

# Data Engineering in the Cloud

## Introduction to Azure Databricks

Xuemao Zhang  
East Stroudsburg University

January 18, 2025

# Outline

- Introduction to Azure Databricks
- Databricks File System
- Lab 1: Creating an Azure Databricks Workspace
- Lab 2: Creating a Cluster
- Lab 3: Using DataFrames in Azure Databricks

# Introduction to Azure Databricks

- Azure Databricks is a fast, easy, and collaborative **Apache Spark-based** analytics platform optimized for Microsoft Azure.
- Key Features
  - ▶ Collaborative Workspaces: Interactive environments for team collaboration
  - ▶ Automated Cluster Management: Easy scaling and management of Spark clusters
  - ▶ Machine Learning: Built-in tools for developing and deploying models
  - ▶ Seamless Integration: Works with Azure services like Synapse Analytics, Data Lake Storage, and Power BI

# Introduction to Azure Databricks



*Data Engineer*



*Data Factory*



*Synapse*



*ADLS*

Data Service Integrations



*Databricks I/O*



*Delta Lake*



*Spark*

Databricks Runtime Engine



*Data Scientist*

ML Management Services



*Azure ML*



*MLflow*

ML Runtime



*PyTorch*



*TensorFlow*



*Databricks I/O*



*Delta Lake*



*Spark*

Databricks Runtime Engine

# Reading and Writing Data in Azure Databricks

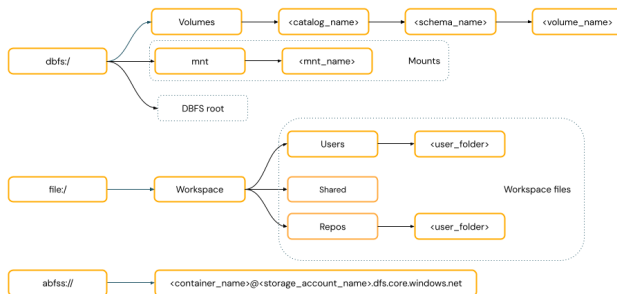
- Several types of data can be read with Azure Databricks
- Reading Data in CSV Format
- Reading Data in JSON Format
  - ▶ `val df = spark.read.json("example.json")`
  - ▶ `val mdf =  
 spark.read.option("multiline","true").json("/tmp/demo.json")  
 mdf.show(false)`
- Reading Data in Parquet Format
  - ▶ `data = spark.read.parquet("/tmp/Parquetfile")`

# Reading and Writing Data in Azure Databricks

- Writing data: Apache Spark can be used to write data frame into disk and data into multiple files.
  - ▶ `df.write.format("csv").mode("overwrite").save("/my-file/demo csv")`

# Databricks File System

- Databricks file system, also known as DBFS, is a distributed file system.
  - ▶ It has a master slave architecture to attain reliability.
  - ▶ <https://learn.microsoft.com/en-us/azure/databricks/files/>
- URI-style paths include a URI scheme.



# Databricks File System

- The structure shows how different storage paths are organized and accessed in Azure Databricks.
- `dbfs:/`
  - ▶ Volumes: Structured as `<catalog_name>/<schema_name>/<volume_name>`.
  - ▶ `mnt`: Mount points for external storage, structured as `<mnt_name>`.

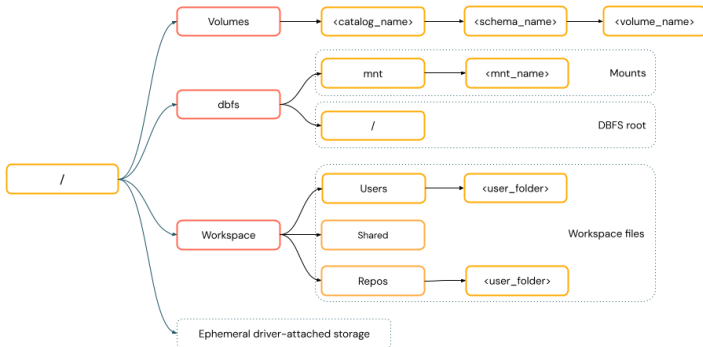
`file:/`

- Workspace: Contains:
  - ▶ Users: Individual user folders.
  - ▶ Shared: Shared files.
  - ▶ Repos: Repositories, structured as `<user_folder>`.
- `abfss:///`: Connects to Azure Blob Storage using the format `<container_name>@<storage_account_name>.dfs.core.windows.net`.



# Databricks File System

- POSIX-style paths provide data access relative to the driver root (/)



# Lab 1: Creating an Azure Databricks Workspace

- First follow what we did before, create a Data Lake Storage account and upload the data population.csv to the storage

## Create a storage account ...

Basics   Advanced   Networking   Data protection   Encryption   Tags   Review + create

[View automation template](#)

### Basics

Subscription	Azure for Students
Resource group	NetworkWatcherRG
Location	East US
Storage account name	xzhang2storage
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

### Advanced

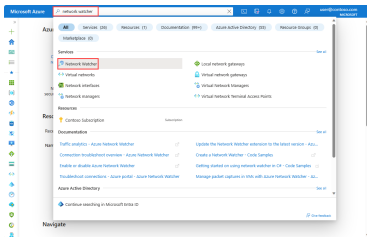
Enable hierarchical namespace	Enabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

### Security

Secure transfer	Enabled
Blob anonymous access	Disabled
Allow storage account key access	Enabled
Default to Microsoft Entra authorization in the Azure portal	Disabled

# Lab 1: Creating an Azure Databricks Workspace

- Network Watcher is a regional service that enables you to monitor and diagnose conditions at a network scenario level in, to, and from Azure. - It includes a suite of tools for monitoring, diagnosing, viewing metrics, and enabling or disabling logs for resources in a virtual network.
- This resource group is automatically generated by Azure when you enable Network Watcher in a region.
  - ▶ <https://learn.microsoft.com/en-gb/azure/network-watcher/network-watcher-overview>
- When you enable Network Watcher, Azure creates a resource group named NetworkWatcherRG in each region where it is enabled.



# Lab 1: Creating an Azure Databricks Workspace

- First follow what we did before, create a Data Lake Storage account and upload the data `population.csv` to the storage

Home > xzhang2storage | Containers >

 **raw** ...  
Container

  Upload  Add Directory  Refresh |  Rename  Delete  Change tier  Acquire lease  Break lease  Give feedback

 Overview

 Diagnose and solve problems

 Access Control (IAM)

> Settings

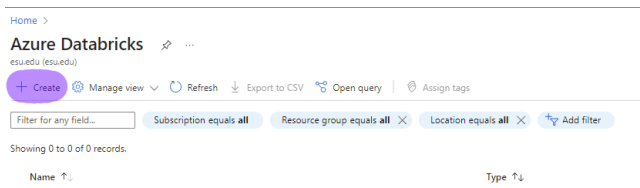
Authentication method: Access key ([Switch to Microsoft Entra user account](#))

Location: raw

Name	Modified	Access tier	Archive status
<input type="checkbox"/>  population.csv	7/18/2024, 11:19:59 AM	Hot (Inferred)	

# Lab 1: Creating an Azure Databricks Workspace

- Search for Azure Databricks in the search bar and click on the Create



No azure

# Lab 1: Creating an Azure Databricks Workspace

- Specify workspace name, region, and pricing tier as shown below:

Microsoft Azure

Home > Azure Databricks >

## Create an Azure Databricks workspace

Basics Networking Encryption Security & compliance Tags Review + create

### Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Azure for Students

Resource group \* ⓘ NetworkWatcherRG  
[Create new](#)

### Instance Details

Workspace name \* AZdatabricks ✓

Region \* East US

Pricing Tier \* ⓘ Standard (Apache Spark, Secure with Microsoft Entra ID)

Managed Resource Group name Enter name for managed resource group

# Lab 1: Creating an Azure Databricks Workspace

- We can leave the other settings to default and go to and click review + create

[Home](#) > [Azure Databricks](#) >

## Create an Azure Databricks workspace ...

✓ Validation Succeeded

[Basics](#) [Networking](#) [Encryption](#) [Security & compliance](#) [Tags](#) [Review + create](#)

### Summary

#### Basics

Workspace name	AZdatabricks
Subscription	Azure for Students
Resource group	NetworkWatcherRG
Region	East US
Pricing Tier	standard
Managed Resource Group name	

#### Networking

Deploy Azure Databricks workspace with Secure Cluster Connectivity (No Public IP)	No
Deploy Azure Databricks workspace in your own Virtual Network (VNet)	No

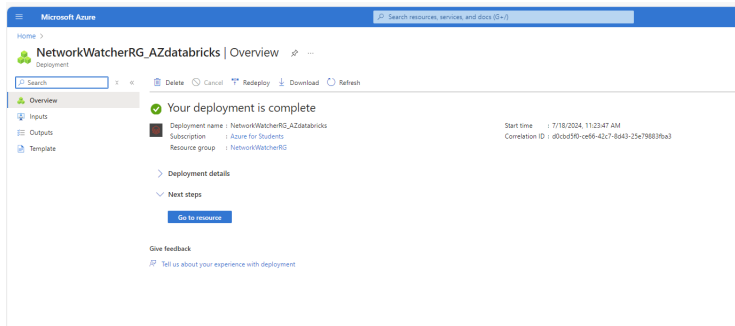
#### Encryption

Enable Infrastructure Encryption	No
Enable CMK for Managed Disks	No
Enable CMK for Managed Services	No

[Security & compliance](#)

# Lab 1: Creating an Azure Databricks Workspace

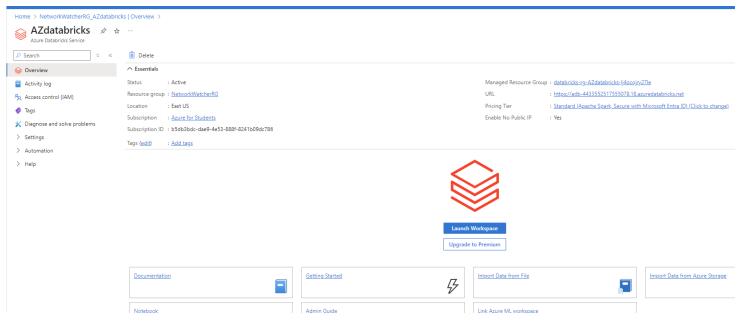
- Once the deployment is completed, click on the Go to resource button





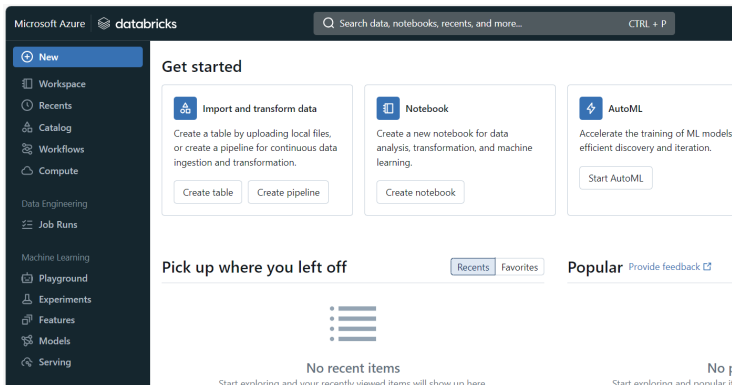
# Lab 1: Creating an Azure Databricks Workspace

- Click on the “Launch workspace” button to launch the workspace



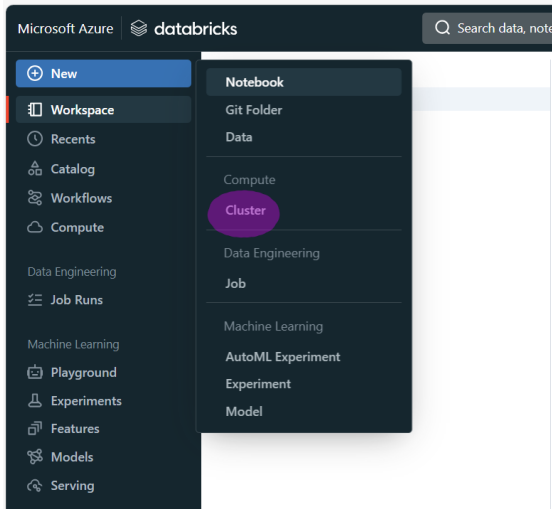
# Lab 1: Creating an Azure Databricks Workspace

- This is how the Azure Databricks workspace looks like



## Lab 2: Creating a Cluster

- Databricks eliminates the complexity and provides convenience to get a Spark cluster. The Spark experience is seamless and requires no management.
- Click on Cluster under the menu + New to create a cluster



# Lab 2: Creating a Cluster

- Specify the name for the “Cluster”; and Select the Cluster mode

Microsoft Azure | databricks

Search data, notebooks, recents, and more...

Compute → New compute →

**Cluster1**

☐ Multi node ☒ Single node

Access mode ⓘ Single user access ⓘ

Single user Xuemao Zhang

**Performance**

Databricks runtime version ⓘ

Runtime: 14.3 LTS (Scala 2.12, Spark 3.5.0)

☐ Use Photon Acceleration ⓘ

Node type ⓘ

Standard\_D4ds\_v5 16 GB Memory, 4 Cores

6 more

**General purpose**

Standard_DS3_v2	14 GB Memory, 4 Cores
Standard_DS4_v2	28 GB Memory, 8 Cores
Standard_DS5_v2	56 GB Memory, 16 Cores
Standard_D4s_v3	16 GB Memory, 4 Cores

## Lab 2: Creating a Cluster

- We can specify the termination time as well. The defaulted value is 120 minutes
  - ▶ And then click on the Create compute button

Microsoft Azure | databricks

Search data, notebooks, recents, and more

New

- Workspace
- Recents
- Catalog
- Workflows
- Compute

Data Engineering

- Job Runs

Machine Learning

- Playground
- Experiments
- Features
- Models
- Serving

Partner Connect

Collapse menu

Compute > New compute >

### Cluster1

☐ Multi node ☒ Single node

Access mode ☐ Single user access ☐ Single user access

Single user | Xuemao Zhang

#### Performance

Databricks runtime version ☐

Runtime: 14.3 LTS (Scala 2.12, Spark 3.5.0)

☐ Use Photon Acceleration

Node type ☐

Standard\_DS3\_v2 14 GB Memory, 4 Cores

☒ Terminate after 60 minutes of inactivity

#### Tags

Add tags

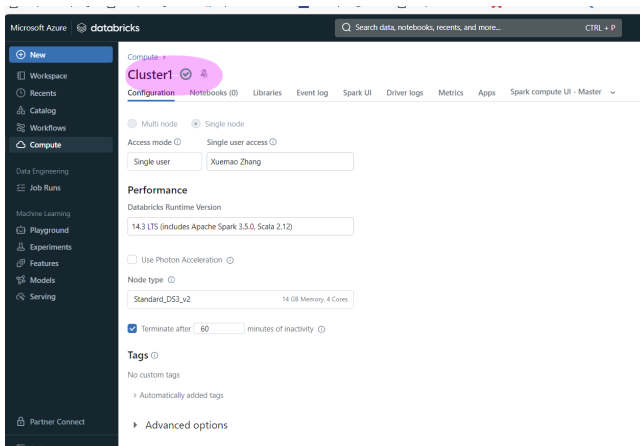
Key	Value
> Automatically added tags	

Advanced options

Create compute Cancel

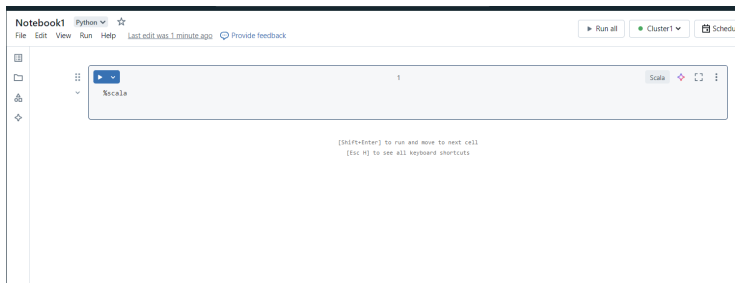
# Lab 2: Creating a Cluster

- If successful, you should be able to see this



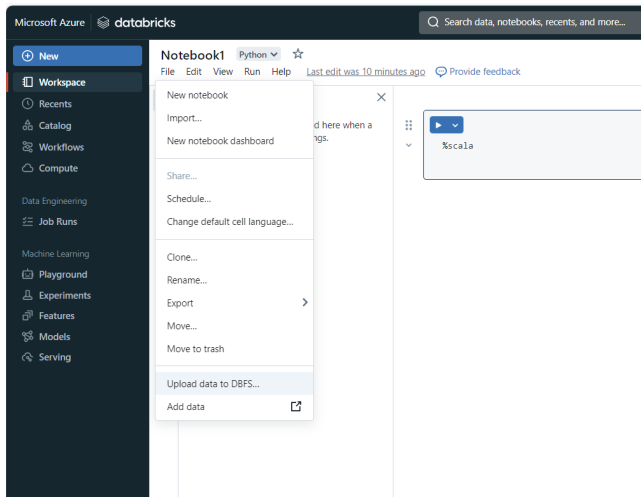
# Lab 3: Using DataFrames in Azure Databricks

- Now create a notebook in the cluster and rename the notebook



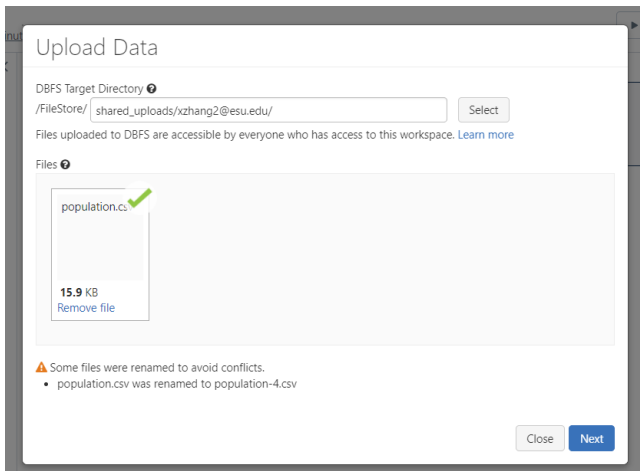
# Lab 3: Using DataFrames in Azure Databricks

- In the menus of the notebook, Click file → Upload data to DBFS... to upload the file population.csv on your local disk
  - ▶ We consider how to get the data from storage later





# Lab 3: Using DataFrames in Azure Databricks

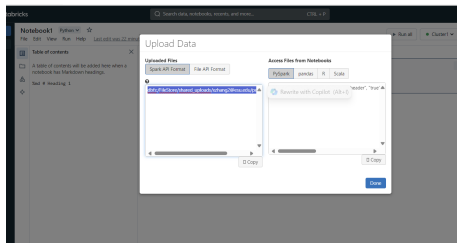


# Lab 3: Using DataFrames in Azure Databricks

- Copy the code from the two boxes

```
dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv
```

```
df1 = spark.read.format("csv").  
option("header", "true").  
load("dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv")
```

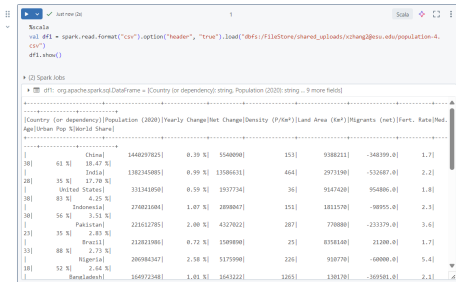


- Click on Done.

# Lab 3: Using DataFrames in Azure Databricks

- Paste the code to the notebook

```
val df1 = spark.read.format("csv").  
option("header", "true").  
load("dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv")  
df1.show()
```



Scala

```
val df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv")  
df1.show()
```

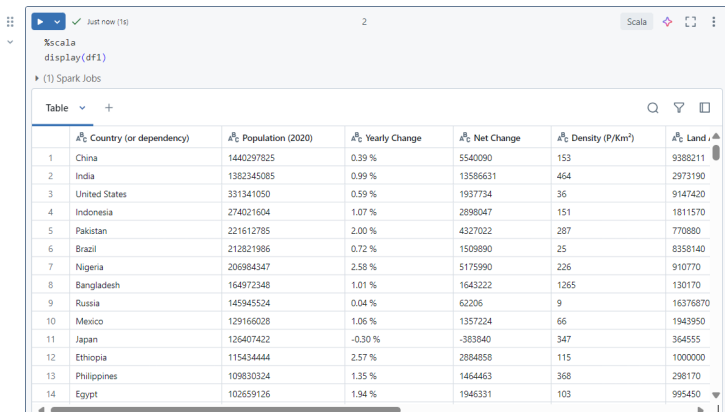
(2) Spark jobs

df1: org.apache.spark.sql.DataFrame = [Country (or dependency): string, Population (2020): string ... 9 more fields]

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/km²)	Land Area (km²)	Migrants (net)	Fort. Rate	Med. Age	Urban Pop %	World Share
1	China	1440297025	0.39 %	5540090	153	9380211	-348399.0	1.7			
38	India	1382345085	0.99 %	13506631	464	2973190	-532687.0	2.2			
28	United States	331341050	0.59 %	1937734	36	9147420	954886.0	1.8			
38	Indonesia	270021004	1.07 %	2098047	151	1811570	-98955.0	2.3			
30	Pakistan	221612785	2.00 %	4327022	207	770880	-233379.0	3.6			
23	Brazil	212823986	0.72 %	1509890	25	8358140	21200.0	1.7			
33	Nigeria	200984347	2.58 %	5175990	226	910770	-60000.0	5.4			
10	Benelux	164972348	1.01 %	1643222	1265	1301701	-309501.0	2.1			

# Lab 3: Using DataFrames in Azure Databricks

- The DataFrame can also be displayed by executing the `display()` function as shown below.



The screenshot shows the Azure Databricks workspace interface. At the top, there's a code editor with the following Scala code:

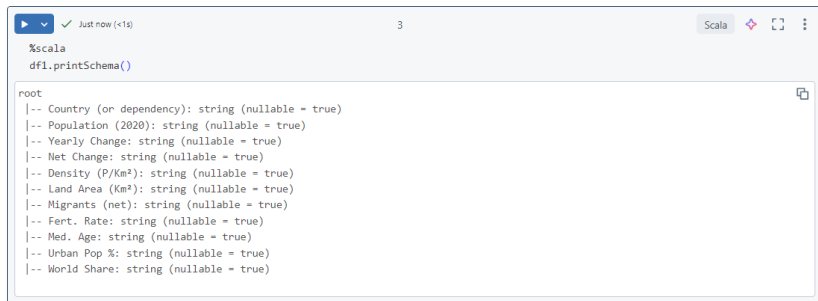
```
%scala
display(df1)
```

Below the code editor, it indicates "(1) Spark Jobs". The main area displays a table with 7 columns: Country (or dependency), Population (2020), Yearly Change, Net Change, Density (P/Km²), and Land. The table contains 14 rows of data, sorted by population.

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land
1	China	1440297825	0.39 %	5540090	153	9388211
2	India	1382345085	0.99 %	13586631	464	2973190
3	United States	331341050	0.59 %	1937734	36	9147420
4	Indonesia	274021604	1.07 %	2898047	151	1811570
5	Pakistan	221612785	2.00 %	4327022	287	770880
6	Brazil	212821986	0.72 %	1509890	25	8358140
7	Nigeria	206984347	2.58 %	5175990	226	910770
8	Bangladesh	164972348	1.01 %	1643222	1265	130170
9	Russia	145945524	0.04 %	62206	9	16376870
10	Mexico	129166028	1.06 %	1357224	66	1943950
11	Japan	126407422	-0.30 %	-383840	347	364555
12	Ethiopia	115434444	2.57 %	2884858	115	1000000
13	Philippines	109830324	1.35 %	1464463	368	298170
14	Egypt	102659126	1.94 %	1946331	103	995450

# Lab 3: Using DataFrames in Azure Databricks

- Study the underlying data types by executing the `printSchema()` function



The screenshot shows a Databricks notebook interface. At the top, there's a status bar with a play button, a checkmark, the text "Just now (<1s)", and the number "3". To the right of the status bar are icons for "Scala", a diamond, a square, and a vertical ellipsis. The main area of the notebook contains the following code:

```
%scala
df1.printSchema()
```

Below the code, the output is displayed in a light blue box. It starts with "root" and then lists the schema of the DataFrame:

```
-- Country (or dependency): string (nullable = true)
-- Population (2020): string (nullable = true)
-- Yearly Change: string (nullable = true)
-- Net Change: string (nullable = true)
-- Density (P/Km²): string (nullable = true)
-- Land Area (Km²): string (nullable = true)
-- Migrants (net): string (nullable = true)
-- Fert. Rate: string (nullable = true)
-- Med. Age: string (nullable = true)
-- Urban Pop %: string (nullable = true)
-- World Share: string (nullable = true)
```

## Lab 3: Using DataFrames in Azure Databricks

- To change the names of the columns in a DataFrame using Scala in Azure Databricks, you can use the `withColumnRenamed` method for each column you want to rename

```
%scala
val df2 = df1
  .withColumnRenamed("Country (or dependency)", "Country")
  .withColumnRenamed("Population (2020)", "Population_2020")
  .withColumnRenamed("Yearly Change", "Yearly_Change")
  .withColumnRenamed("Net Change", "Net_Change")
  .withColumnRenamed("Density (P/Km²)", "Density_PKm2")
  .withColumnRenamed("Land Area (Km²)", "Land_Area_Km2")
  .withColumnRenamed("Migrants (net)", "Migrants_net")
  .withColumnRenamed("Fert. Rate", "Fertility_Rate")
  .withColumnRenamed("Med. Age", "Median_Age")
  .withColumnRenamed("Urban Pop %", "Urban_Pop_Percent")
  .withColumnRenamed("World Share", "World_Share")

df2.printSchema()
```

## Lab 3: Using DataFrames in Azure Databricks

- To filter the columns that are needed, execute the command given below.

```
▶ ✓ Just now (1s) 5

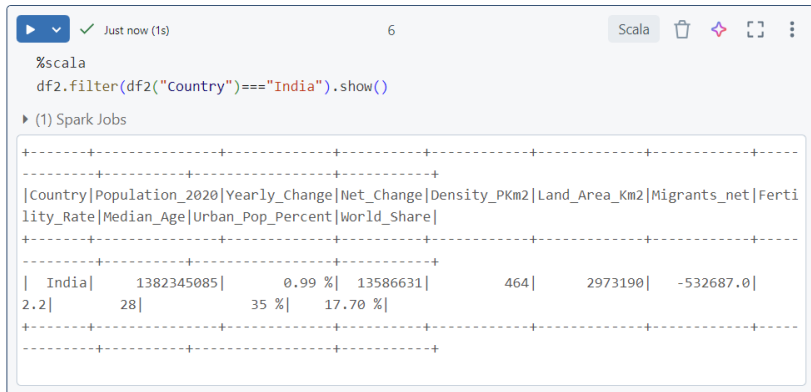
%scala
df2.select("Country", "Population_2020").show()

▶ (1) Spark Jobs
```

Country	Population_2020
China	1440297825
India	1382345085
United States	331341050
Indonesia	274021604
Pakistan	221612785
Brazil	212821986
Nigeria	206984347
Bangladesh	164972348
Russia	145945524
Mexico	129166028
Japan	126407422
Ethiopia	115434444
Philippines	109830324
Egypt	102659126
Vietnam	97490013
DR Congo	90003954
Turkey	84495243

## Lab 3: Using DataFrames in Azure Databricks

- To filter the row that contains the data for India, execute the filter function given below.



The screenshot shows a Databricks notebook interface. At the top, there's a toolbar with a play button, a green checkmark indicating the job is running, the text 'Just now (1s)', a cell count of '6', and icons for 'Scala', trash, a star, a full-screen icon, and a vertical ellipsis. Below the toolbar, the code cell contains the following Scala code:

```
%scala
df2.filter(df2("Country")==="India").show()
```

Below the code cell, the output is displayed under the heading '(1) Spark Jobs'. The output is a table with 8 columns: Country, Population\_2020, Yearly\_Change, Net\_Change, Density\_PKm2, Land\_Area\_Km2, Migrants\_net, and Fertility\_Rate. The table shows data for India.

Country	Population_2020	Yearly_Change	Net_Change	Density_PKm2	Land_Area_Km2	Migrants_net	Fertility_Rate
India	1382345085	0.99 %	13586631	464	2973190	-532687.0	2.2



# License



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).