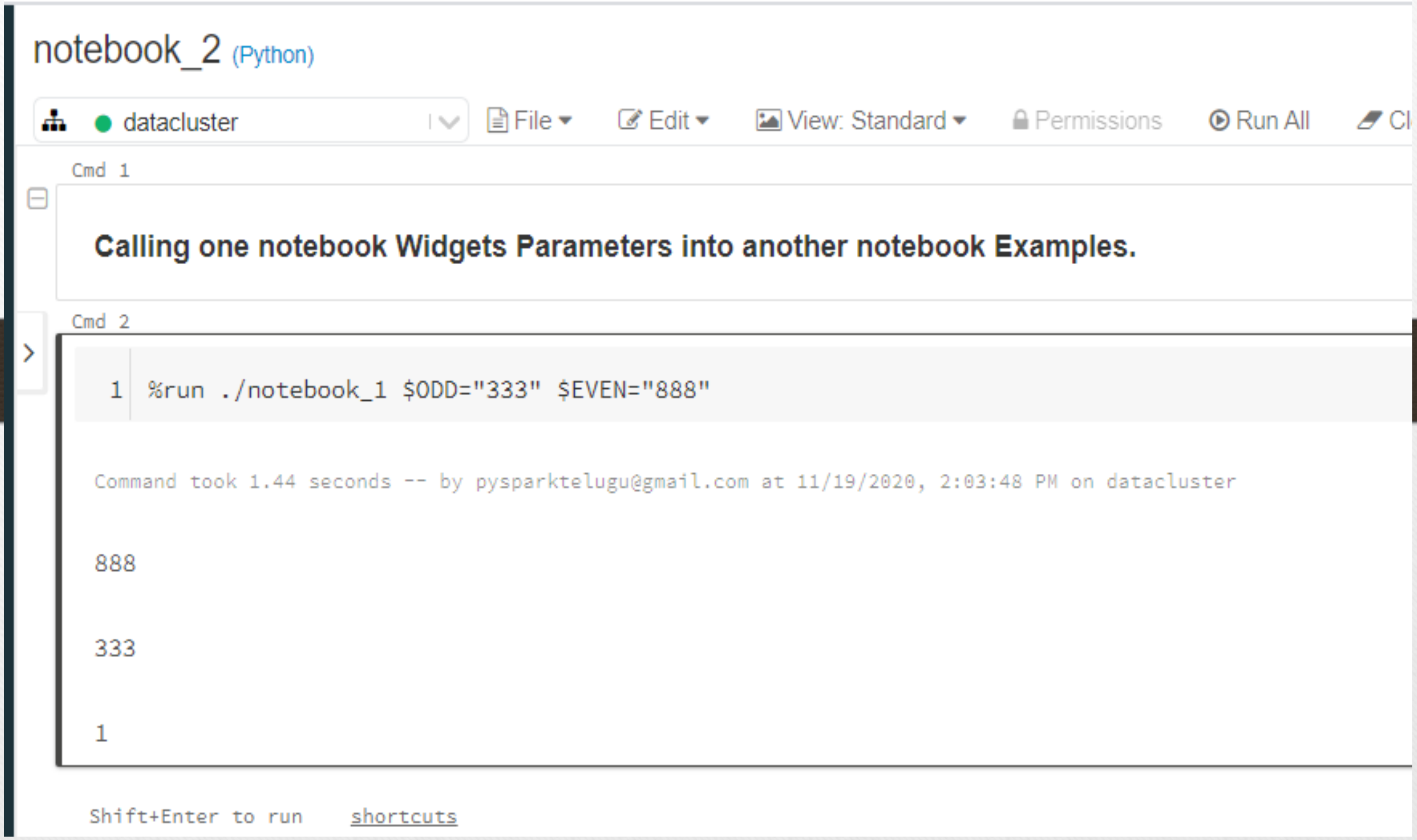


Remove WIDGET using SQL

Cmd 35

```
%sql
REMOVE WIDGET P_DROPDOWN
```

Calling One Notebook Into Another using widgets parameters.

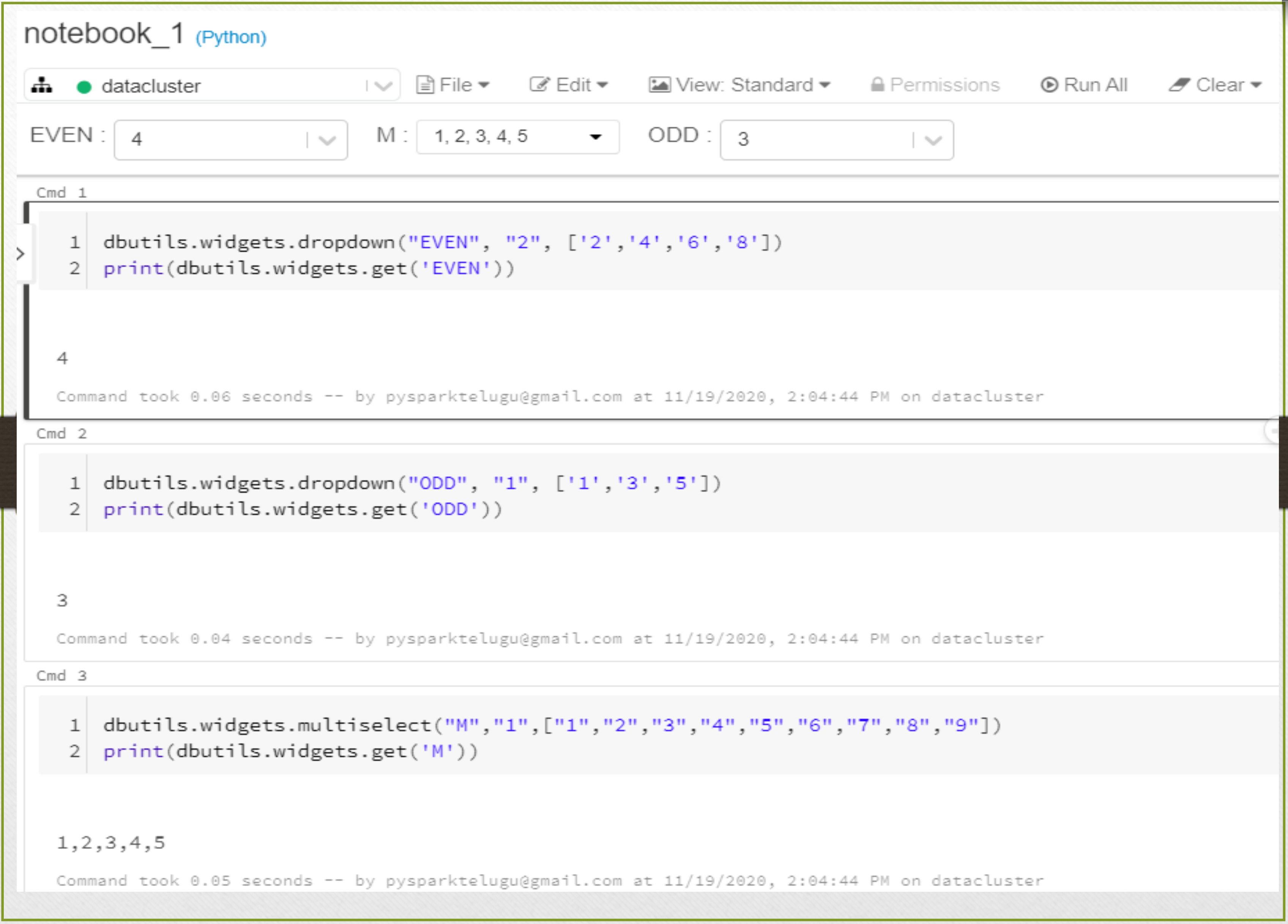


Calling one notebook Widgets Parameters into another notebook Examples.

```
%run ./notebook_1 $ODD="333" $EVEN="888"
```

```
Command took 1.44 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 2:03:48 PM on datacluster

888

333

1
```

Shift+Enter to run   shortcuts



EVEN : 4   M : 1, 2, 3, 4, 5   ODD : 3

Cmd 1

```
dbutils.widgets.dropdown("EVEN", "2", ['2','4','6','8'])
print(dbutils.widgets.get('EVEN'))
```

```
4
```

Command took 0.06 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 2:04:44 PM on datacluster

Cmd 2

```
dbutils.widgets.dropdown("ODD", "1", ['1','3','5'])
print(dbutils.widgets.get('ODD'))
```

```
3
```

Command took 0.04 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 2:04:44 PM on datacluster

Cmd 3

```
dbutils.widgets.multiselect("M","1",["1","2","3","4","5","6","7","8","9"])
print(dbutils.widgets.get('M'))
```

```
1,2,3,4,5
```

Command took 0.05 seconds -- by pysparktelugu@gmail.com at 11/19/2020, 2:04:44 PM on datacluster

All The Best. 📚

Keep Learning and Sharing