ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Micro Project – Problem Statement

We have seen a Spark example earlier which gives us the following feel:

1. What do we do in the Spark application?
2. How does a typical Spark application code looks?
3. How to write and execute your Spark application?

Apache Spark is a data processing engine.
In a typical Spark application, we read data from a source, process it, and write or present the results.
And this course is all about learning how to do these three things - Read, Process and Write.
We have seen a sample code of the Spark application, where we learned how to write and execute Spark code.

It's time to start learning Spark Programming and develop your Spark coding skills. We will take a project-driven approach to learning things.
We have a micro project problem statement shown below.
I am giving you the Fire Calls-For-Service dataset.

Problem Statement

Data Set: Fire Department Calls for Service

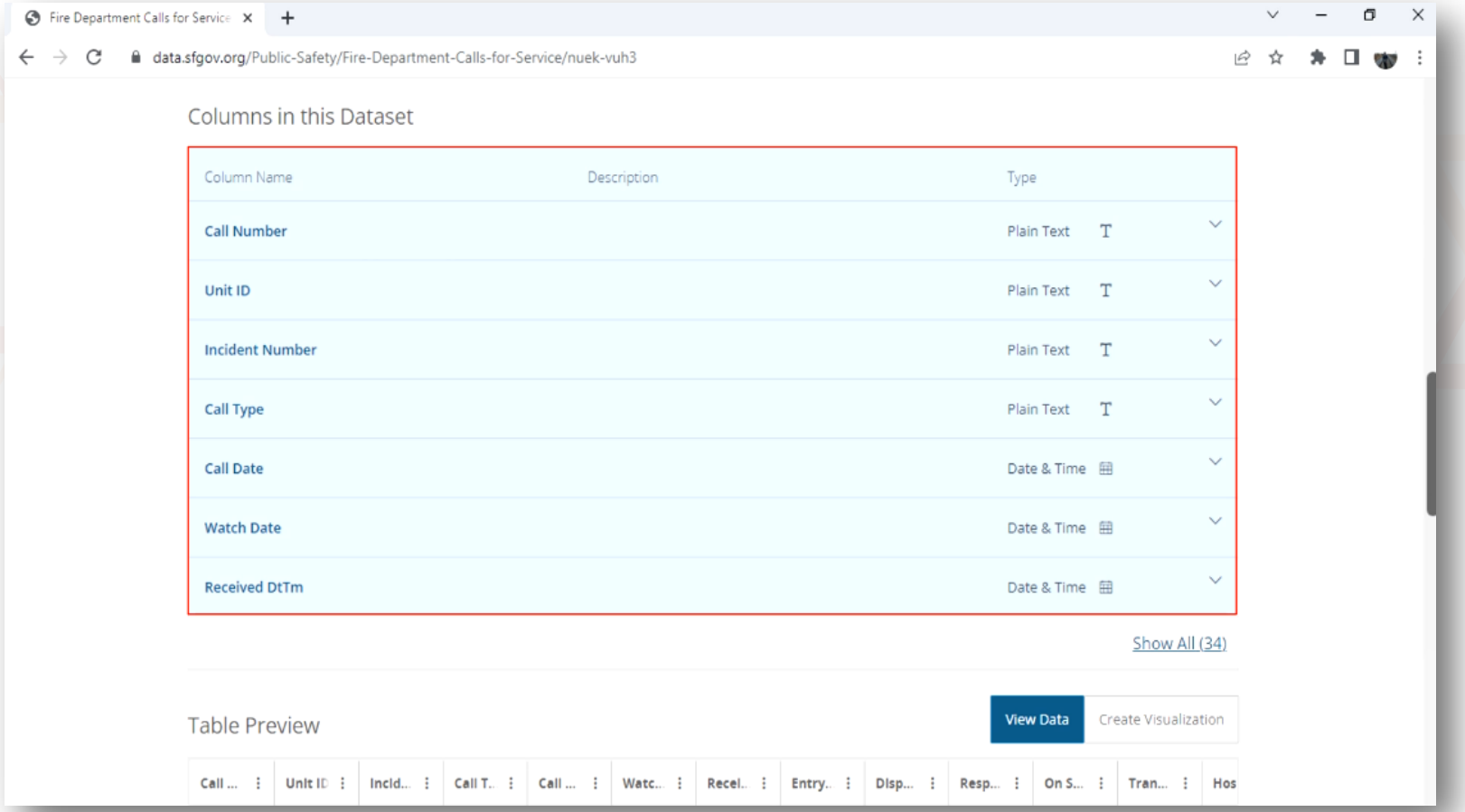
URL: https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

What are the requirements

1. Load the given data file and create a Spark data frame.
2. Use the Spark data frame to answer the following questions.
    1. How many distinct types of calls were made to the fire department?
    2. What are distinct types of calls made to the fire department?
    3. Find out all responses or delayed times greater than 5 mins?
    4. What were the most common call types?
    5. What zip codes accounted for the most common calls?
    6. What San Francisco neighborhoods are in the zip codes 94102 and 94103
    7. What was the sum of all calls, average, min, and max of the call response times?
    8. How many distinct years of data are in the CSV file?
    9. What week of the year in 2018 had the most fire calls?
    10. What neighborhoods in San Francisco had the worst response time in 2018?

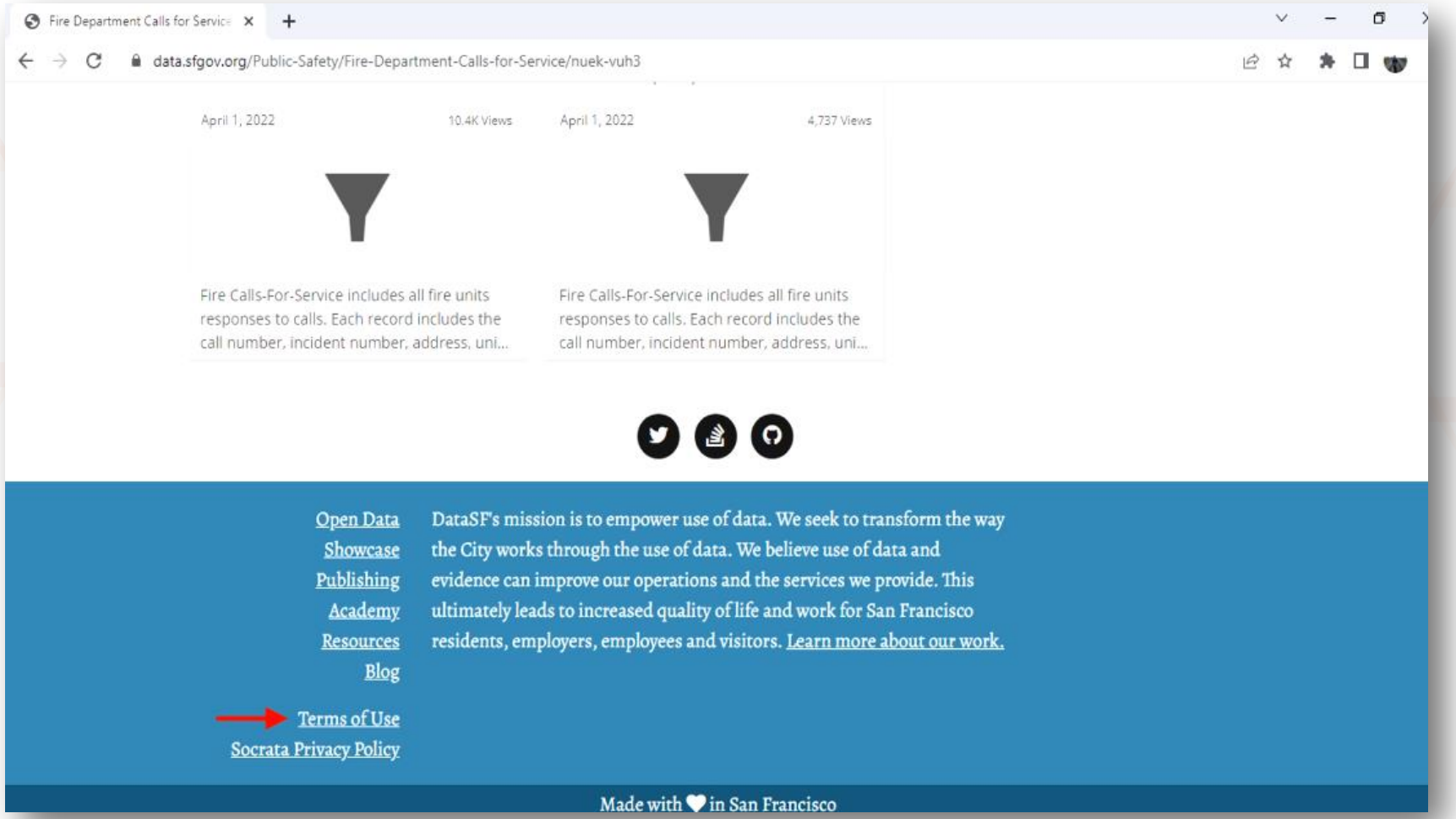
Reference: https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

This data set represents a response to calls at fire units. If we go to the above link, we can see the 34 column names available in this data set. You can also see the table preview in this page if you scroll down.

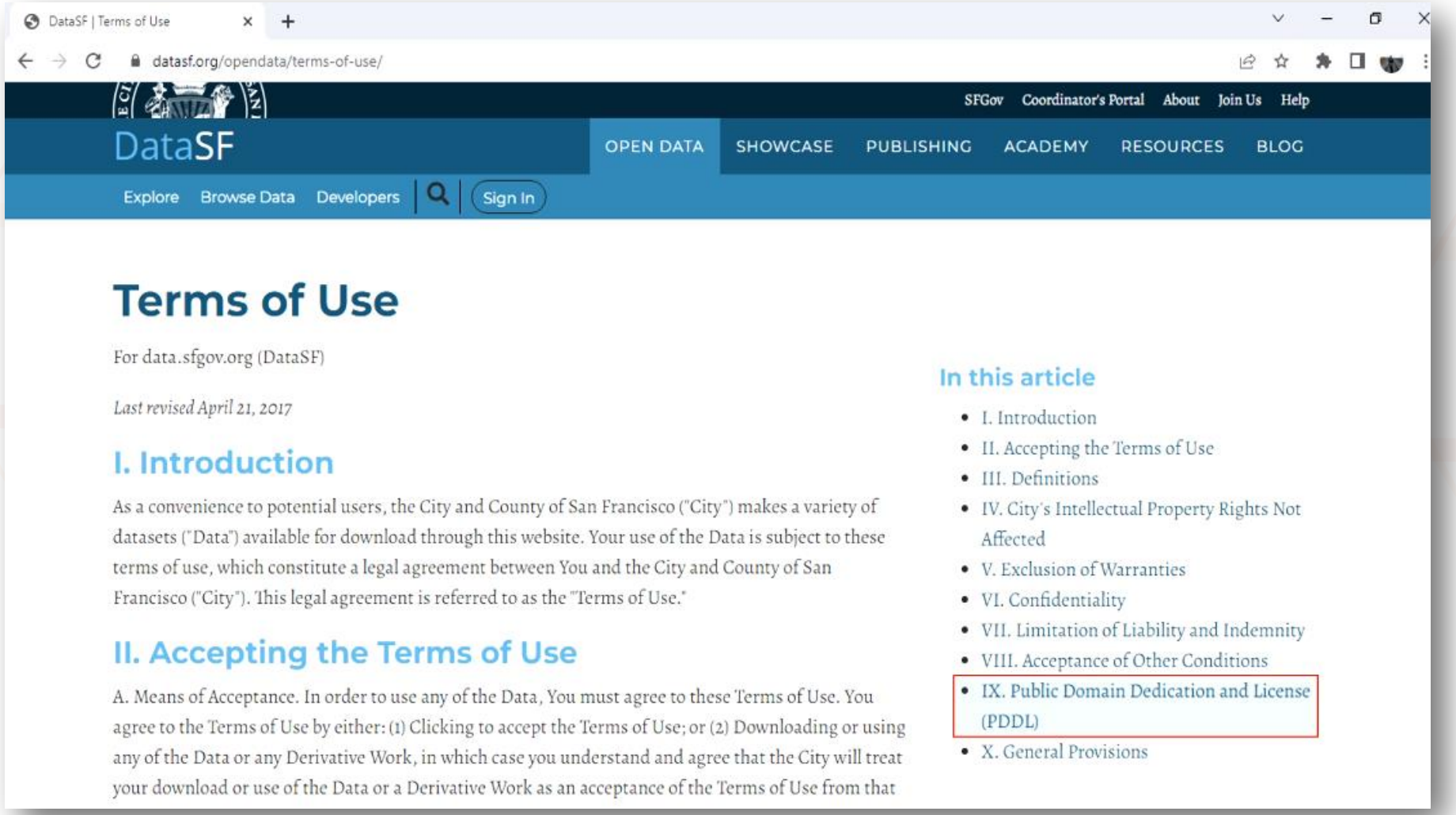


Reference: https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

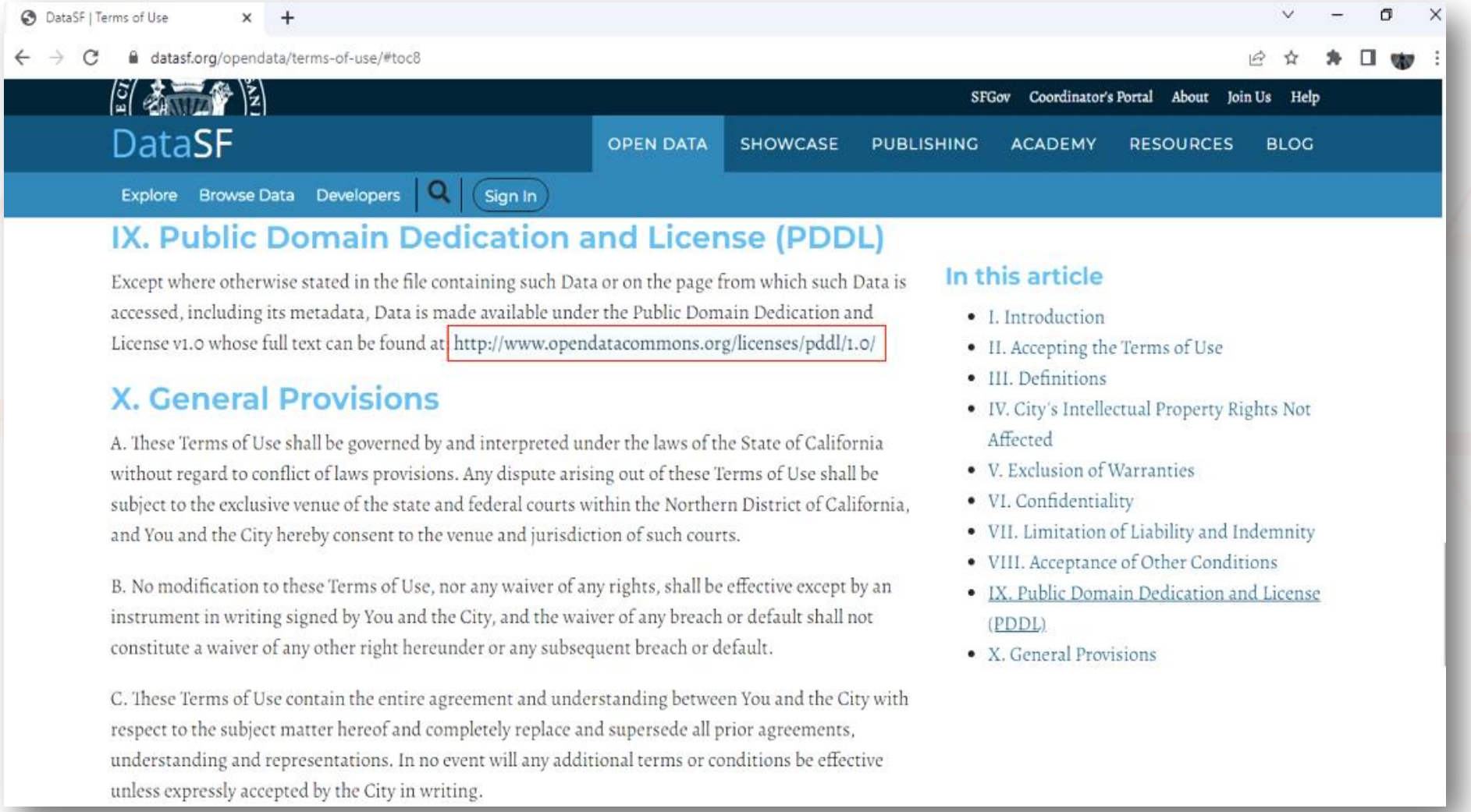
Scroll to the bottom of this page, you will see a link for terms of use, click that.



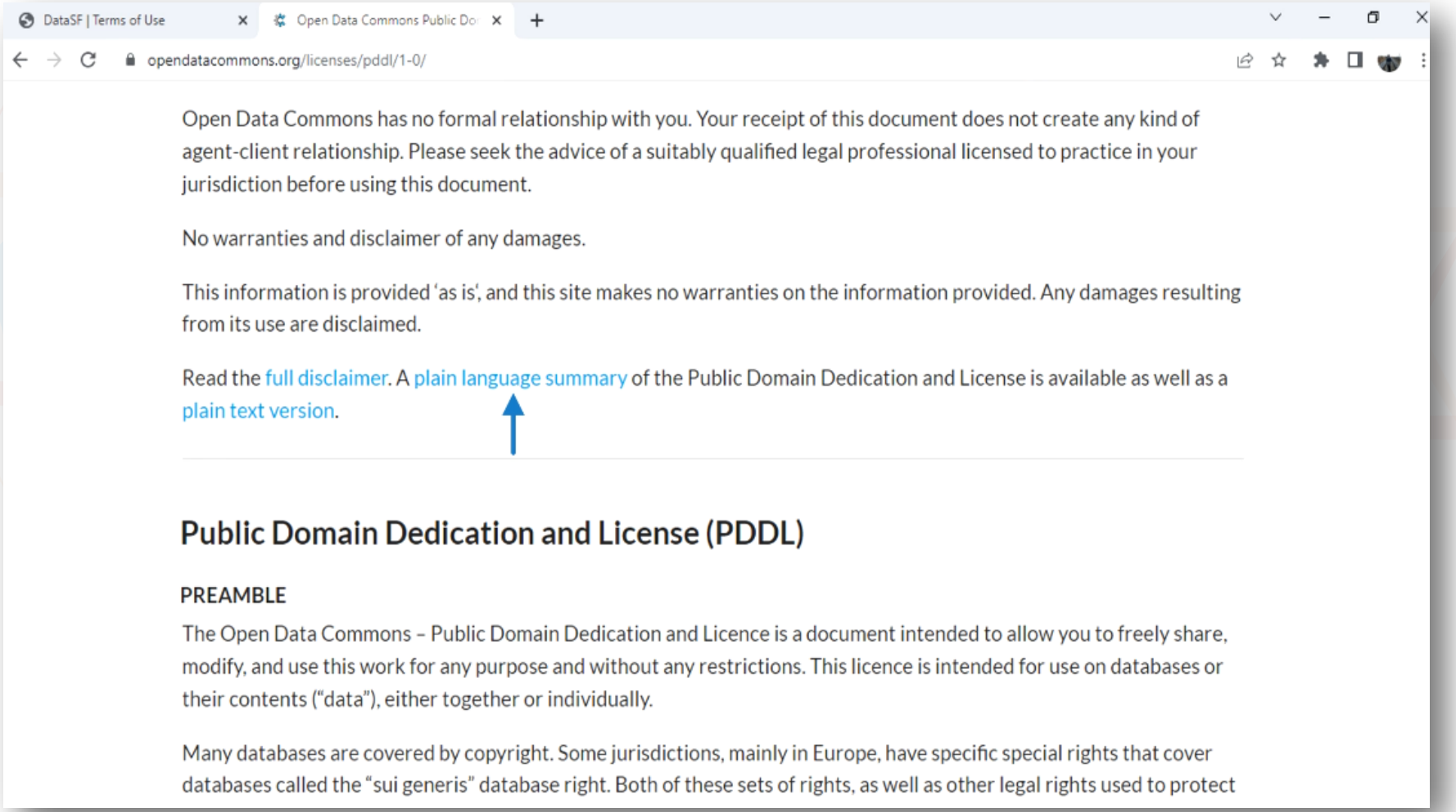
The data set is available under the PDDL terms. Click the PDDL link you see in the side panel.



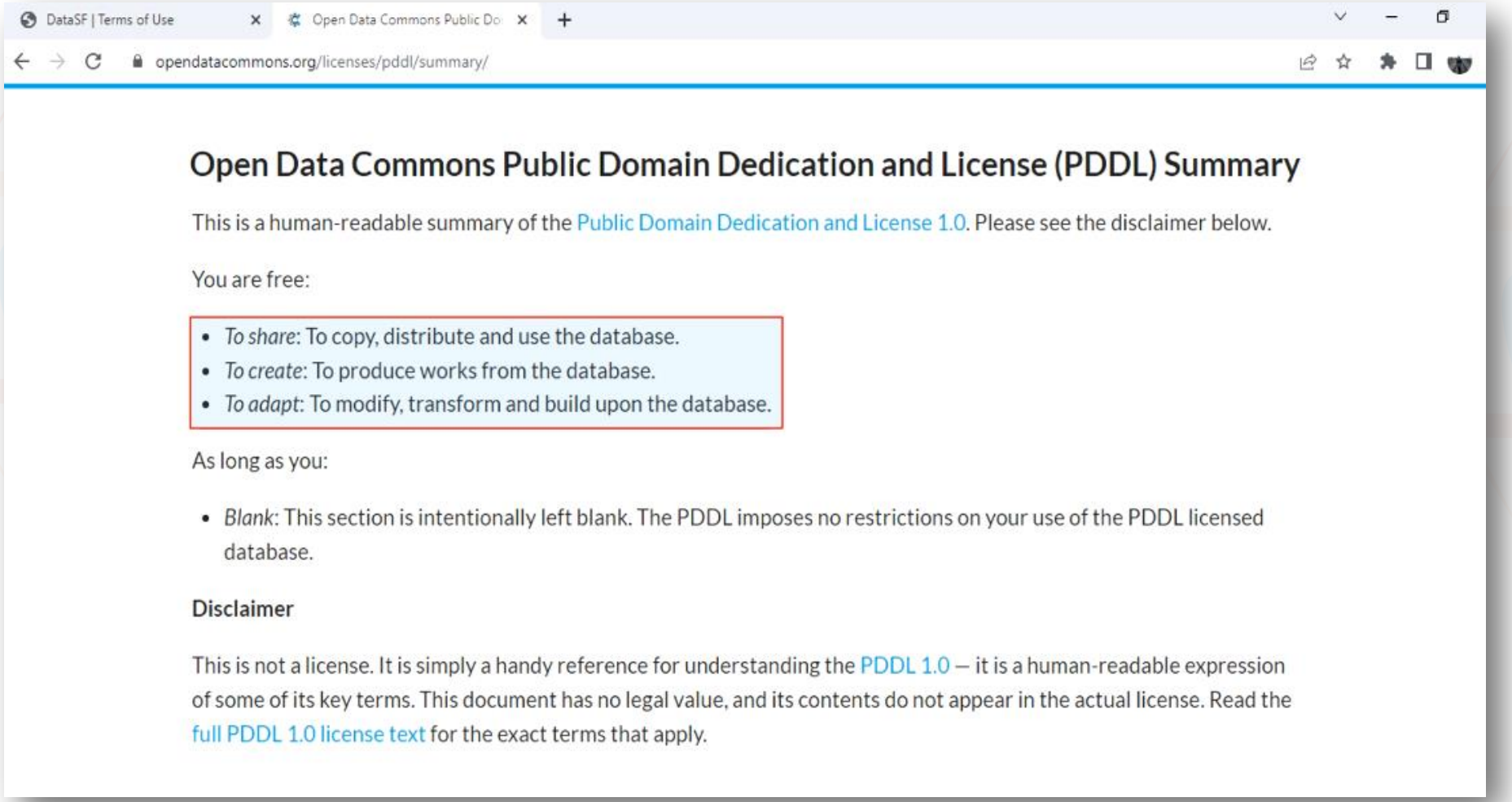
You will see the open data commons URL. Copy the URL and paste it into a new browser tab.



If you go to the open data commons URL, and scroll down, you will see a plain-language summary. Click that.



So you are free to share, create and adapt this data set. It is essential to check the terms of use for any data set that you are using for your learning and experiments.
I showed you how to check the license terms for the Fire Calls-For-Service dataset.



You are given a San Francisco fire call response data set. And you are asked to fulfil the following set of requirements

Problem Statement

Data Set: Fire Department Calls for Service

URL: https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

What are the requirements

1. Load the given data file and create a Spark data frame.
2. Use the Spark data frame to answer the following questions.
    1. How many distinct types of calls were made to the fire department?
    2. What are distinct types of calls made to the fire department?
    3. Find out all responses or delayed times greater than 5 mins?
    4. What were the most common call types?
    5. What zip codes accounted for the most common calls?
    6. What San Francisco neighborhoods are in the zip codes 94102 and 94103
    7. What was the sum of all calls, average, min, and max of the call response times?
    8. How many distinct years of data are in the CSV file?
    9. What week of the year in 2018 had the most fire calls?
    10. What neighborhoods in San Francisco had the worst response time in 2018?

We want to do only two things:

1. Load the data into a Spark Data frame
2. Query it to find the answers to the ten questions shown earlier.

The main objective of the Micro Project is to learn two things:

1. How to load data into Spark
2. How to query data in Spark.

We have two approaches to do it:

1. Using Spark SQL
2. Using Spark Dataframe API

While we work on the solution of the given 10 questions, we will learn the following concepts:

1. Using Spark SQL and DDL statements
2. Spark Data Frame and Data Frame Reader API
3. Working with CSV Files
4. Data Frame Transformations and Actins
5. How to read Spark API Documentation

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Spark Dataframes - Introduction

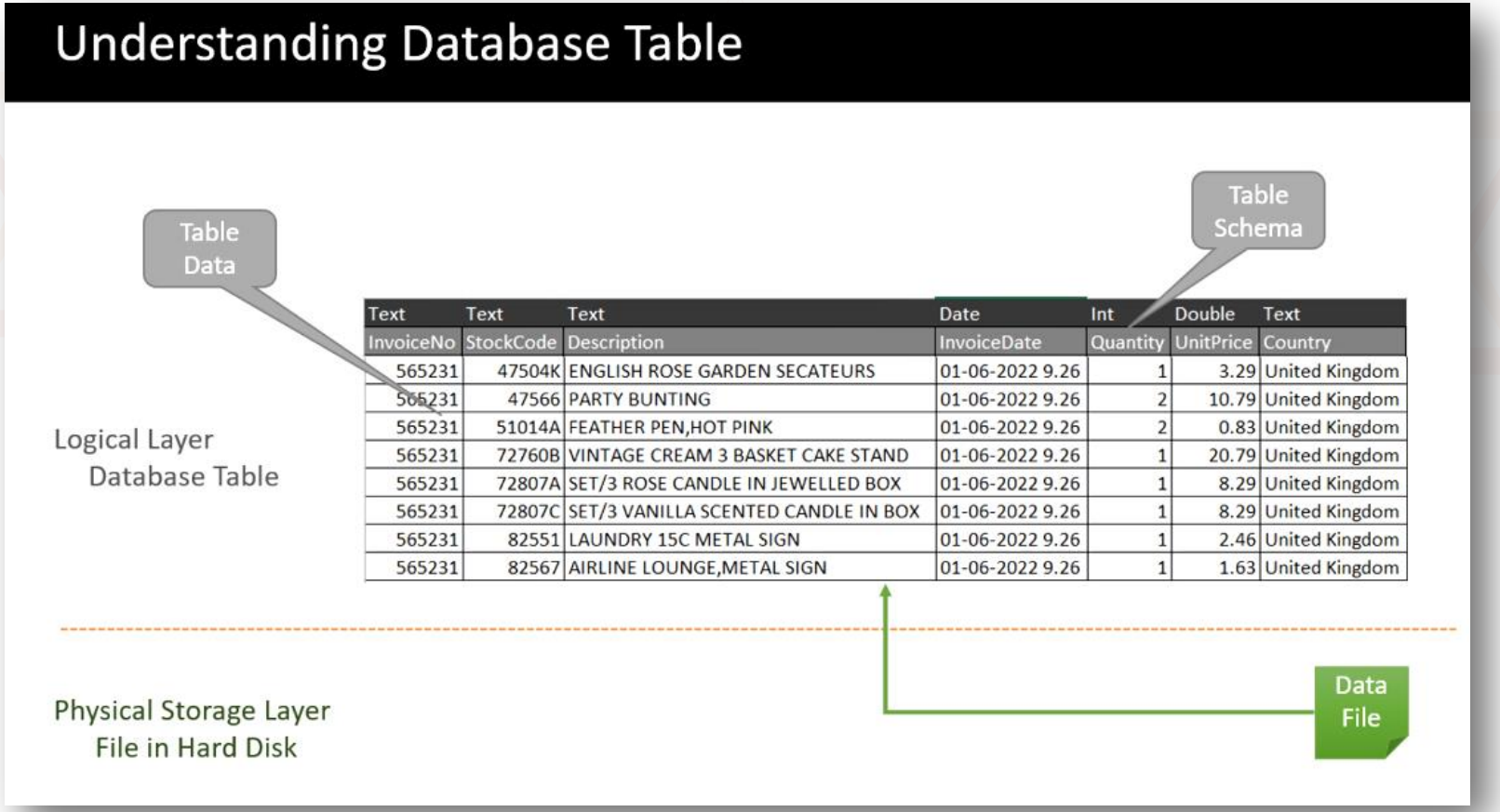
We learned that Spark is a data processing platform.
We also know that Databases are the most popular and most widely used data processing platforms.
They offer two things at a high level.

1. Tables
2. SQL

A database table allows you to load the data in the table. The data in a table is internally stored as a .dbf file, which are stored on the disk. But we don't care about this dbf file and storage layer, we always look at the table, and query the table.

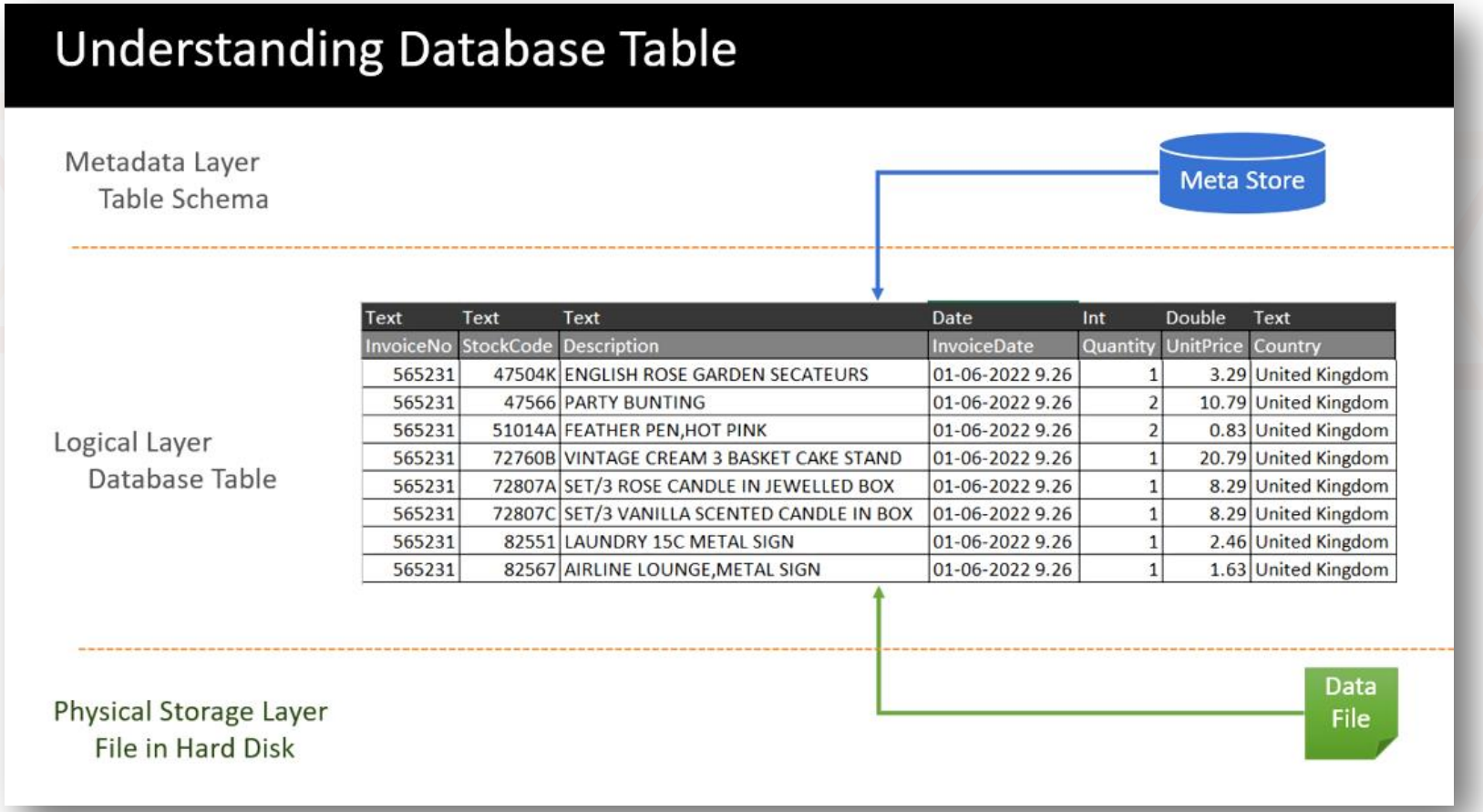
A table contains two things:

1. Table Schema – It is a list of column names and data types.
2. Table Data – It is the data stored inside the table.



We have three layers to form a table:

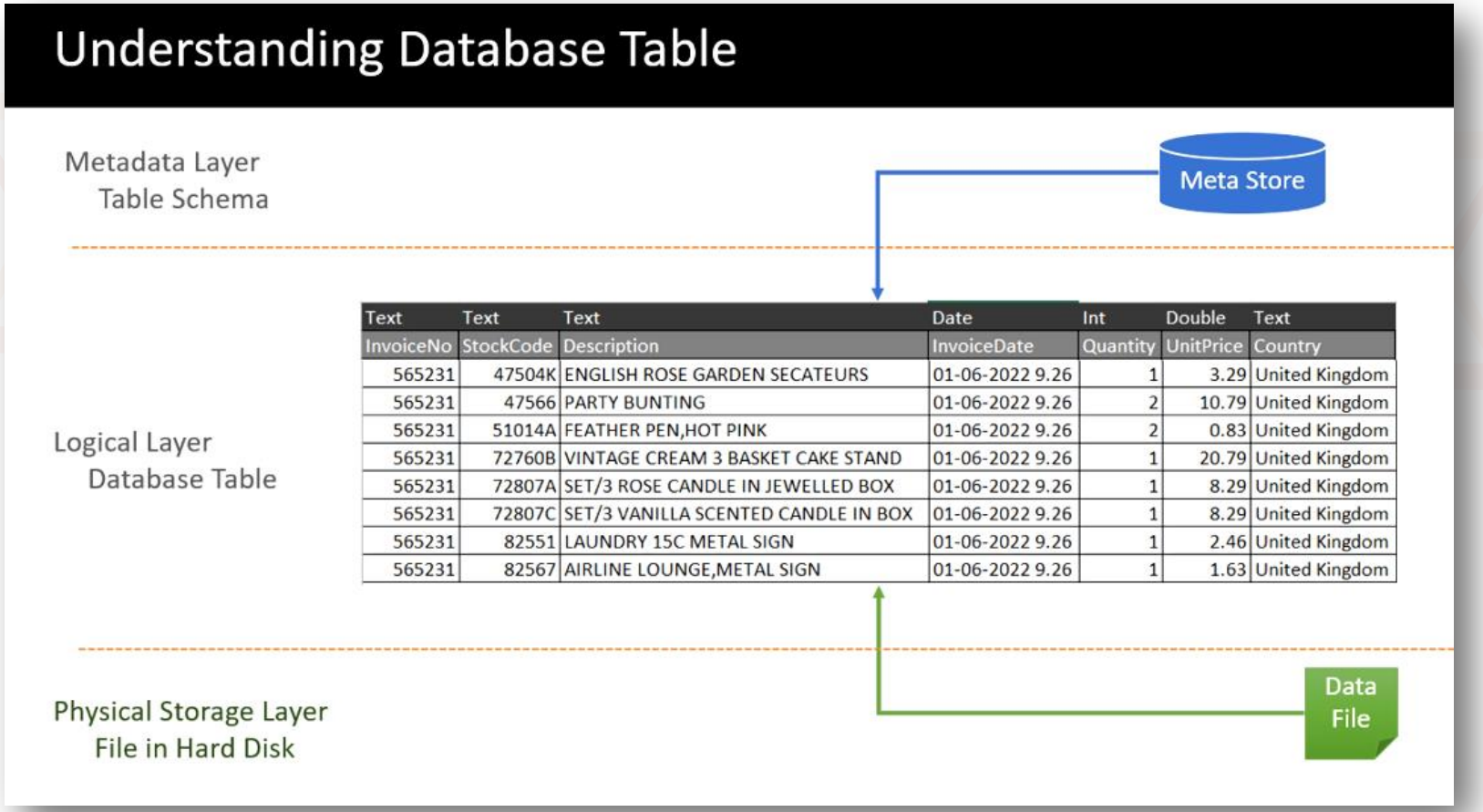
1. Storage layer stores the table data in a file
2. Metadata layer stores the table schema and other important information
3. The Logical layer presents you with a database table, and you can execute SQL queries on the logical table.



Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

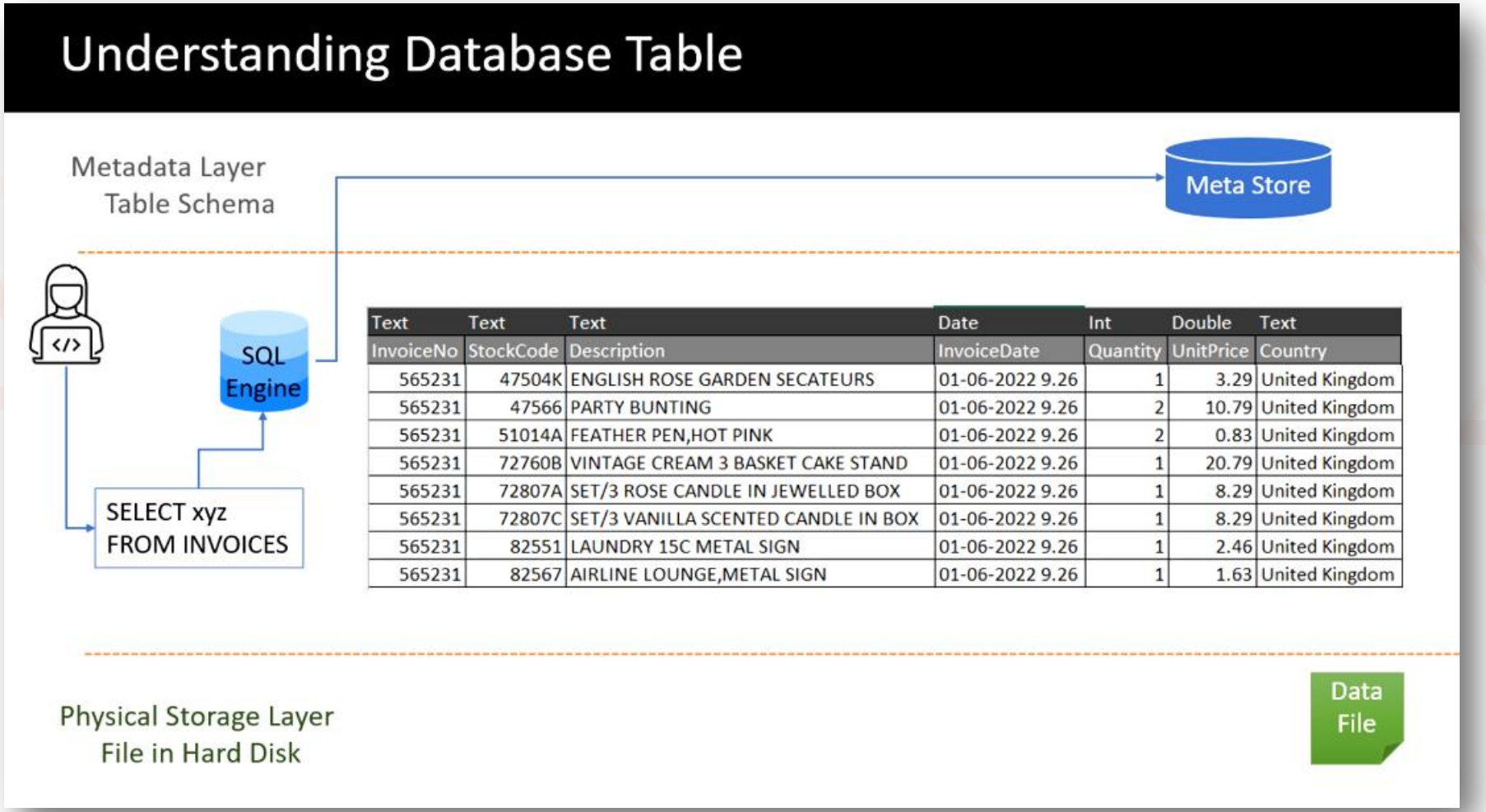
We have three layers to form a table:

1. Storage layer stores the table data in a file
2. Metadata layer stores the table schema and other important information
3. The Logical layer presents you with a database table, and you can execute SQL queries on the logical table.



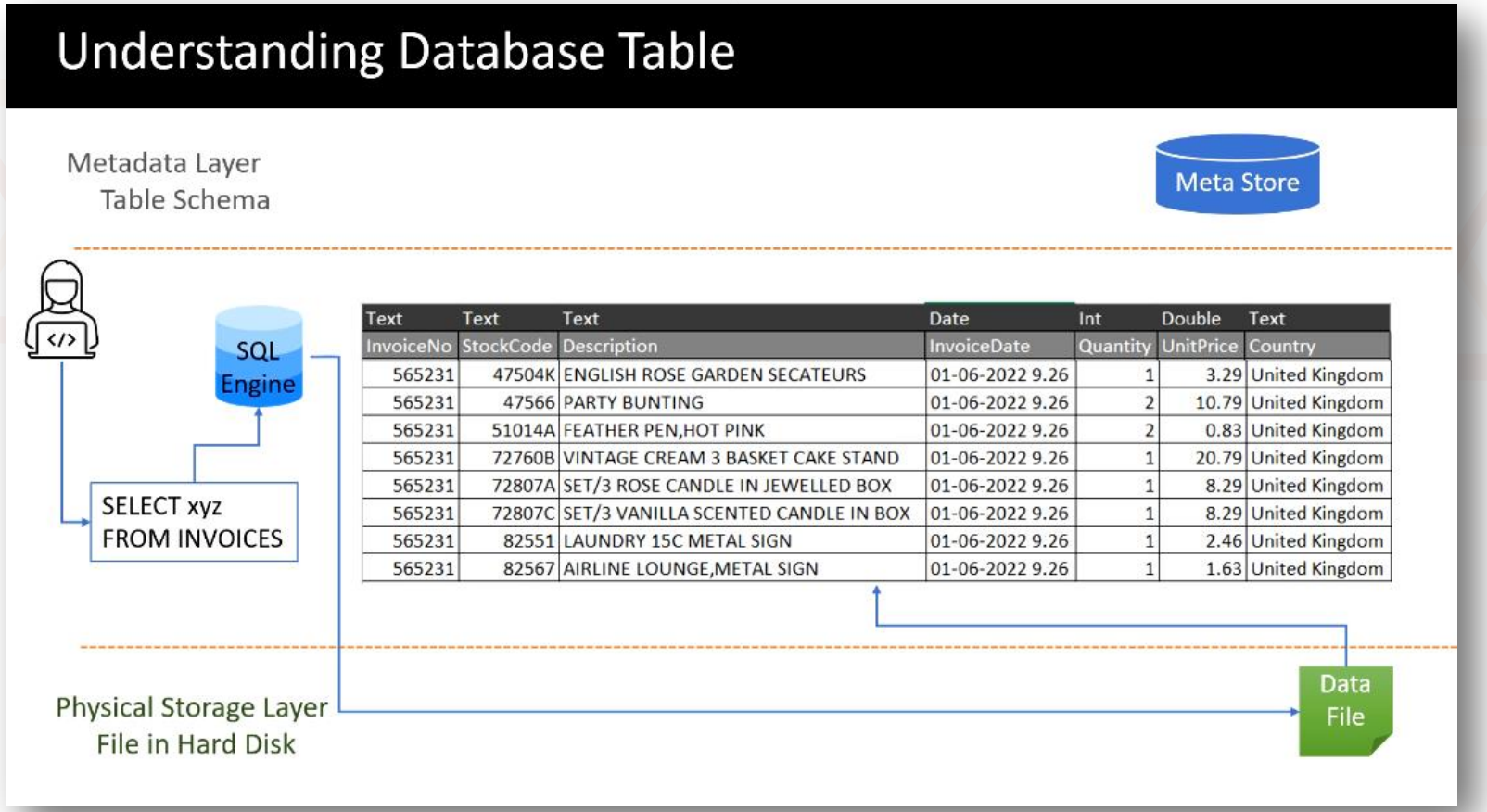
Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

When you submit a SQL query to your database SQL Engine, the database will refer to the metadata store for parsing your SQL queries. And the database will throw a syntax error or an analysis error if you are using a column name in your SQL that does not exist in the metadata store.



Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

The schema is essential for the database table and for the SQL expressions to work correctly. The table data is stored in the dbf files behind the table. So if your SQL query is correct and passes all the schema validation, you will see query results.
The database will read data from the "dbf" file, process it according to your SQL query, and show you the results. And that is all about the data processing in databases.



Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

Apache Spark offers you two ways of data processing:

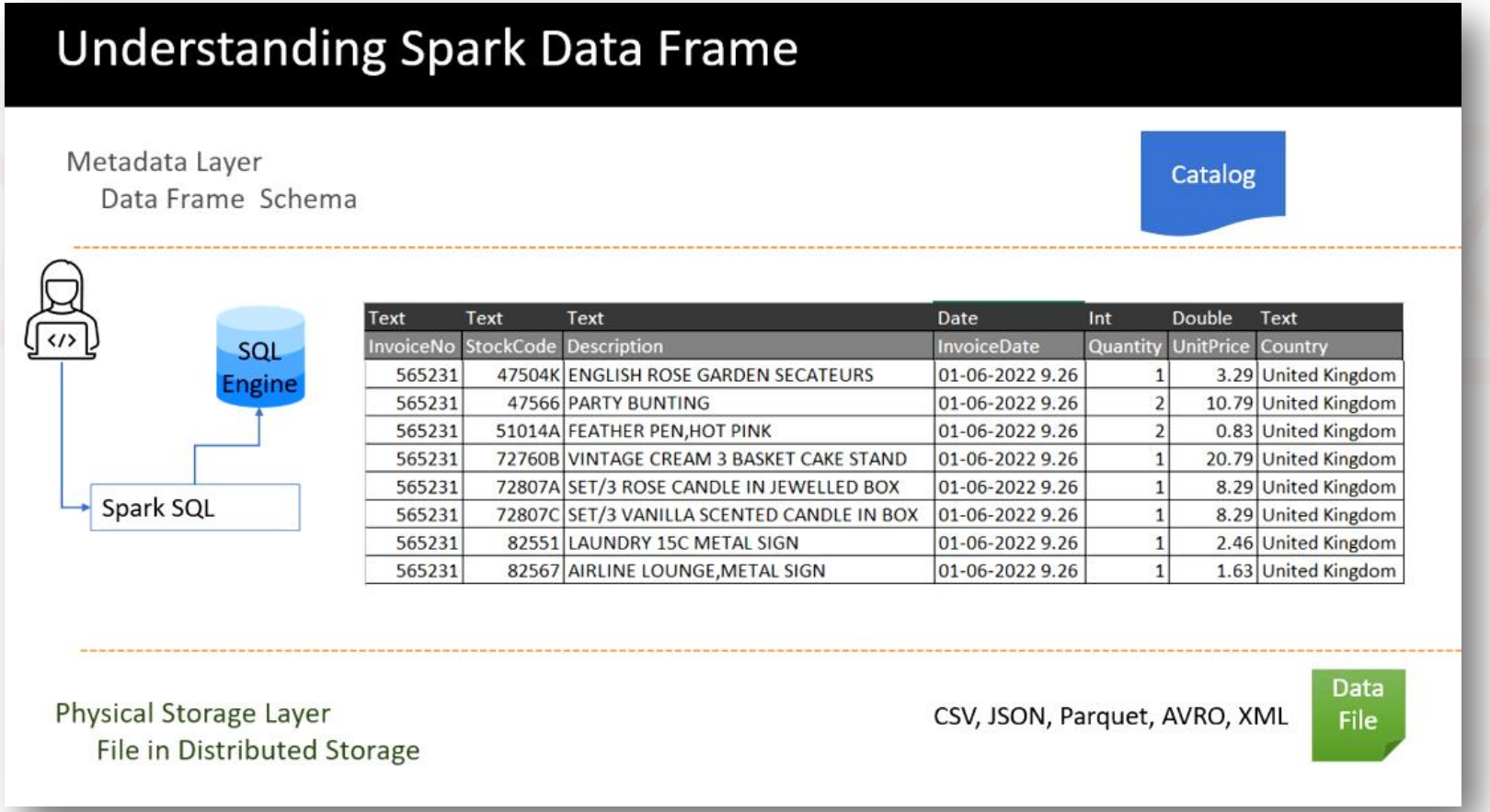
1. Spark Database and SQL
2. Spark Dataframe and Dataframe API

The first approach is precisely the same as a typical database. You will create table and load data into the table. Spark table data is internally stored in the data files. But these files are not dbf files like in the case of a database.
Spark gives you the flexibility to choose the file format and supports many file formats such as CSV, JSON, Parquet, AVRO and XML.
The DBF file format and database engine were designed to support structured data. However, Spark supports structured, semi-structured, and unstructured data.

The spark storage layer also supports distributed storage such as HDFS and Cloud storage such as Amazon S3 and Azure ADLS. So you are not limited to disk storage capacity. You can use distributed storage and store large data files. Spark also has a metadata store for storing table schema information. So that part is similar to the databases.

Then Spark also comes with an SQL query engine and supports standard SQL syntax for processing and querying data from Spark tables.

The second approach to data processing in Apache Spark is **Spark Dataframe and Dataframe API**. Spark Dataframe is structurally the same as the table. However, it does not store any schema information in the metadata store. Instead, we have a runtime metadata catalog to store the Dataframe schema information. It is similar to the metadata store, but Spark will create it at the runtime to store schema information in the catalog.



Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

We have two reasons for storing schema information in the catalog.

1. Stark Dataframe is a runtime object – You can create a Spark Data frame at runtime and keep it in memory until your program terminates. Once your program terminates, your Dataframe is gone. So, it is an in-memory object.
2. Spark Dataframe supports schema-on-read – Dataframe does not have a fixed and predefined schema stored in the metadata store. We load the data into a Dataframe and tell the schema when loading the data. And Spark will read the file, apply the schema at the time of reading, create the Dataframe using the schema and load the data.

We have two reasons for storing schema information in the catalog.

1. Stark Dataframe is a runtime object – You can create a Spark Data frame at runtime and keep it in memory until your program terminates. Once your program terminates, your Dataframe is gone. So, it is an in-memory object.
2. Spark Dataframe supports schema-on-read – Dataframe does not have a fixed and predefined schema stored in the metadata store. We load the data into a Dataframe and tell the schema when loading the data. And Spark will read the file, apply the schema at the time of reading, create the Dataframe using the schema and load the data.

Here is a quick summary discussing the differences between Spark Tables and Dataframes.

Spark Table Vs Data Frame

Spark Table

1. Tables store schema information in metadata store
2. Table and metadata are persistent objects and visible across applications
3. We create tables with a predefined table schema
4. Table supports SQL Expressions and does not support API

Spark Data Frame

1. Data Frame stores schema information in runtime Catalog
2. Data Frame and Catalog are runtime objects and live only during the application runtime. Data Frame is visible to your application only.
3. Data Frame supports schema-on-read
4. Data Frame offers APIs and does not support SQL expressions

Spark Table and a Data Frame are convertible objects

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Creating Spark Dataframe

You have already seen the requirement for creating a Dataframe using the San Francisco fire call response data set. The data set is already available in the Databricks community cloud, and it is made available for you by the Databricks for learning purposes. You can access it from the following location in your Databricks community cloud. There is no need to download the data from the source and bring it to the cloud. That part is already done for you.

Problem Statement

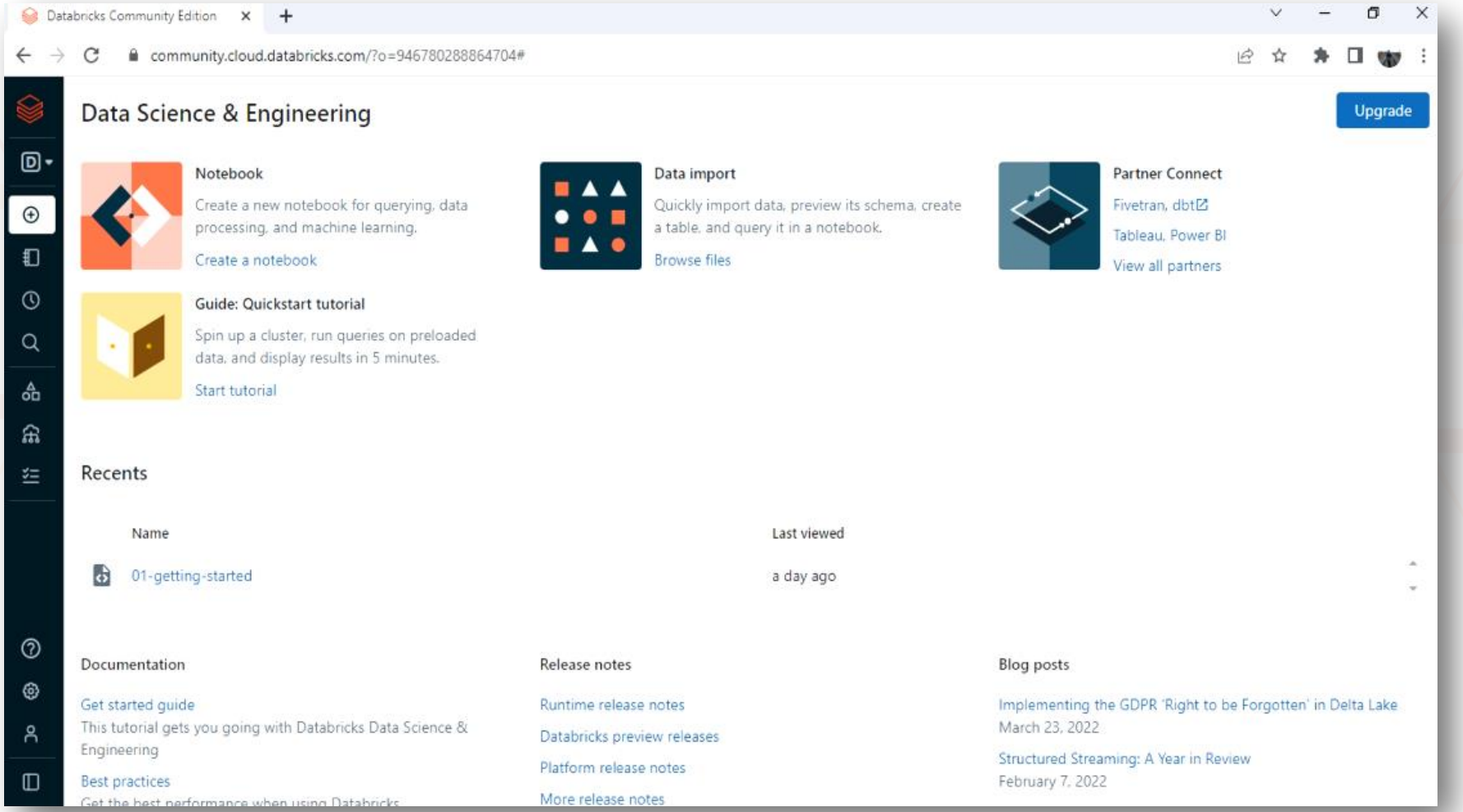
Data Set: Fire Department Calls for Service

Location: /databricks-datasets/learning-spark-v2/sf-fire/sf-fire-calls.csv

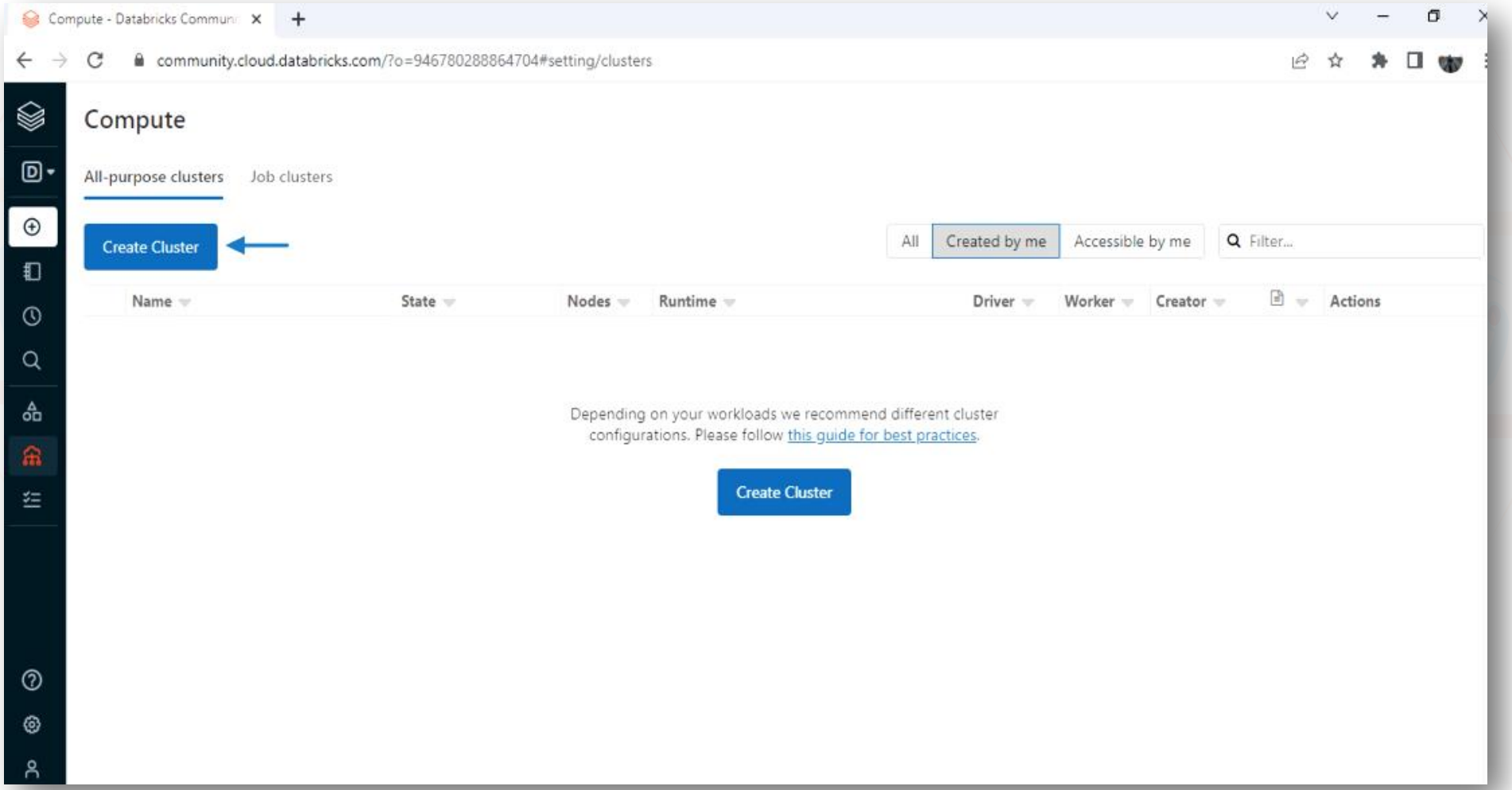
What are the requirements

1. Load the given data file and create a Spark data frame.
2. Use the Spark data frame to answer the following questions.
    1. How many distinct types of calls were made to the fire department?
    2. What are distinct types of calls made to the fire department?
    3. Find out all responses or delayed times greater than 5 mins?
    4. What were the most common call types?
    5. What zip codes accounted for the most common calls?
    6. What San Francisco neighborhoods are in the zip codes 94102 and 94103
    7. What was the sum of all calls, average, min, and max of the call response times?
    8. How many distinct years of data are in the CSV file?
    9. What week of the year in 2018 had the most fire calls?
    10. What neighborhoods in San Francisco had the worst response time in 2018?

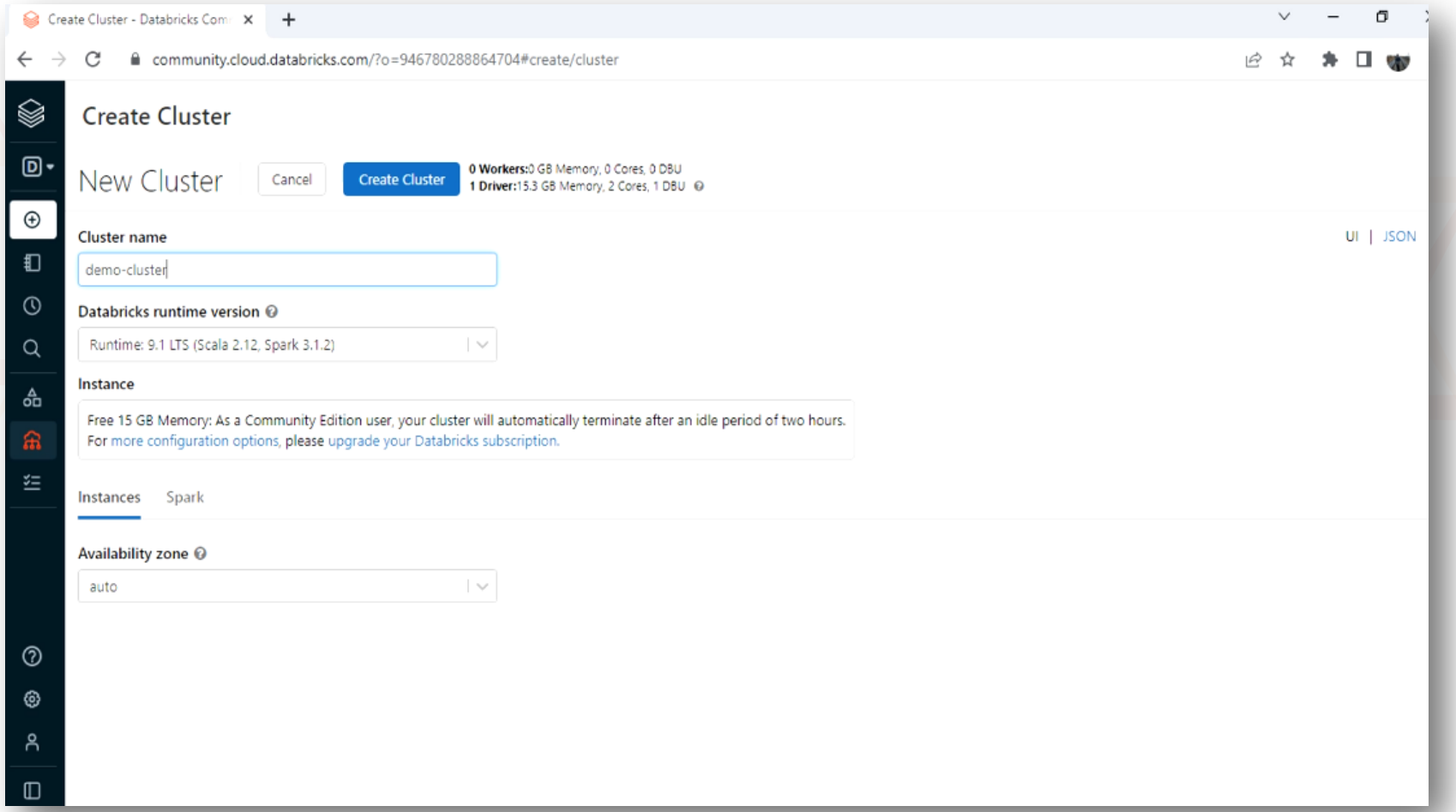
Start your Databricks Workspace.



Go to the Compute menu and create a new cluster.



Give a name to your cluster and create it. Your cluster will start in a few minutes, and Databricks will terminate it if your cluster is idle for 2 hours. Once terminated, you cannot start it again.
So if you want to work the next day, you should delete the older terminated cluster and create a new one. You can use your cluster for as long as you want and delete it when you are done.

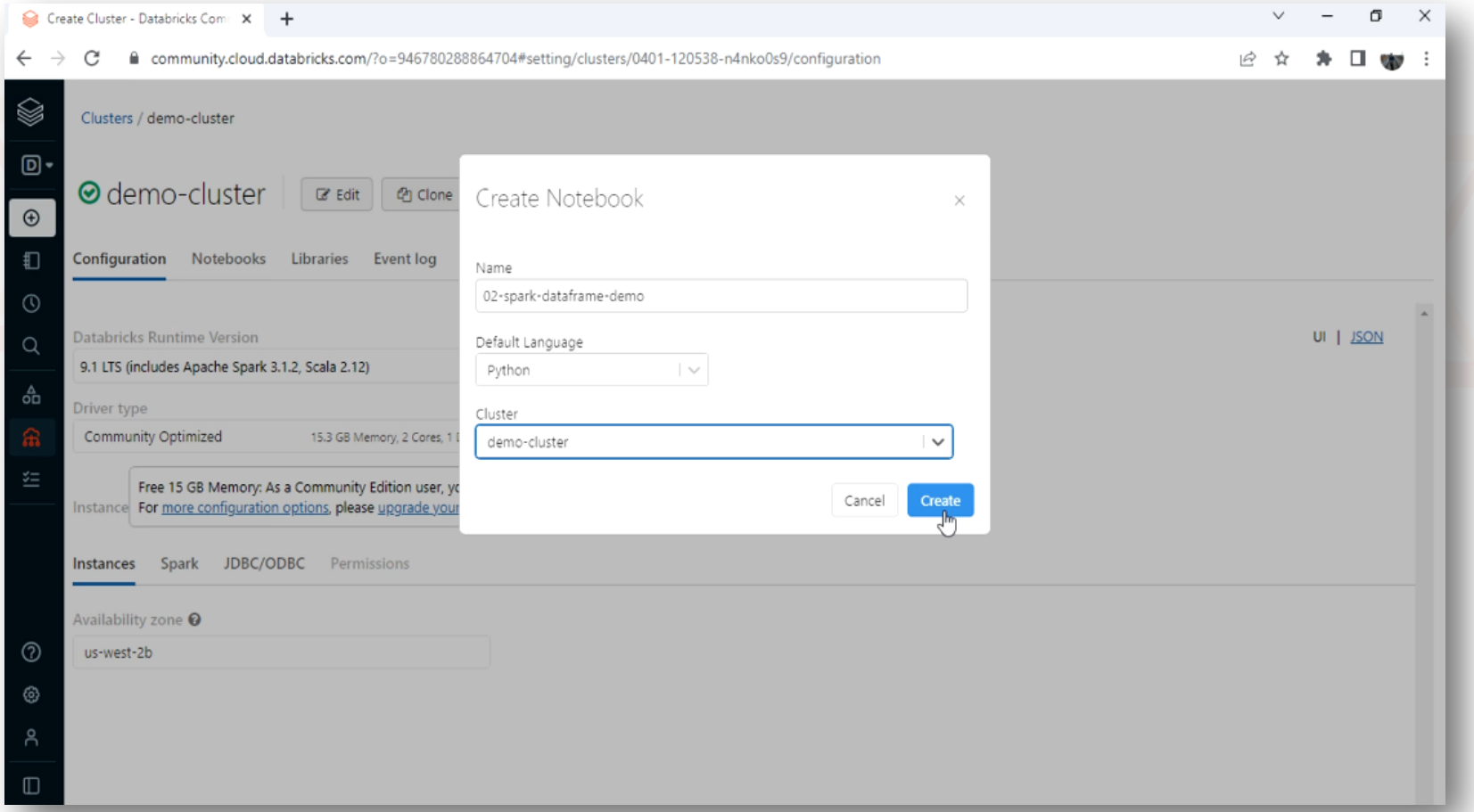


Once your cluster is up and running, you can go ahead to the "*create*" menu options and create a new notebook.

Give a name to the notebook. Every notebook should have a default language.

You have four language options: Python, Scala, SQL and R.

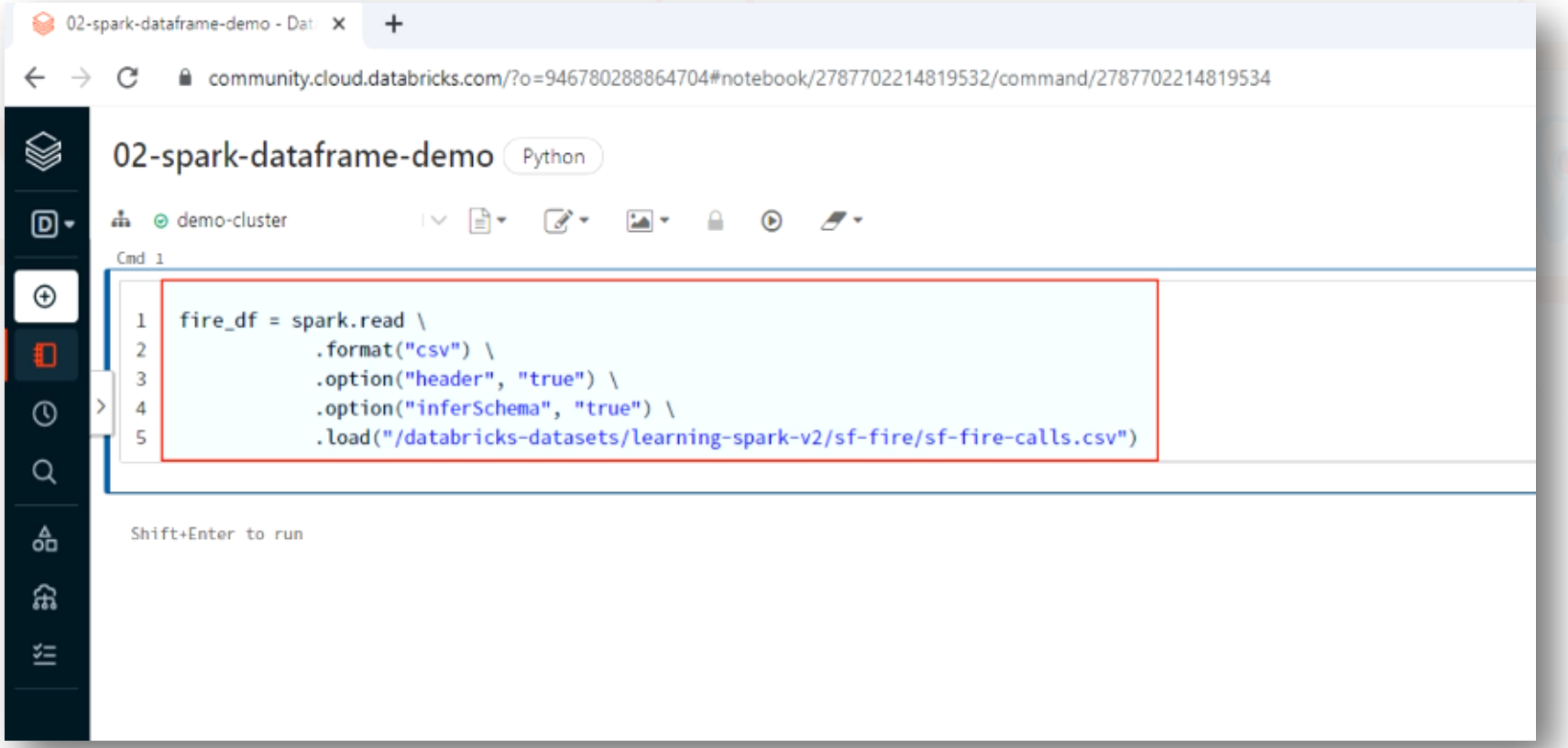
We want Python for Spark programming, so keep the default language Python. Your notebook will not run without a cluster. So you should also attach your cluster with the notebook. And finally, hit the create button.



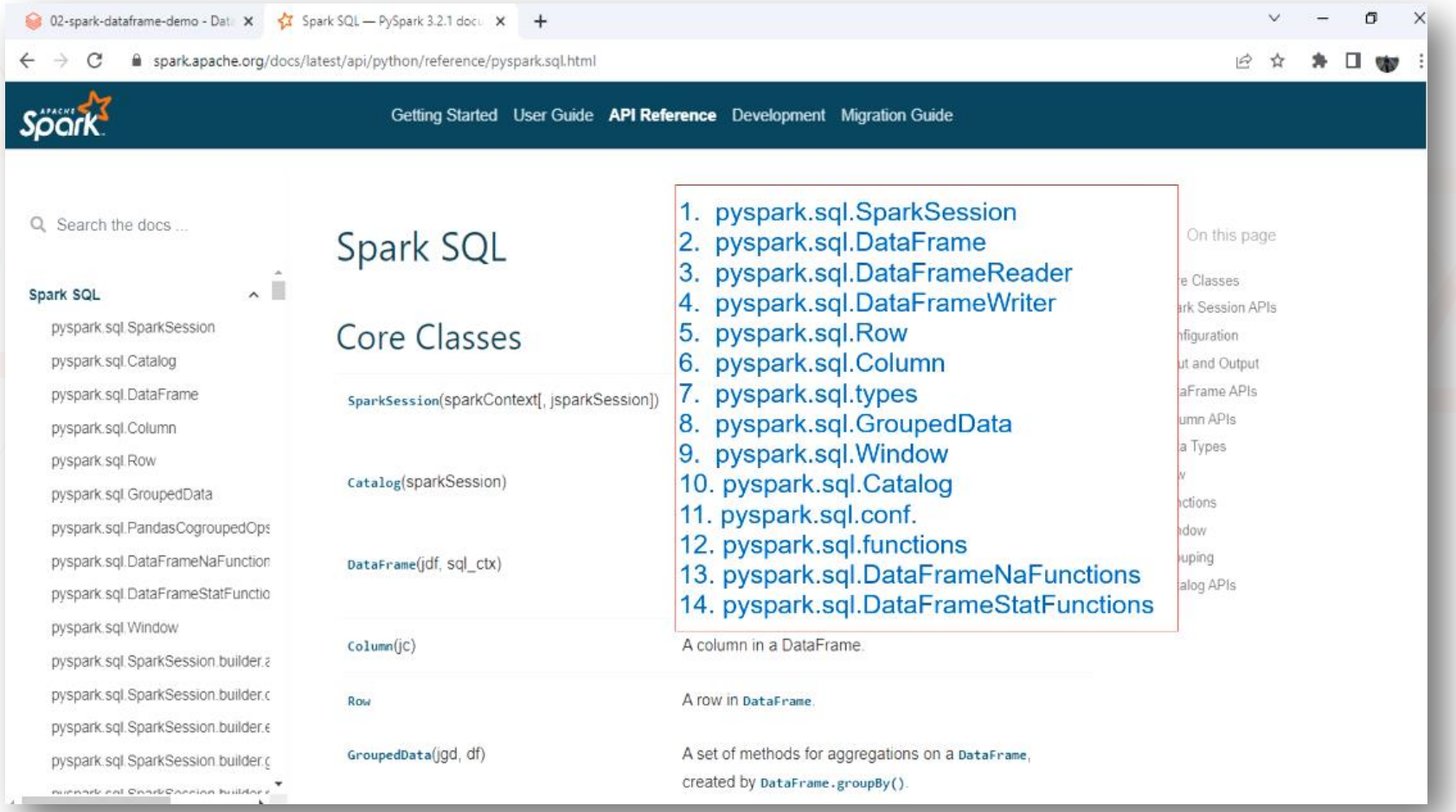
Once the notebook is created, you can write the code to create a Spark Dataframe using the San Francisco fire call response data set.

And here we have the code for the same shown below. **Reference: (02-spark-dataframe-demo.ipynb)**

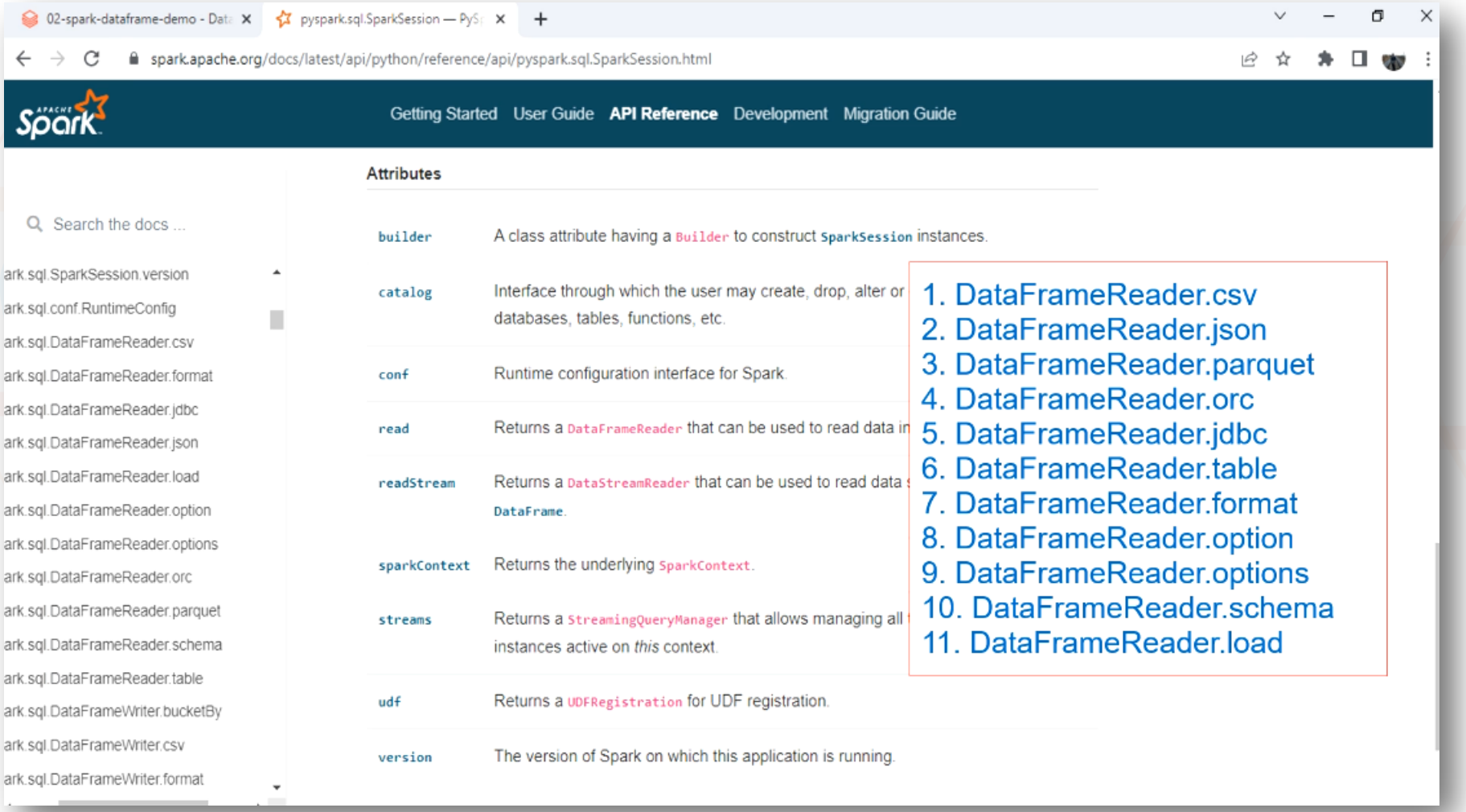
I am using spark.read which is a Spark Session object. It is the entry point for the Spark programming APIs. The rest of the code, such as format, option, and load, are the methods of the DataframeReader. I am telling the DataframeReader that the data file is a CSV file. Then I am telling that the CSV file comes with a header row. I am also saying that the DataframeReader should infer schema from the file itself. Finally, I call the load method so the DataframeReader can load the file from the given location. The code will return a dataframe, and I am assigning the dataframe to the fire_df variable.



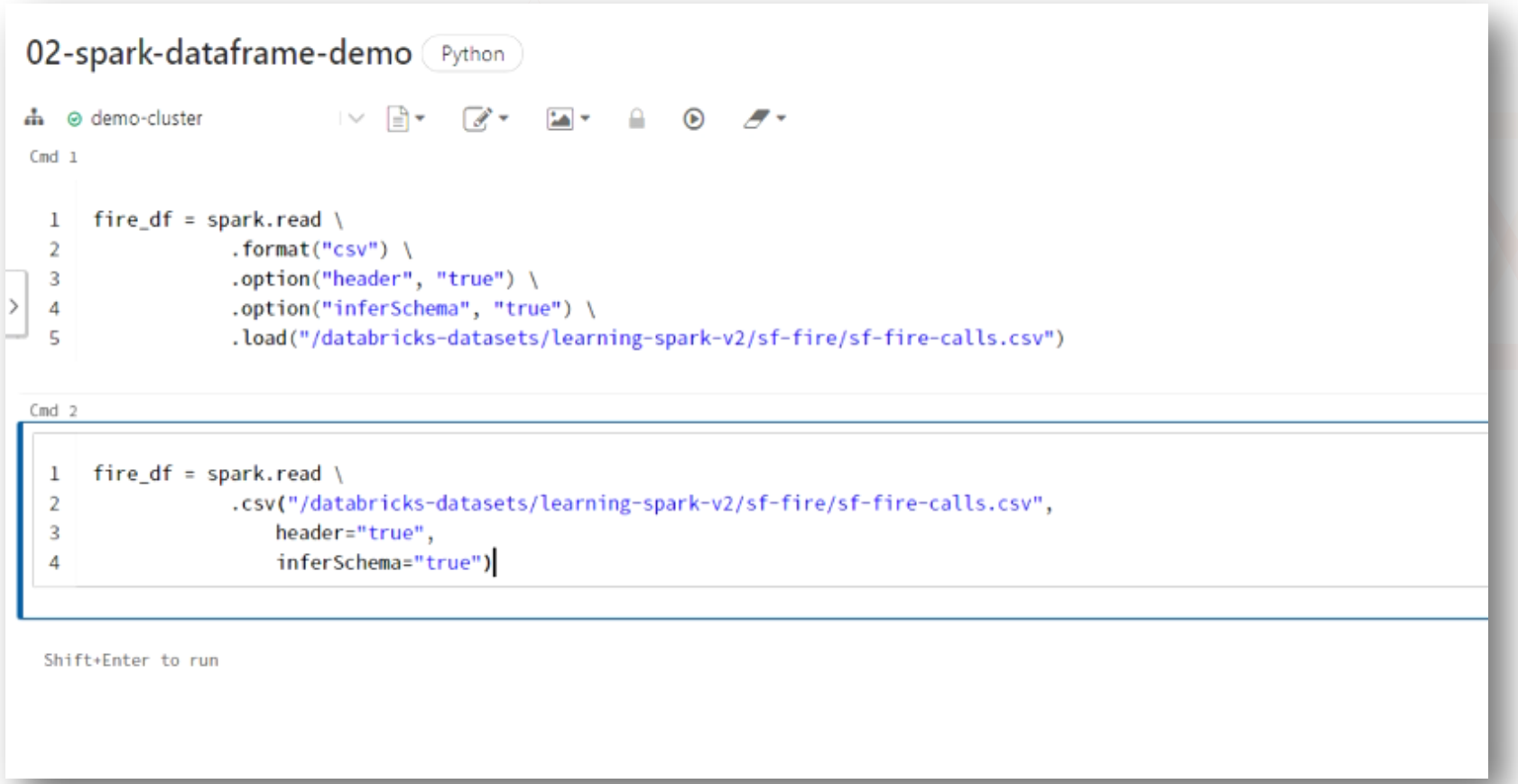
Visit https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html to go to the Spark SQL documentation. You will see that all the things listed in this page belongs to the 14 objects in the pyspark.sql package which are listed in the image below. Each object offers some attributes and methods.



Visit https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html to go to the DataFrameReader object under the Spark SQL documentation. The DataFrameReader offers eleven methods as shown in the screenshot below.

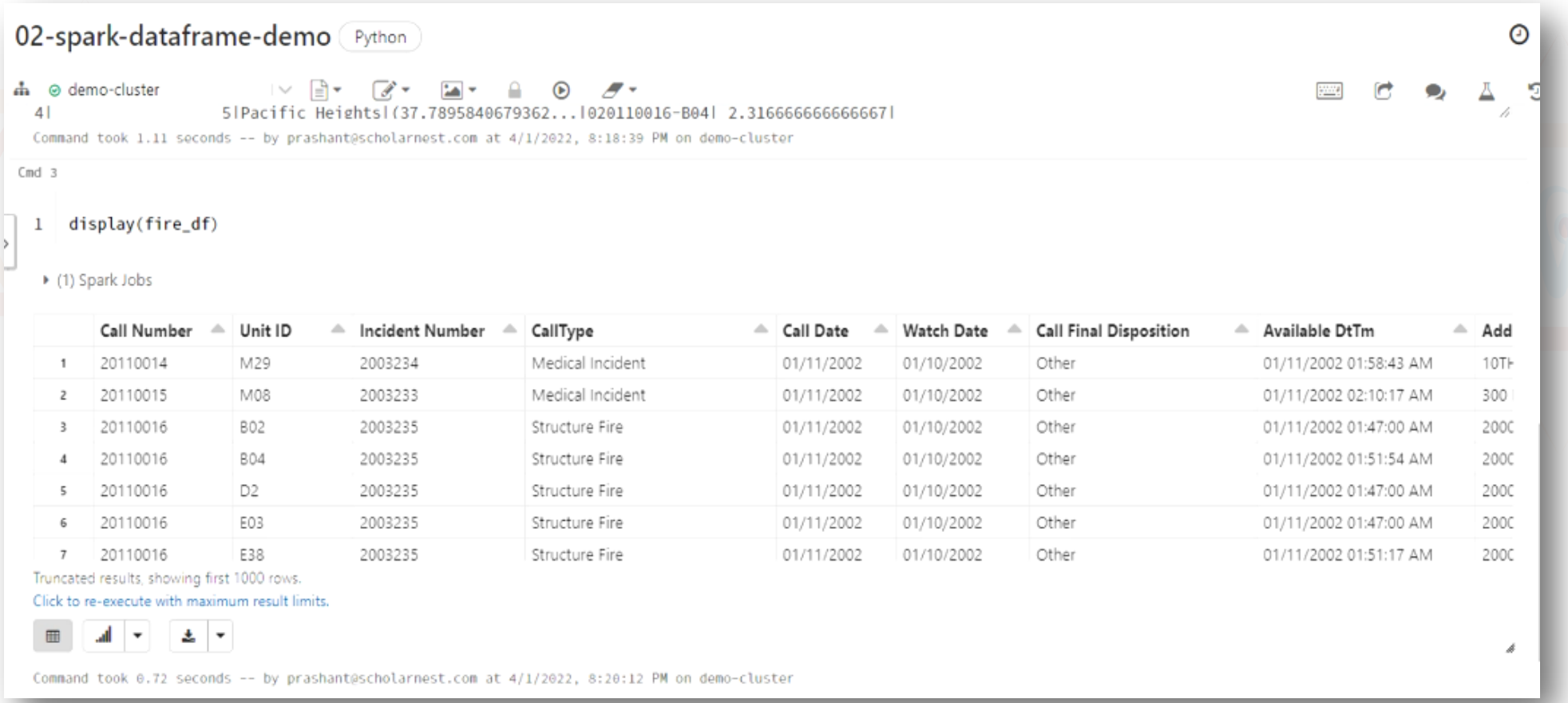


We can also create the same Dataframe using CSV method as shown in the second cell of the screenshot.
In the first cell, I am using the format method to tell that I want to load a CSV file and the option method to set other configurations.
In the second cell, we are using the shortcut CSV method. I am not setting the format and options.
The CSV method implicitly tells that the format is CSV, and all other details will go as function arguments.
So, both the code are doing the same thing, but I prefer the first one as this code is standardized for loading data from any file format.

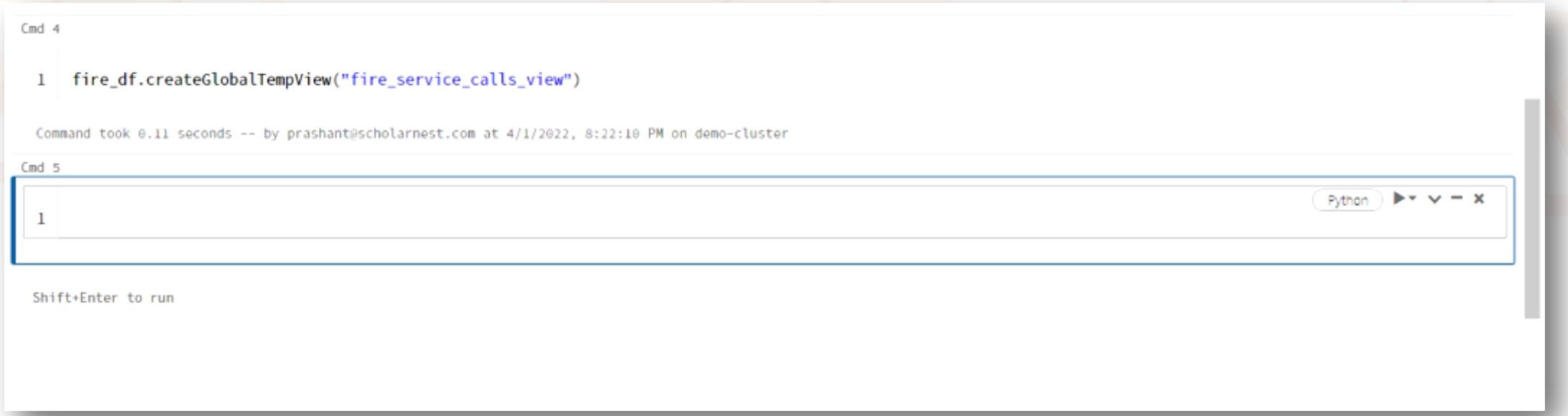


I created a DataFrame, and now I want to see some records from this dataframe. You can use the show method. I take this data frame variable and call the Dataframe show method. I want to see ten records, so I use show(10). And you can see the output below the code cell. But it doesn't look nice and easy to read.

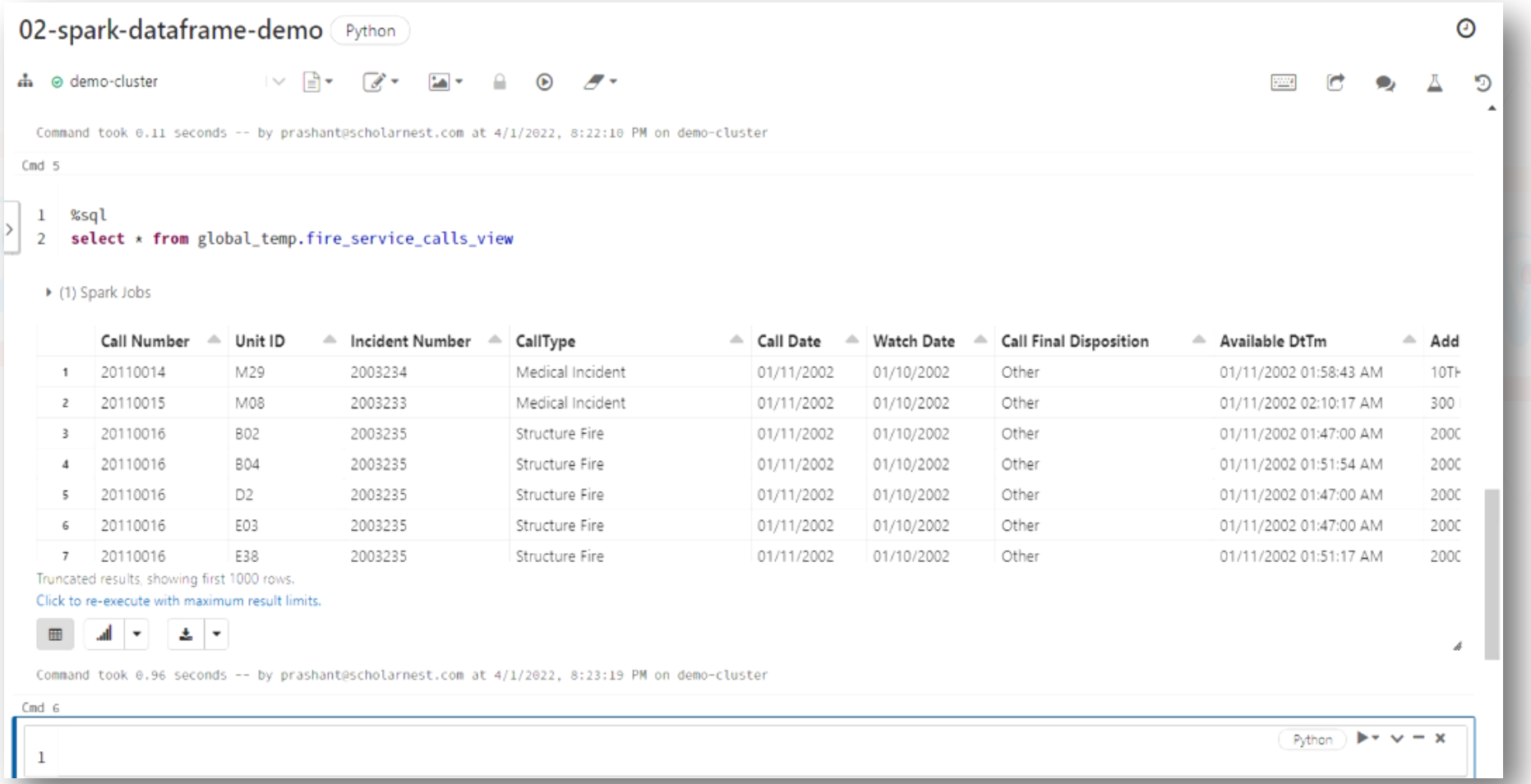
So, you can use display() function. And now the output looks much better and easy to read. The display function is not part of Apache Spark. It is an additional tool by Databricks to format the output of a DataFrame. So it takes a Dataframe and shows the content of the data frame.



In the code shown below, I am taking my dataframe and creating a global-temporary view for the dataframe. The view name is *fire_service_calls_view*. I am not converting my Dataframe to a table. Instead, I am creating a global-temporary view that is similar to the table.



You can run SQL expression on the global-temporary view. But you need to change the cell type from Python to SQL. The global_temp is a hidden database, and all global temporary tables are created in this global_temp database. So you must use global_temp for accessing a global-temporary view.



Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

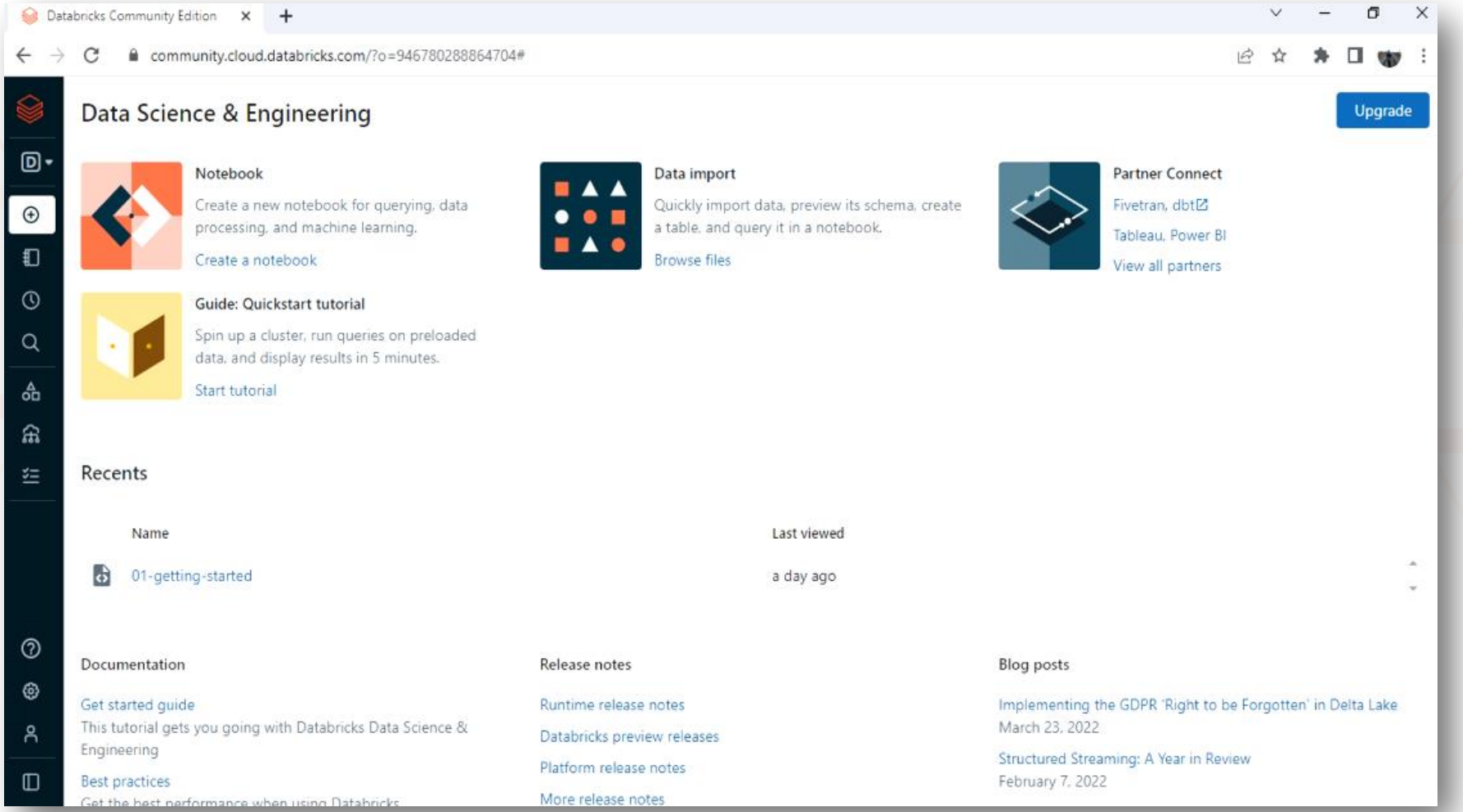
Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

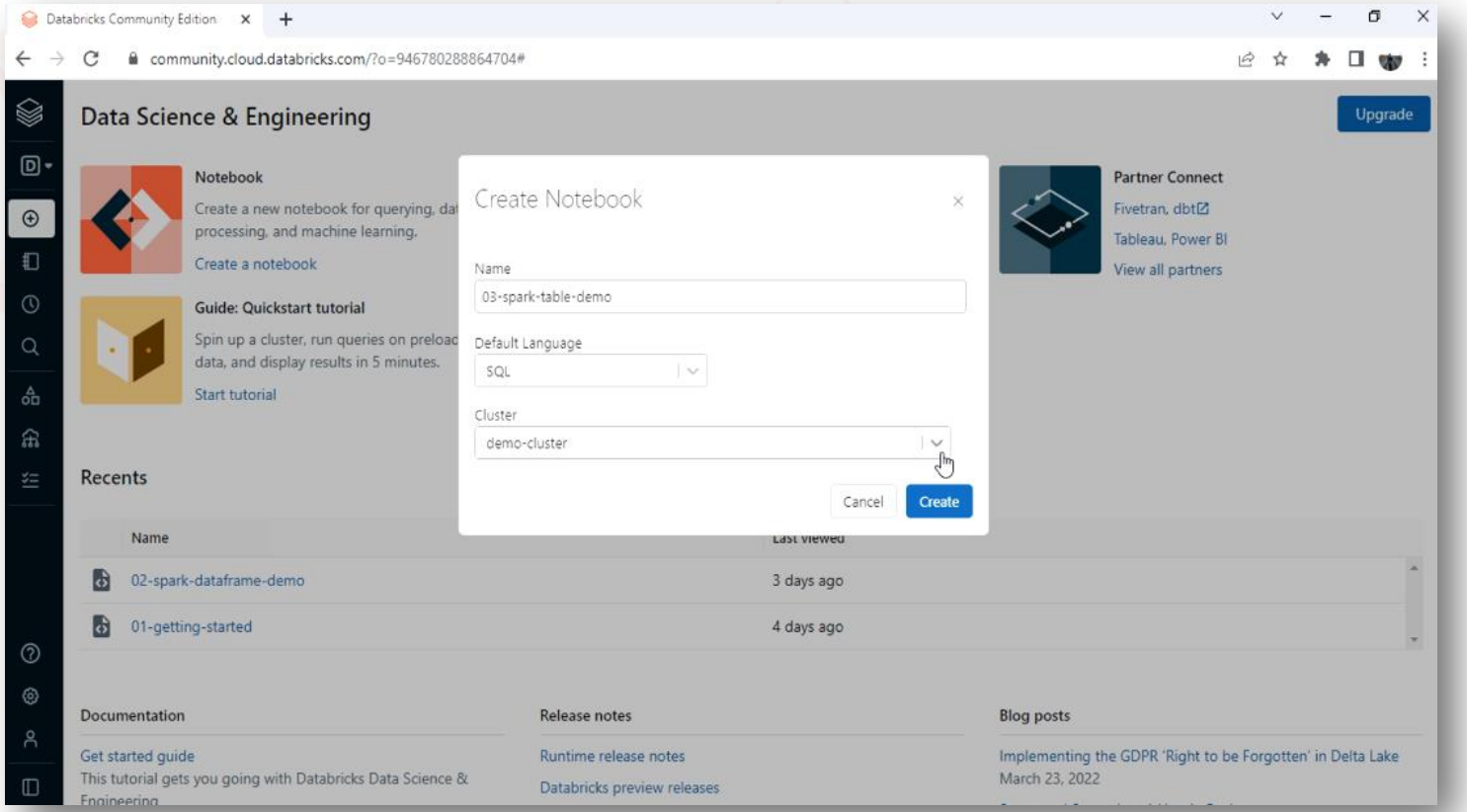
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Creating Spark Tables

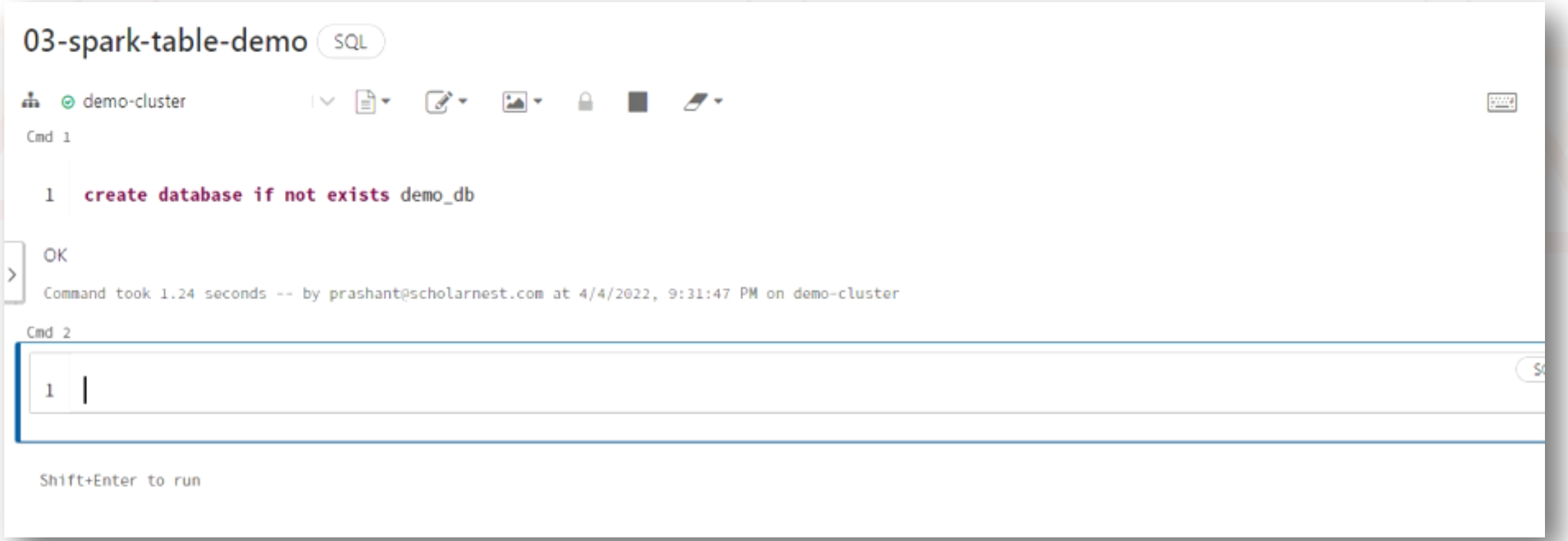
Start your Databricks Workspace.



Create a new notebook. (**Reference: 03-spark-table-demo.sql)**
Give a name to your notebook.
Choose your notebook's default language. This time, we want to use Spark SQL, so the default language for the notebook should be SQL.
Attach a cluster to your notebook. If you do not have a running cluster, go back to the Compute menu and create a new cluster.

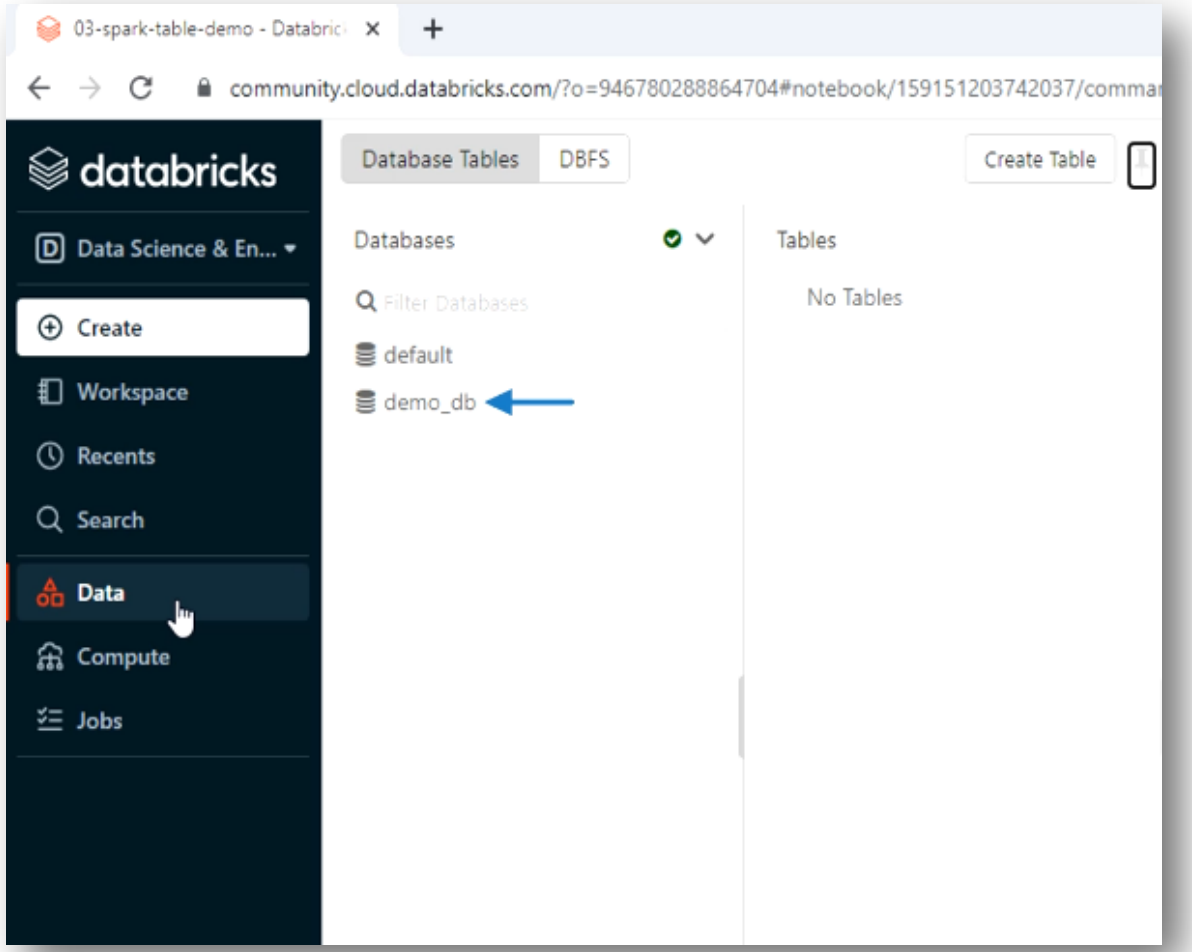


Create a database as shown below.
The code shown here is purely a SQL DDL Statement. A database should have been created.



To see your database, click the Data menu item, and you will see two databases:

1. Spark created one default database for you. So if you create a table, it will be created inside the default database. But we do not recommend using the default database.
2. Instead, you should create a database for your project and keep all your project tables inside your database. So we have already created one database for ourselves.



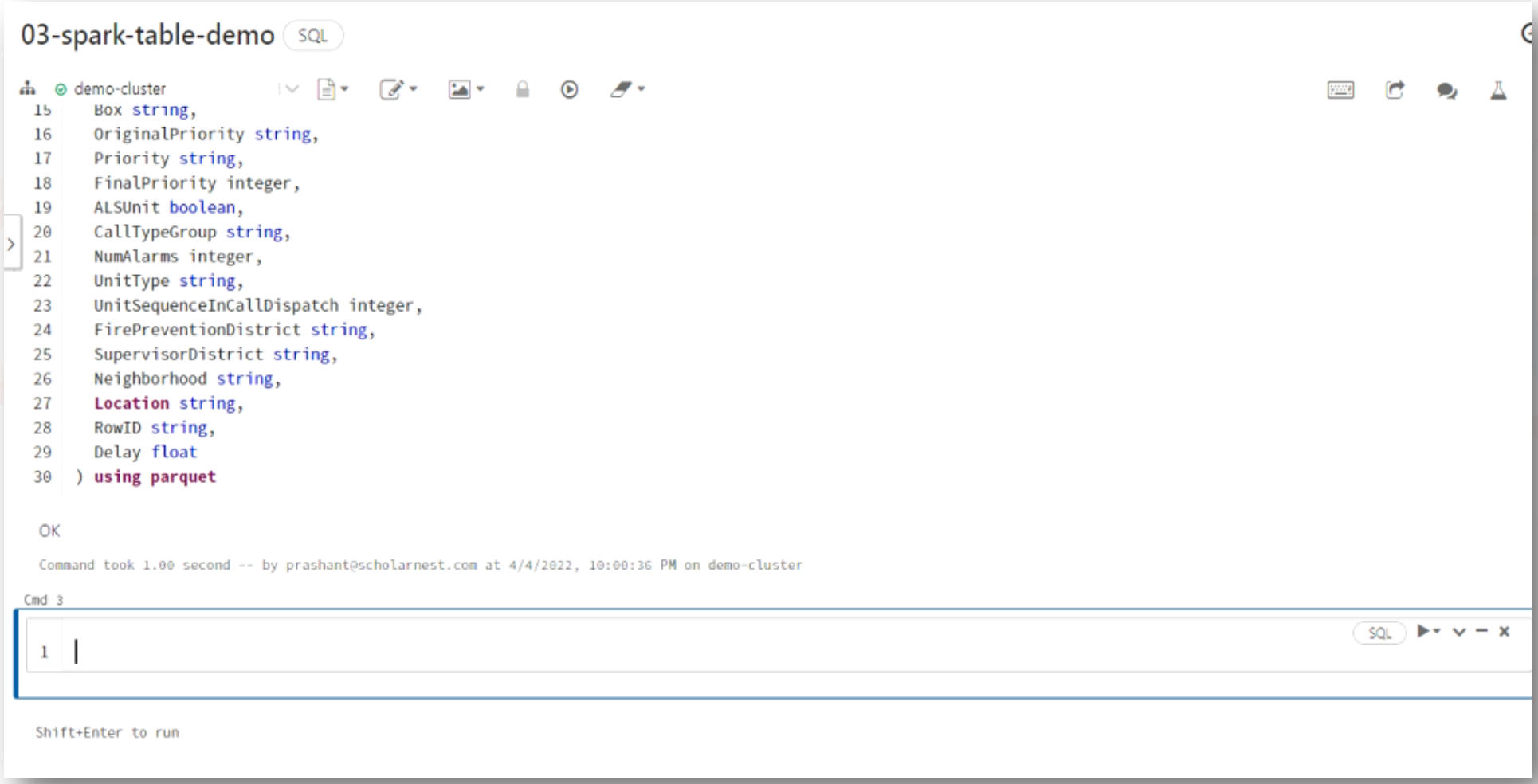
Now the next aim is to create a table. The creation of table also follows a create table DDL statement as shown below. We have 28 columns in the fire response call dataset. Most of them are string columns, but some are integer and float columns.

03-spark-table-demo SQL

demo-cluster

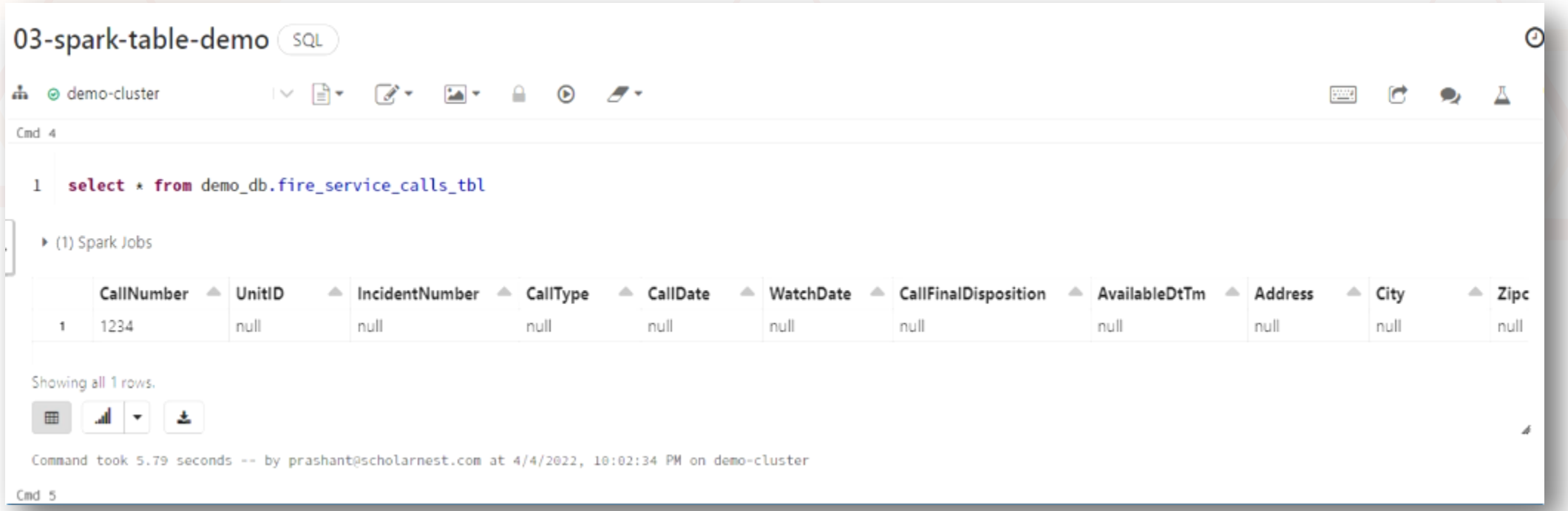
```
1  create table if not exists demo_db.fire_service_calls_tbl(
2    CallNumber integer,
3    UnitID string,
4    IncidentNumber integer,
5    CallType string,
6    CallDate string,
7    WatchDate string,
8    CallFinalDisposition string,
9    AvailableDtTm string,
10   Address string,
11   City string,
12   Zipcode integer,
13   Battalion string,
14   StationArea string,
15   Box string,
16   OriginalPriority string,
17   Priority string,
18   FinalPriority integer,
19   ALSUnit boolean,
20   CallTypeGroup string,
21   NumAlarms integer,
22   UnitType string,
23   UnitSequenceInCallDispatch integer,
24   FirePreventionDistrict string,
25   SupervisorDistrict string,
26   Neighborhood string,
27   Location string,
28   RowID string,
29   Delay float
```

Finally, we will specify the file format for the table. I will be using parquet for the table. Every database table internally stores data in files. So for this table, I want Spark to use parquet file format. And if we run this following code, a table would be created for us.



We do not have any data in the table. So we can use the insert-into command to insert some records into the table in a traditional database table, as shown in the image below. I am inserting one row into the *demo_db.fire_service_calls_tbl*. All the column values are null except the first column.

You can run a select * expression to see the content of the table. And we can see in the output that data is inserted successfully.



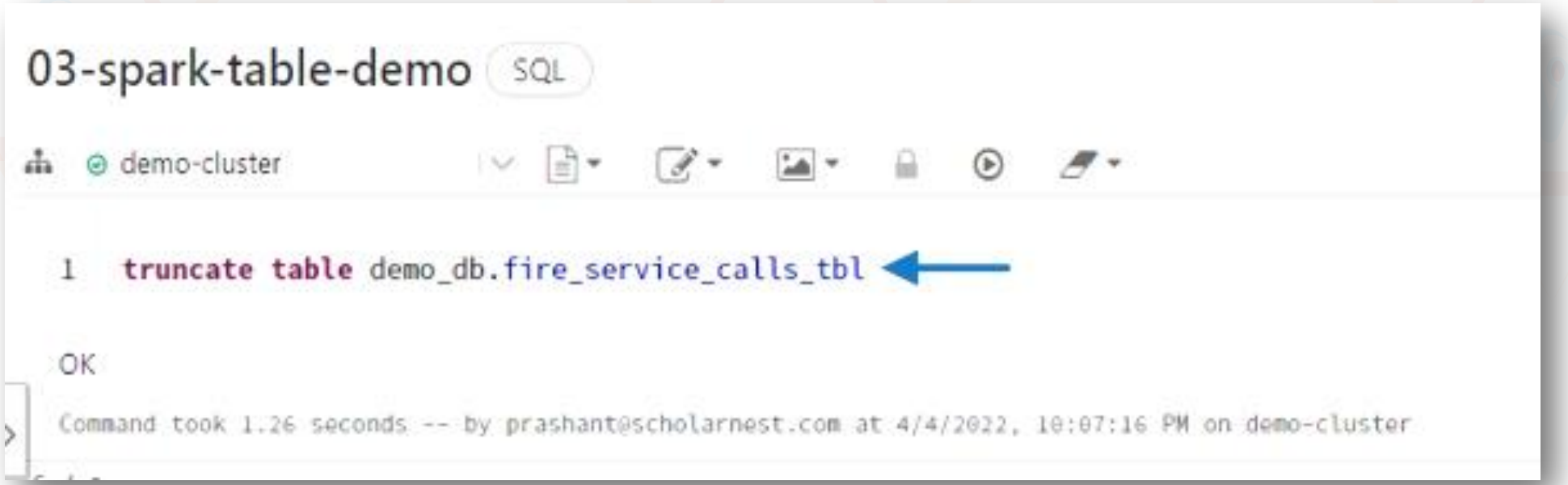
Spark tables are expected to hold millions and billions of records. And we can’t write the insert-into values command to load those records.

The insert into values expression doesn't make much sense in the big data world.

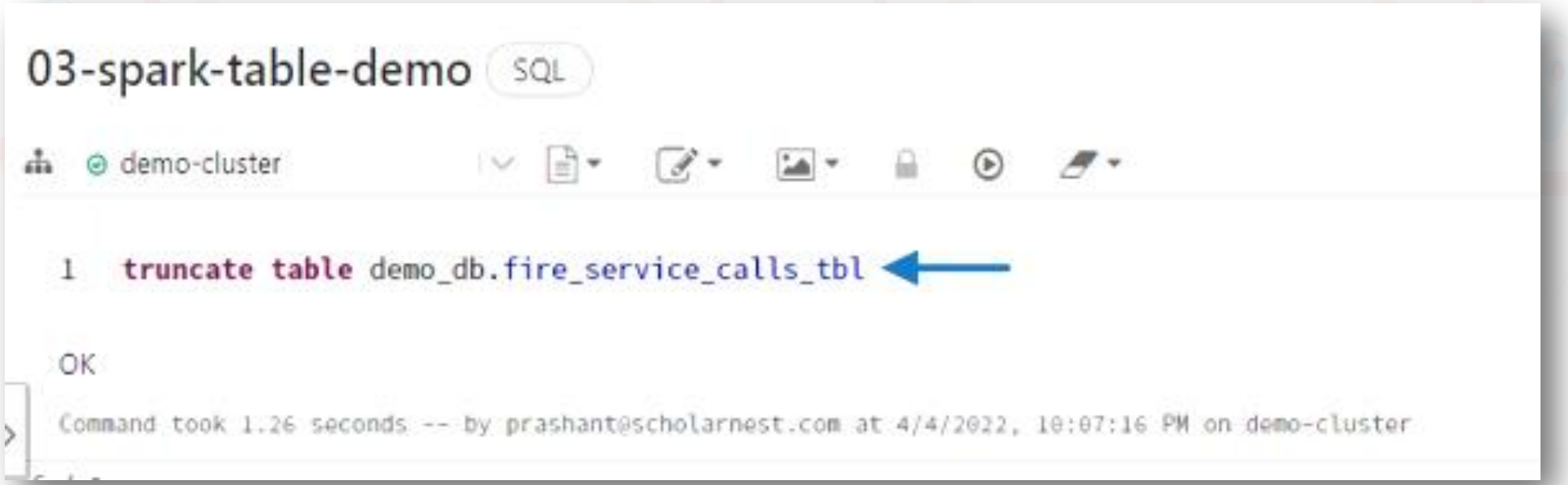
You will not find a project that uses insert into values to load data into Spark tables.

There are many ways to load data into Spark tables. But for now, we will insert records into the table from another table.

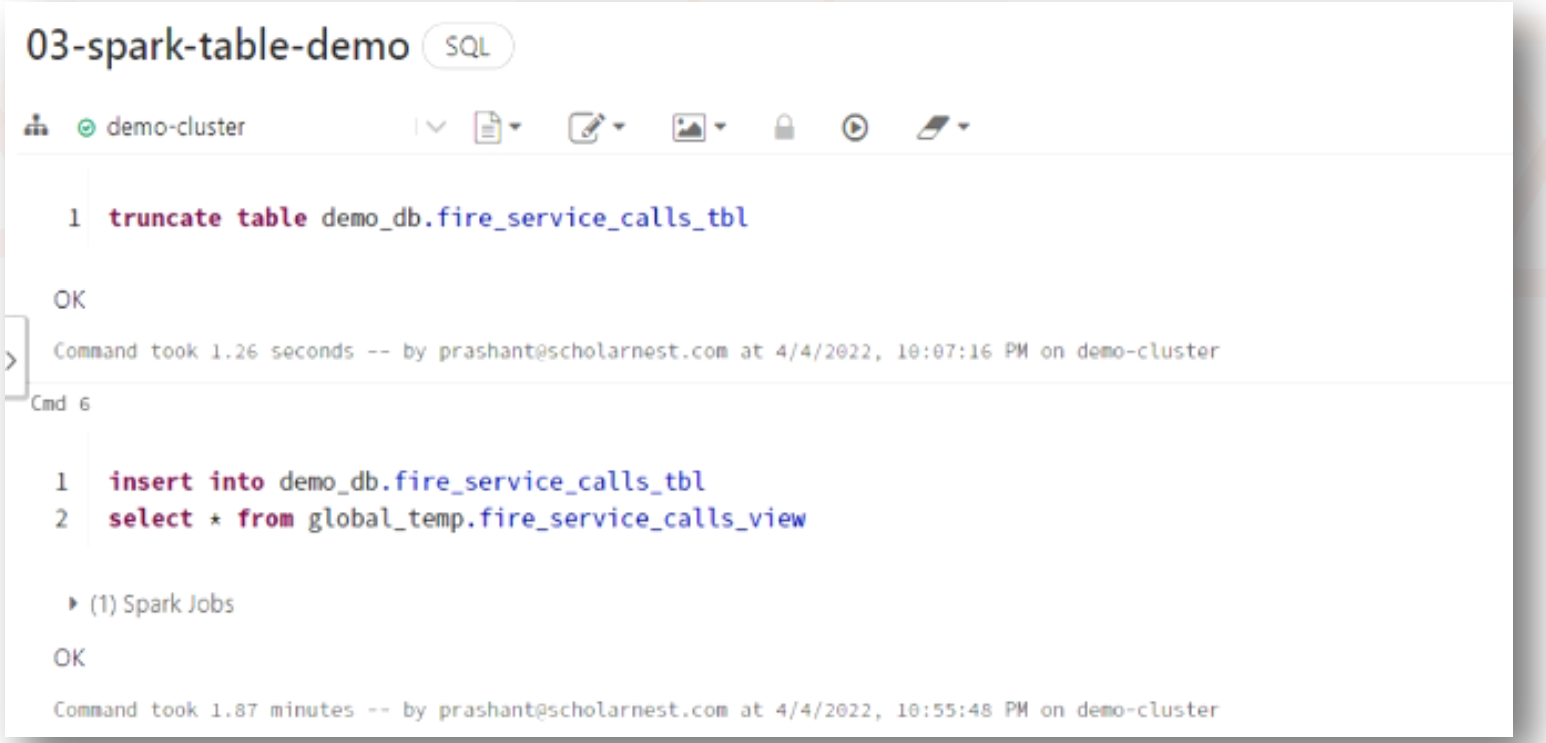
Let us truncate the table, so that we clean the garbage record that we just inserted.
We used the truncate statement because Spark SQL doesn't offer to delete statements.



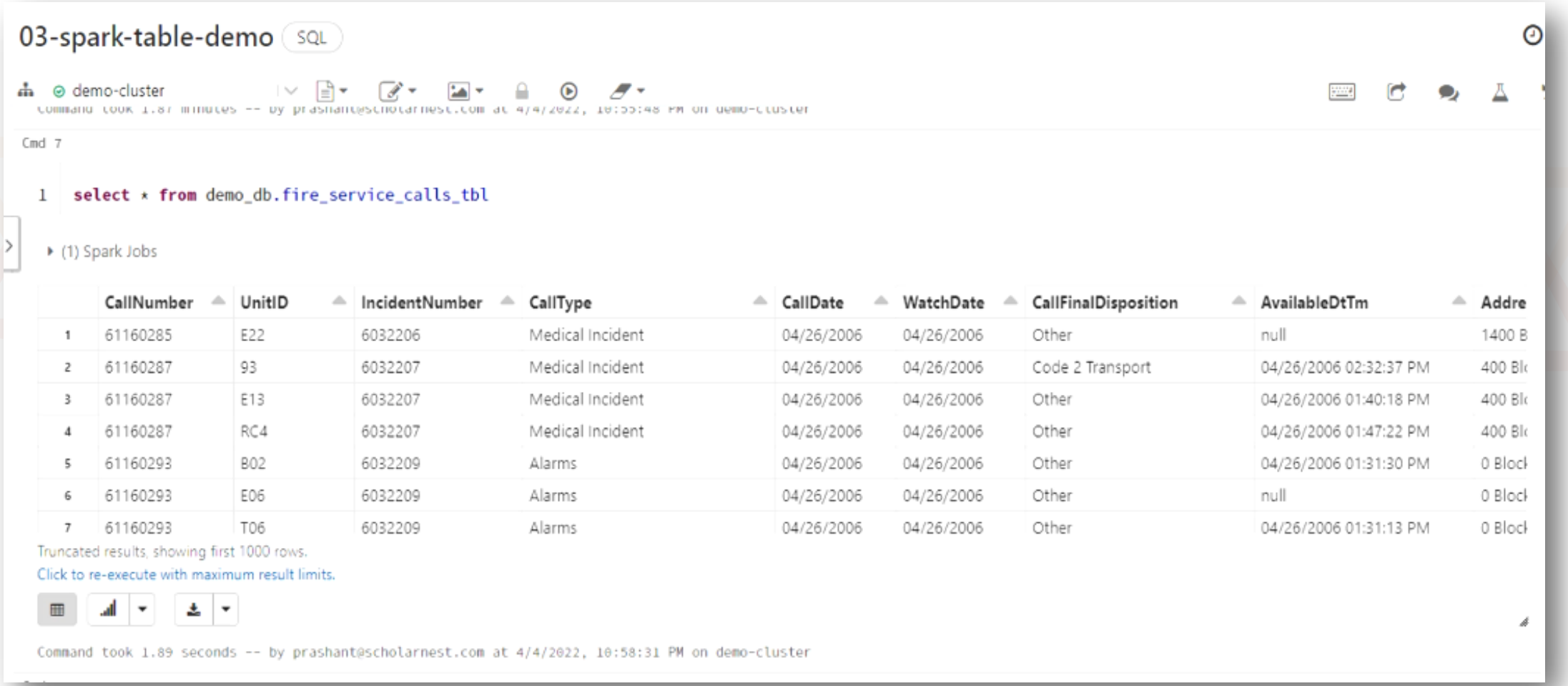
Let us truncate the table, so that we clean the garbage record that we just inserted.
We used the truncate statement because Spark SQL doesn't offer to delete statements. You cannot delete data from a Spark Table using Spark SQL. Databricks does offer a delete expression on Spark tables.



The next target is to insert data into the table after it is truncated. And we have the code for the same in the screenshot shown below. I am inserting records into the *demo_db.fire_service_calls_tbl* from *global_temp.fire_service_calls_view*.



Finally, you can see the content of the table using the select expression.



	CallNumber	UnitID	IncidentNumber	CallType	CallDate	WatchDate	CallFinalDisposition	AvailableDtTm	Addre
1	61160285	E22	6032206	Medical Incident	04/26/2006	04/26/2006	Other	null	1400 B
2	61160287	93	6032207	Medical Incident	04/26/2006	04/26/2006	Code 2 Transport	04/26/2006 02:32:37 PM	400 Bl
3	61160287	E13	6032207	Medical Incident	04/26/2006	04/26/2006	Other	04/26/2006 01:40:18 PM	400 Bl
4	61160287	RC4	6032207	Medical Incident	04/26/2006	04/26/2006	Other	04/26/2006 01:47:22 PM	400 Bl
5	61160293	B02	6032209	Alarms	04/26/2006	04/26/2006	Other	04/26/2006 01:31:30 PM	0 Block
6	61160293	E06	6032209	Alarms	04/26/2006	04/26/2006	Other	null	0 Block
7	61160293	T06	6032209	Alarms	04/26/2006	04/26/2006	Other	04/26/2006 01:31:13 PM	0 Block

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

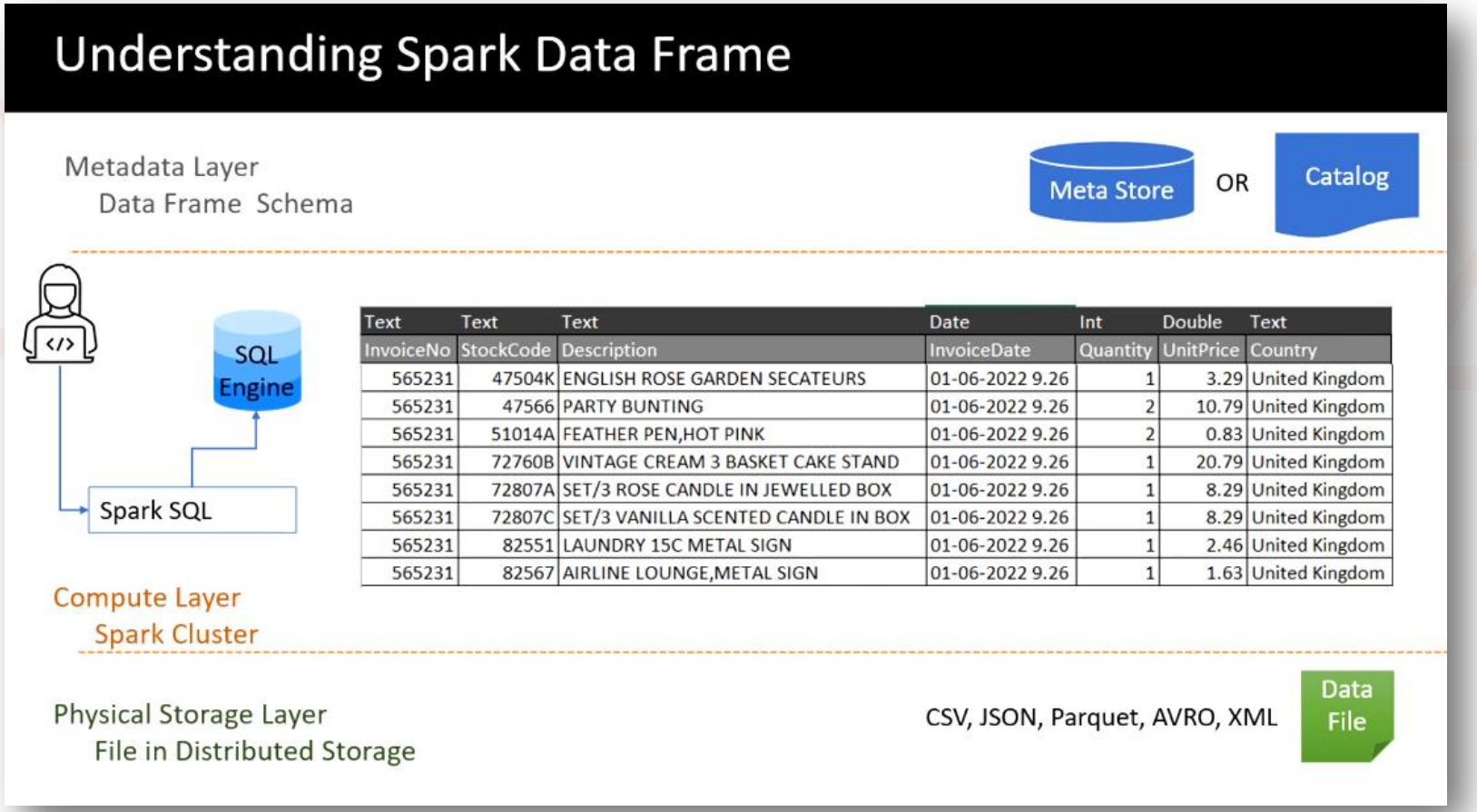
Instructor: Prashant Kumar Pandey

Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Problems with Databricks Community

Before we proceed further with the micro project, let us have a look at the below diagram once again. Every Spark Table or the Spark Dataframe lives in three layers:

1. Layer – It stores Table or Dataframe definition and the schema information.
2. Compute Layer – It runs your Spark SQL engine.
3. Physical Storage Layer – It stores the spark data in a data file.



Text	Text	Text	Date	Int	Double	Text
InvoiceNo	StockCode	Description	InvoiceDate	Quantity	UnitPrice	Country
565231	47504K	ENGLISH ROSE GARDEN SECATEURS	01-06-2022 9.26	1	3.29	United Kingdom
565231	47566	PARTY BUNTING	01-06-2022 9.26	2	10.79	United Kingdom
565231	51014A	FEATHER PEN,HOT PINK	01-06-2022 9.26	2	0.83	United Kingdom
565231	72760B	VINTAGE CREAM 3 BASKET CAKE STAND	01-06-2022 9.26	1	20.79	United Kingdom
565231	72807A	SET/3 ROSE CANDLE IN JEWELLED BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	72807C	SET/3 VANILLA SCENTED CANDLE IN BOX	01-06-2022 9.26	1	8.29	United Kingdom
565231	82551	LAUNDRY 15C METAL SIGN	01-06-2022 9.26	1	2.46	United Kingdom
565231	82567	AIRLINE LOUNGE,METAL SIGN	01-06-2022 9.26	1	1.63	United Kingdom

When you execute a Spark SQL query, or you run a Dataframe code, the Spark SQL engine will refer to the metadata store. And the metadata will tell the following things:

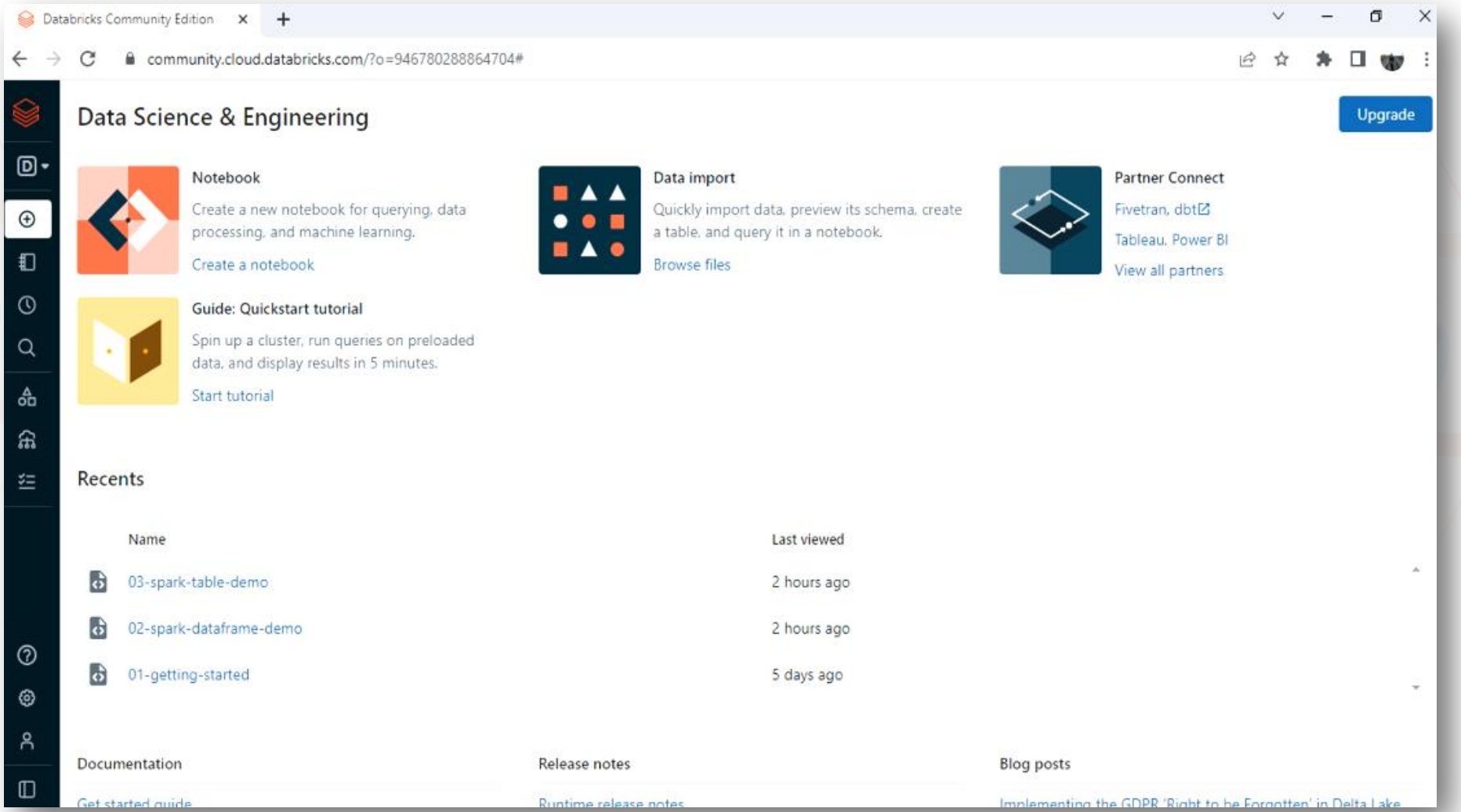
1. Where is the data for this table?
2. What is the schema of the data and table?

If the Spark SQL engine doesn't find metadata data for the table, it throws an analysis exception.

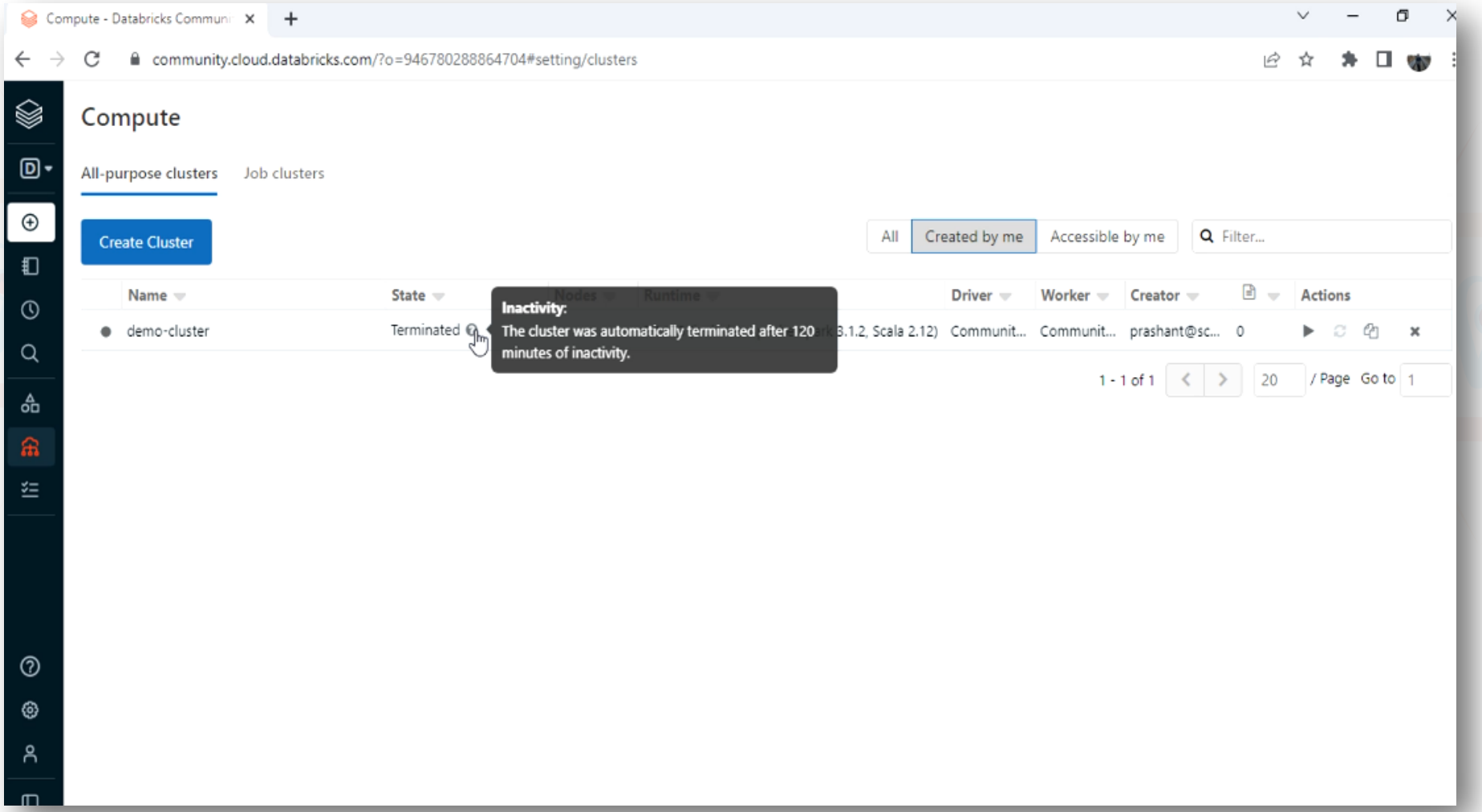
If metadata is there, the SQL engine will read the data from the data file and present it to you like a table or Dataframe. So, the three(Metadata, Compute and Storage) layers of Spark are supercritical.

You will face problems and see errors when any of these three layers are missing or broken.

Go to your Databricks workspace and go to the compute menu.

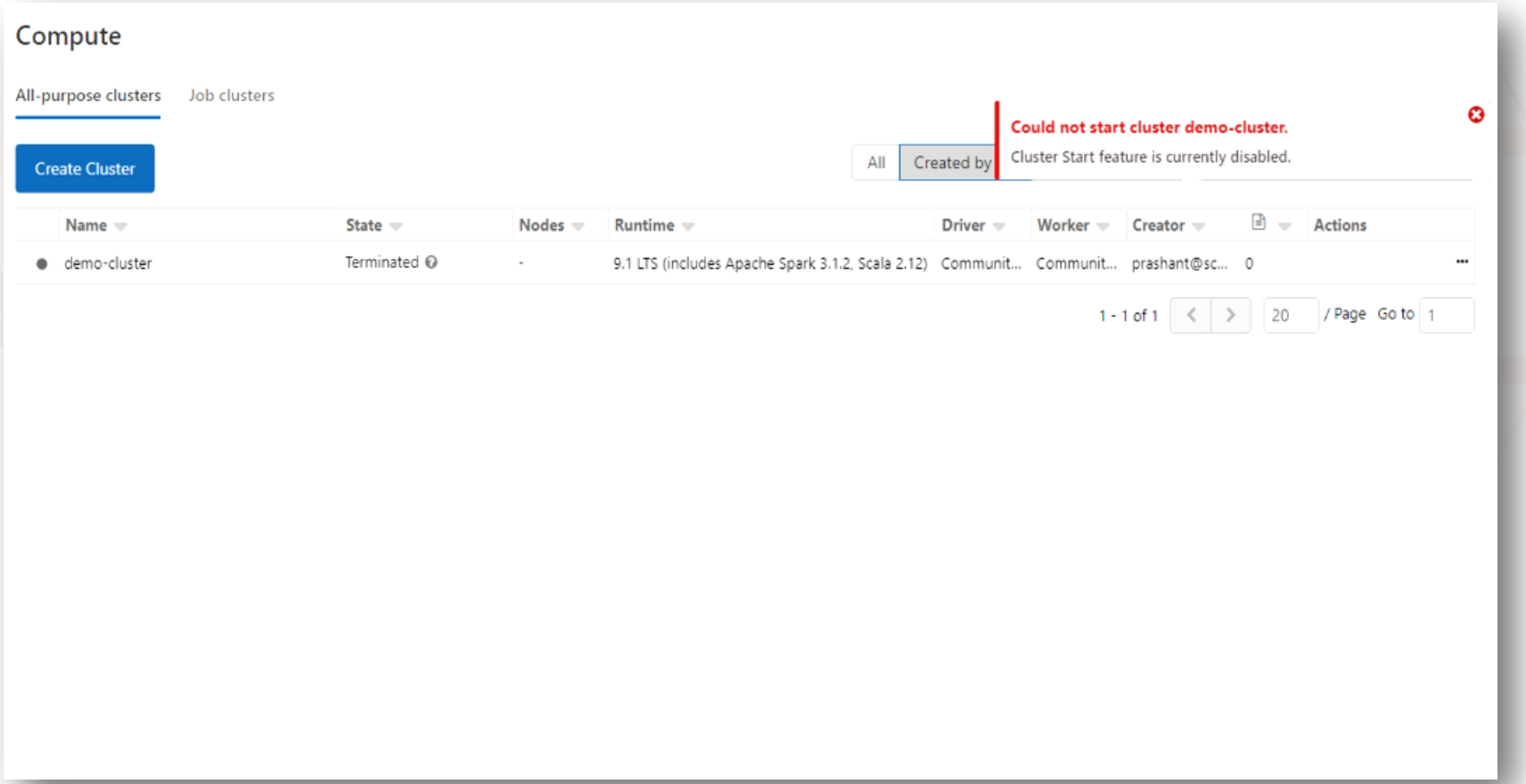


You need a running cluster for doing anything in the Databricks environment because you need a compute layer. But we have an old cluster here which is in terminated state because I left it idle for more than 2 hours, so Databricks terminated my cluster.

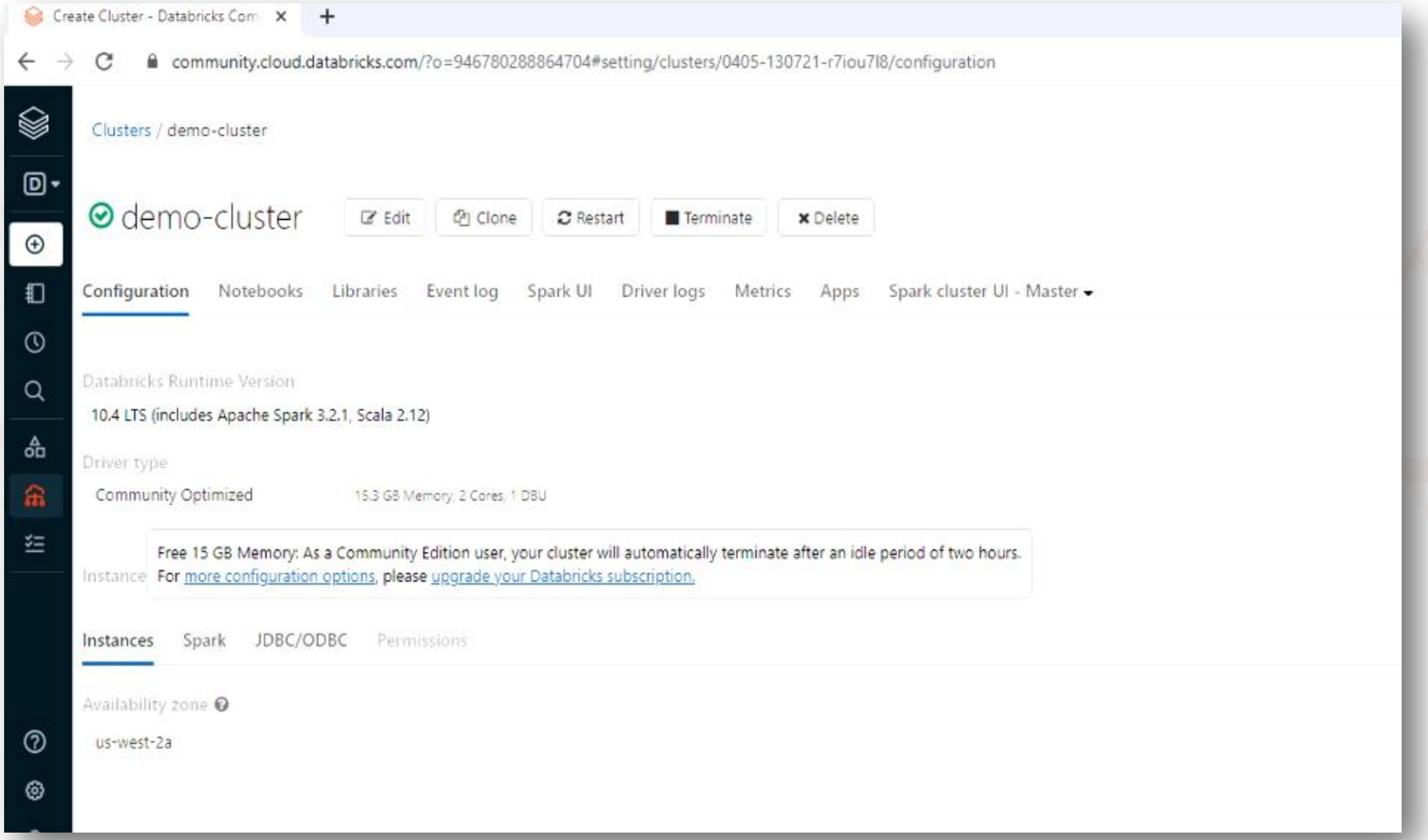


I cannot restart this cluster again. I must create a new cluster if I want to work again. Databricks community edition will terminate your cluster after 2 hours of idle time, and once your cluster is terminated, Databricks will clean up the following things:

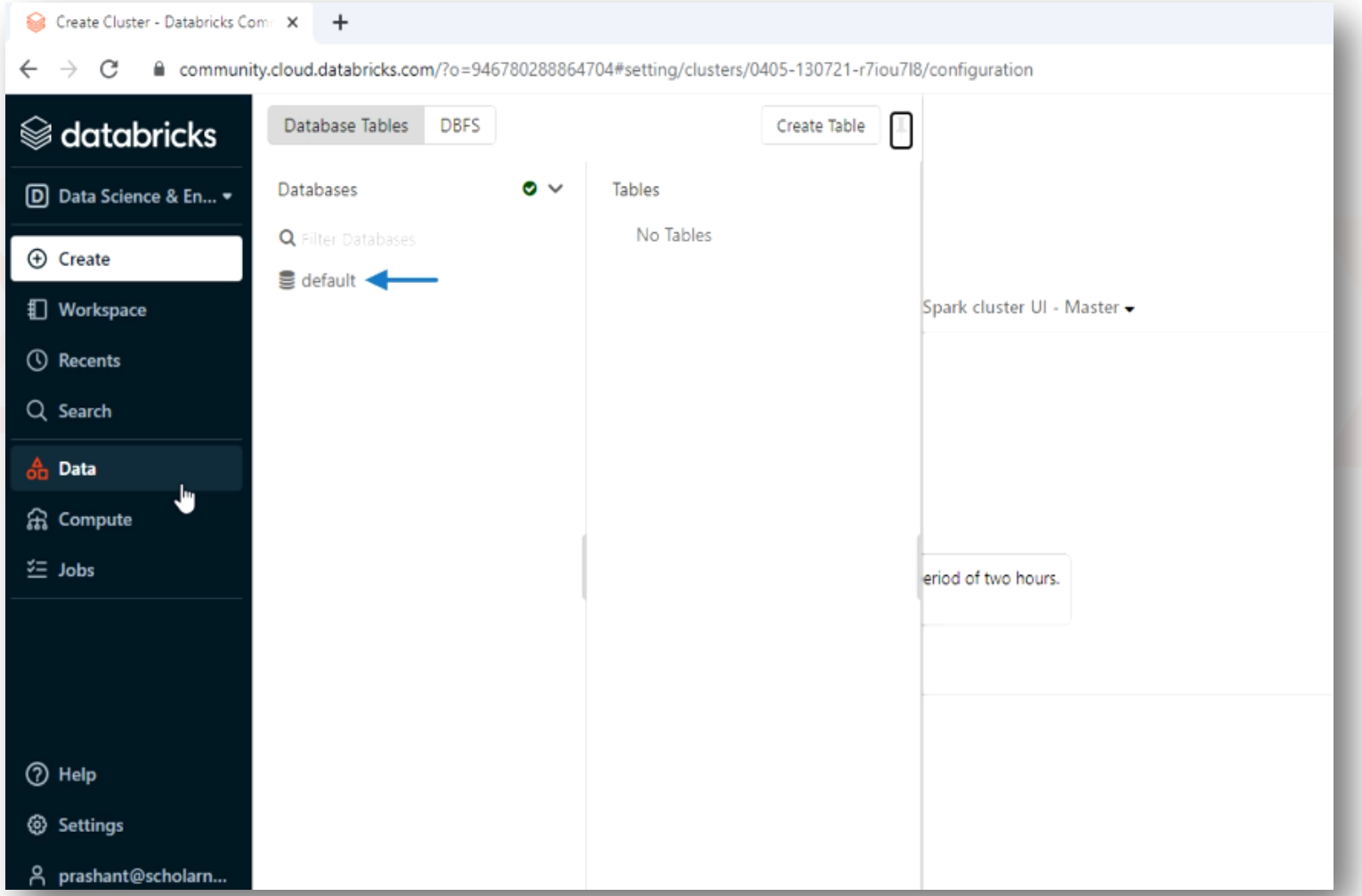
1. Cluster VM (Compute Layer) – You cannot restart your cluster again.
2. Spark Metadata Store (Metadata Layer) – You will lose your databases and tables.



You can delete the old cluster, and create a new one with same name. Creating a cluster takes about 5-10 minutes.

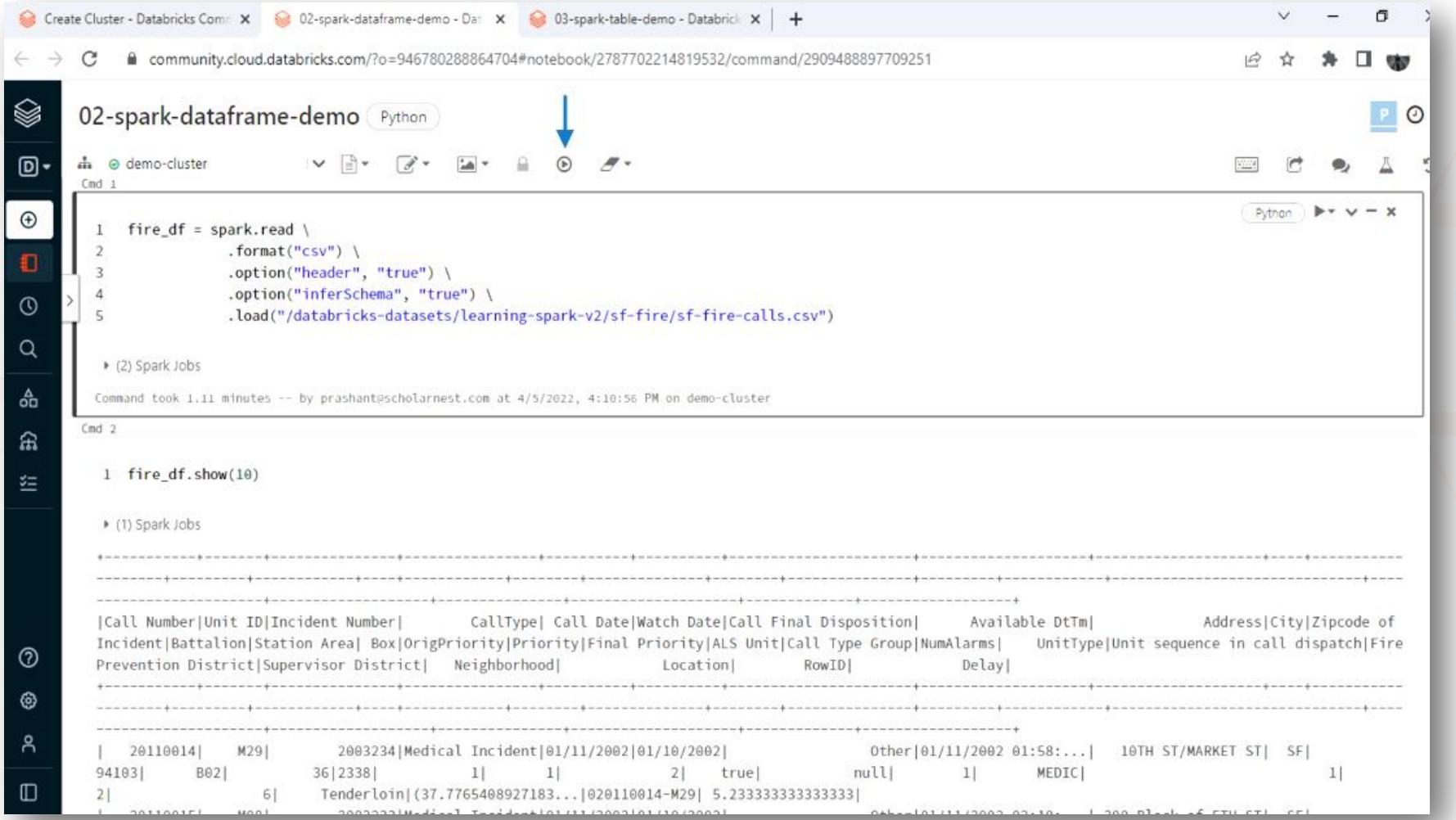


You can click the data menu, and you can see the database and tables. We only have a default database the *demo_db* database that we created yesterday is gone. Because Databricks cleaned my Spark metadata store as my cluster was idle for more than 2 hours. So I lost my database and table definition. This problem does not exist for the full licensed version of the Databricks Cloud.

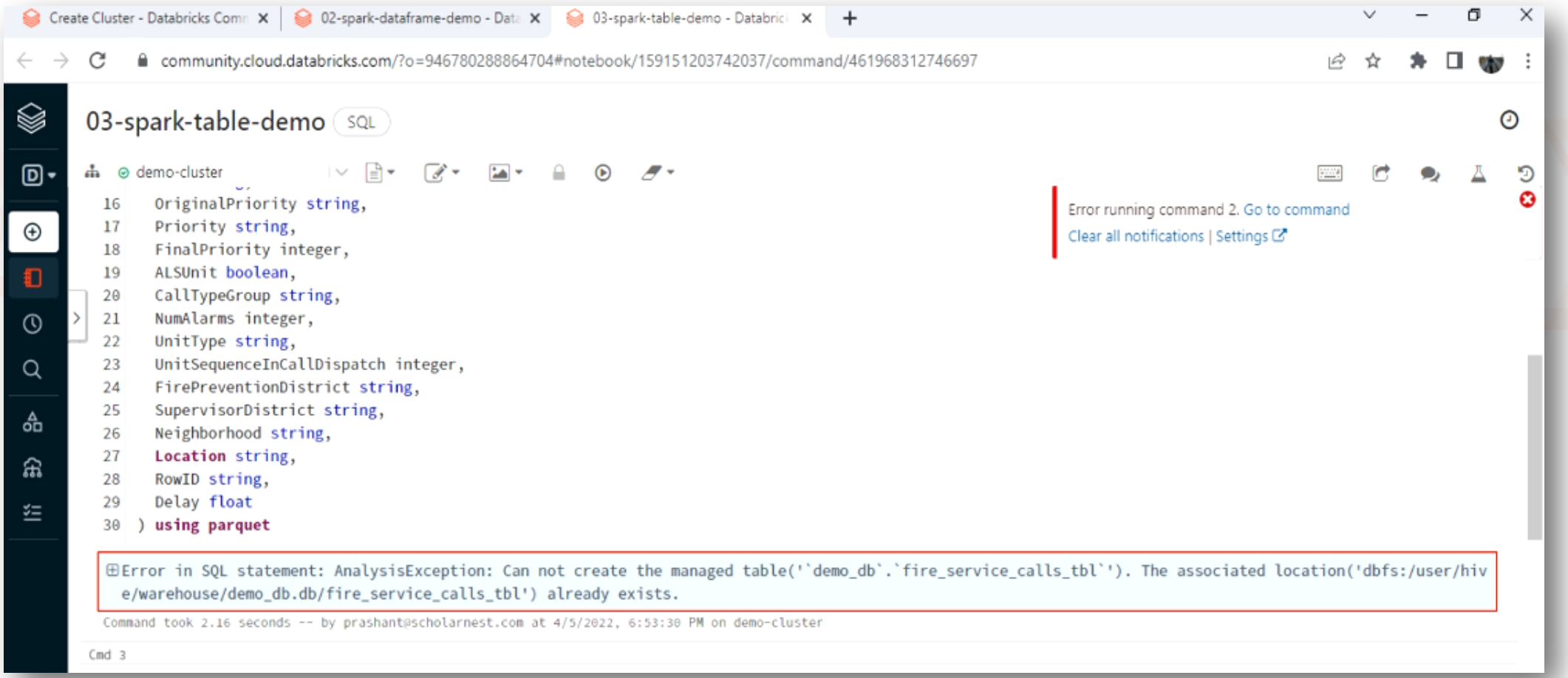


The community edition will clean up the VM and the metadata store.
But it doesn't clean up the directories and files.
So you do not lose your storage layer. Your data files remain in place even if your compute and metadata layer is gone.
Directories are created in an external distributed storage, and files are stored in these directories on external storage.

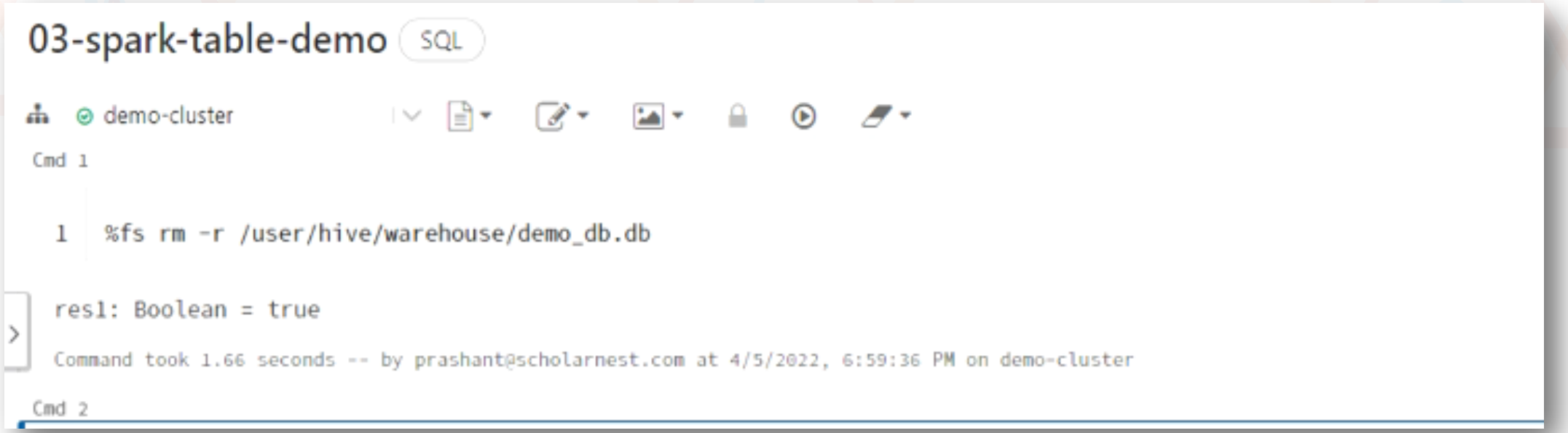
You can go the notebooks you created earlier (02-spark-dataframe-demo.ipynb and 03-spark-table-demo.sql) from the workspace menu option, and click the run all option shown below to run the entire notebook at once. Make sure your cluster is attached to both the notebooks.



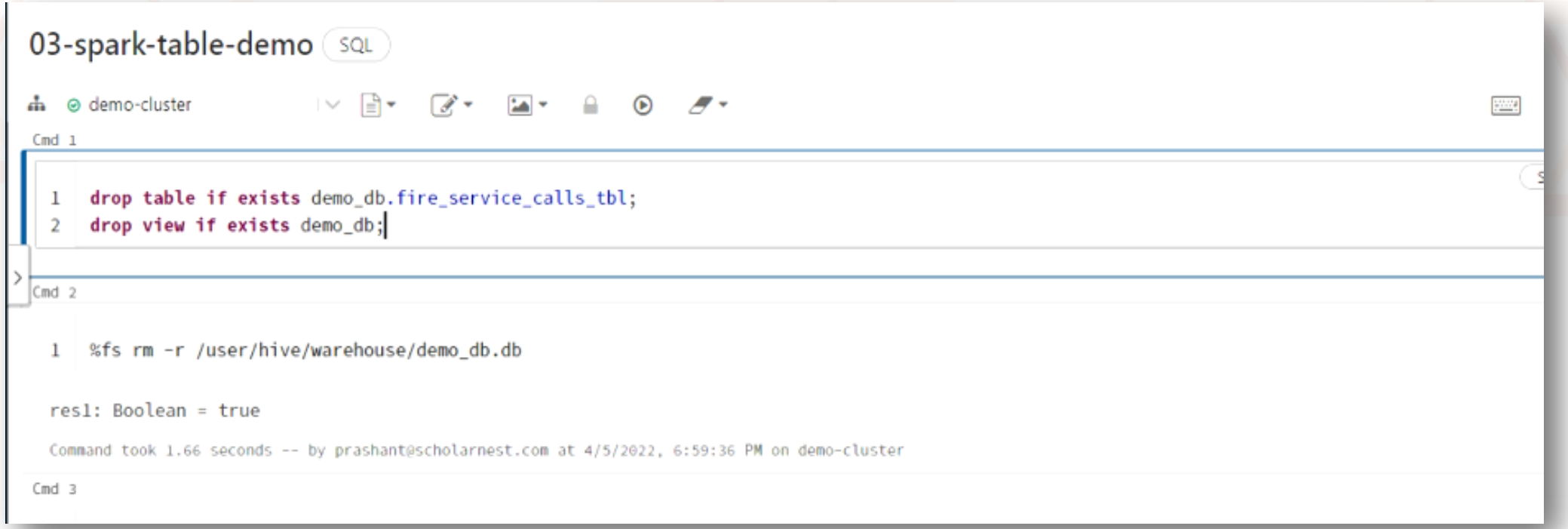
We ran the *02-spark-dataframe-demo.ipynb* successfully but encountered an error in the *03-spark-table-demo.sql* notebook. The error says that “can not create the managed table. The associated location already exists.” We didn't face any errors yesterday when we created this table. But we are getting an error today because Databricks didn't clean the directory and data files. My storage layer is still there. So my data directory is not cleaned.



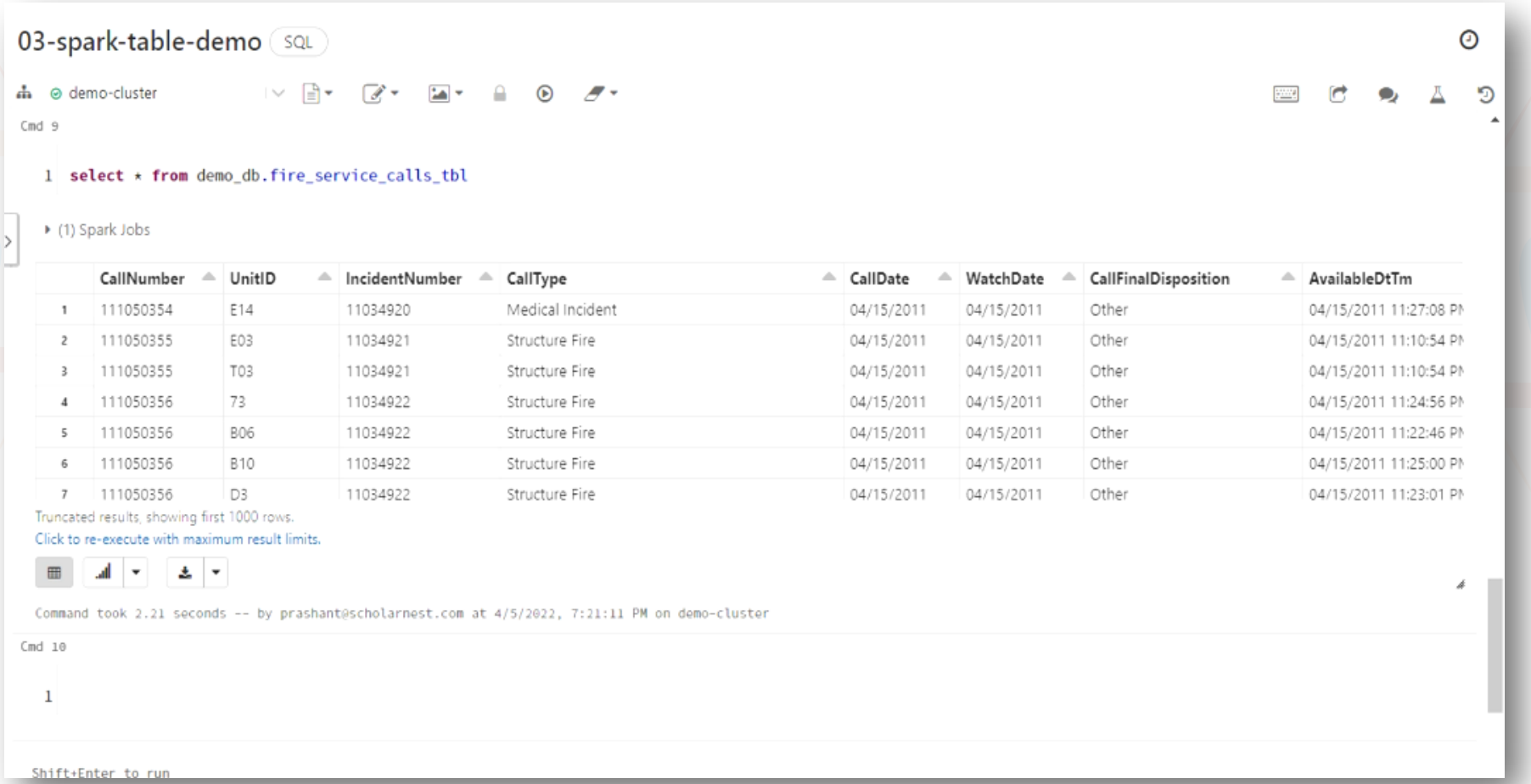
In order to fix the Analysis Exception, we can add a new cell at the top of the notebook and type the filesystem rm command to clean the directory. Copy the directory location from the error message.



I also recommend adding a drop table and drop database command at the top. You can add another cell at the top of the notebook and run two DDL statements to drop your table and database. Do not forget to add a semicolon at the end of the SQL expression.



Now, you can go ahead and click the run all button for the *03-spark-table-demo.sql* notebook and it should work perfectly fine without giving any errors.



Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

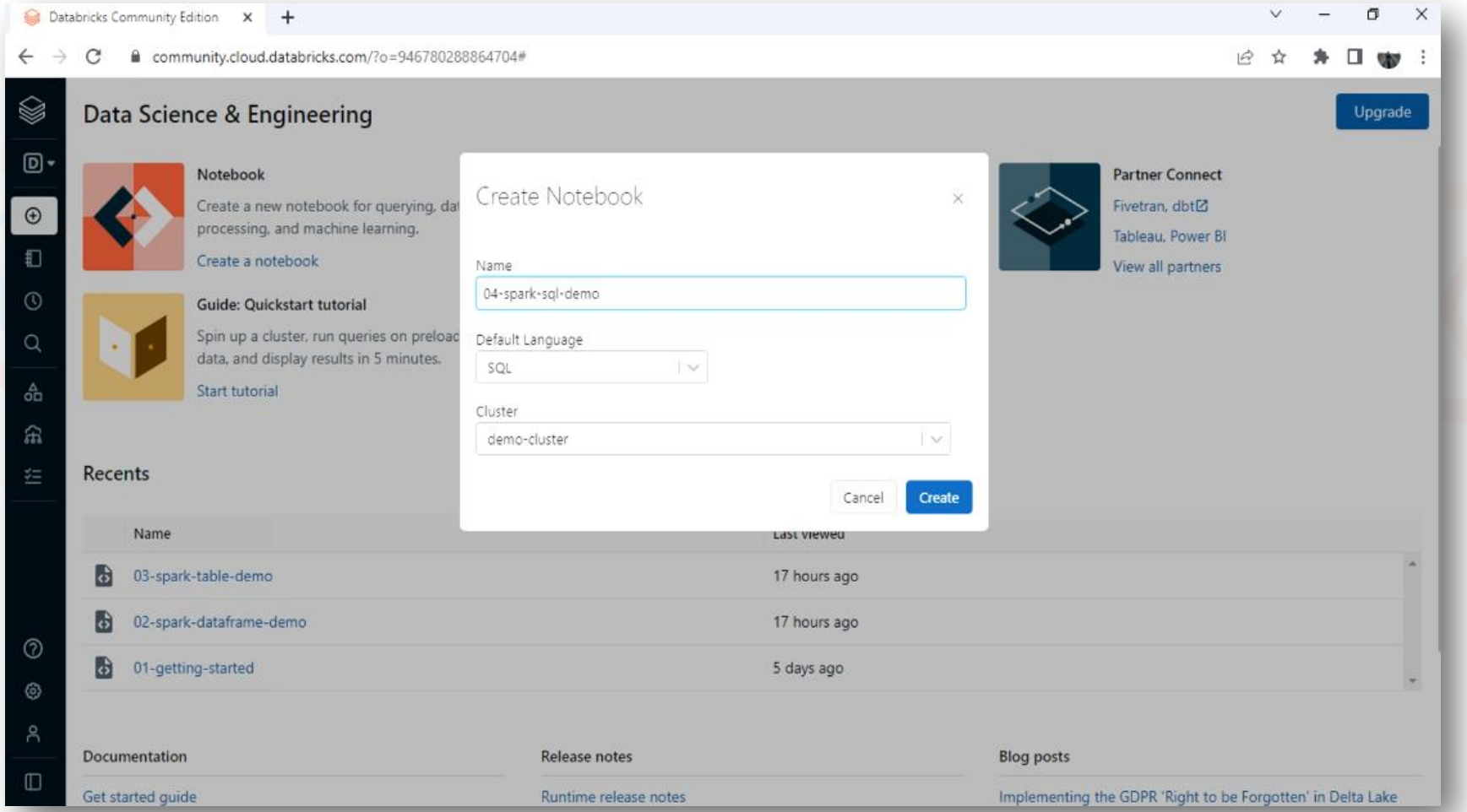
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Data Caching Essentials

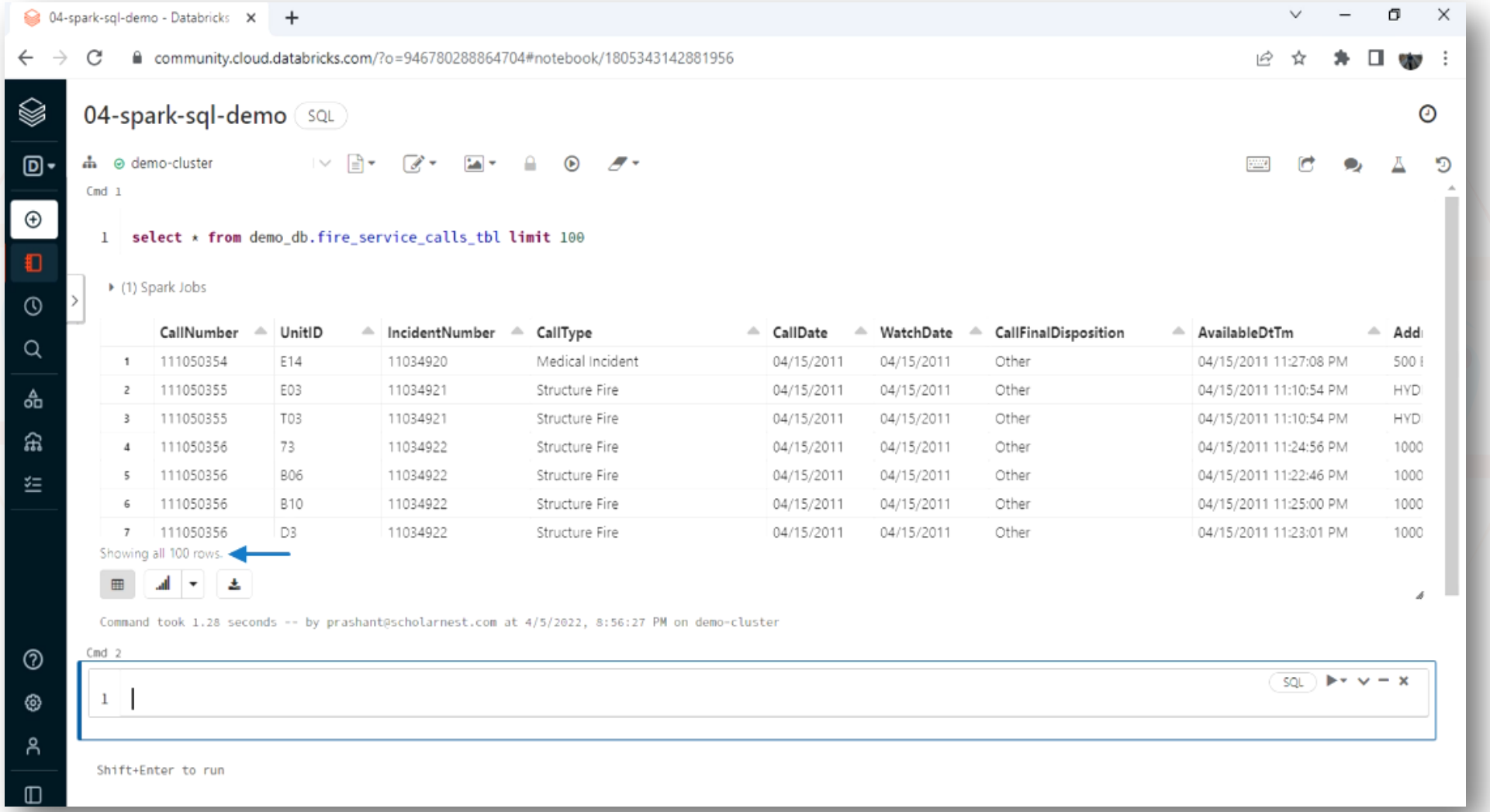
Go to your Databricks workspace.

Create a notebook. **(Reference: 04-spark-sql-demo.sql)**

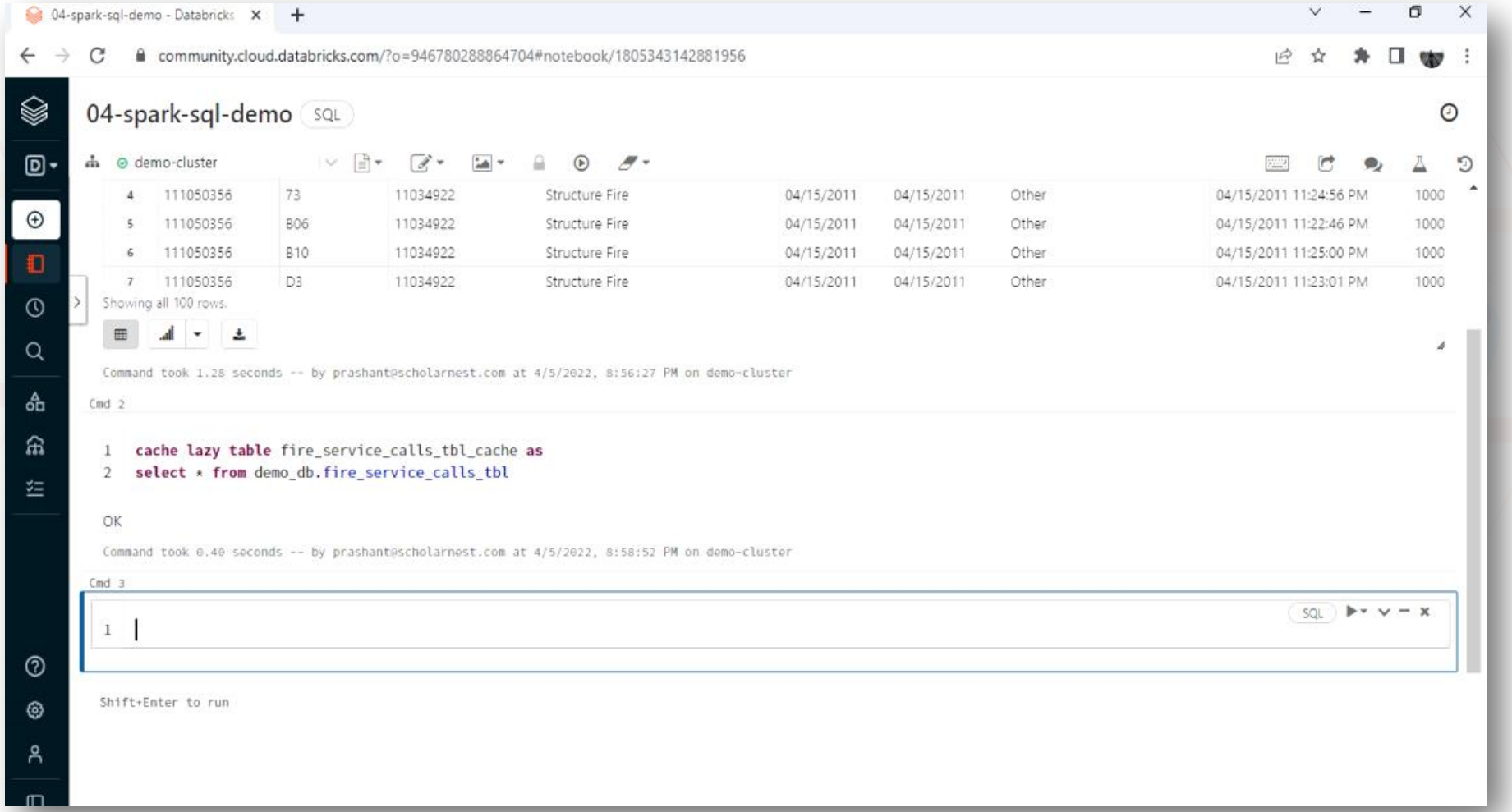
We want to use Spark SQL for data analysis. So choose SQL as the default language of your notebook. Make sure to attach a cluster to your notebook.



We created a table for the San Fransisco fire call response dataset. Now let me type the code to select hundred records from the table.



I want to cache this table into memory to speed up my queries. I want to run ten queries on the same table. So it makes sense that I cache this table into the memory and avoid reading data from the disk for every query. And we have the code for the same in the snapshot below.



You must learn the following things about the Spark SQL cache table.

1. How does the Spark cache table works?

   The cache-table-as command will create a temporary view using the Select expression. So you already have a table. But when you cache it, you will create a temporary view on the table. Once your table is cached as a temporary view, you can use the view or the table. Once cached, Spark will always bring data from the cache.

2. How to make it re-executable?

   The cache-table-as command creates a temporary view, so you cannot rerun it. And if you try re-running the cache command again, you will see a “temporary view already exists” error. You can fix this issue by dropping the temporary view before rerunning the cache-table-as command.

3. What is Lazy?

   The lazy is to make sure we do not cache the table until it is used. It is optional. You can also remove it, but it is recommended to keep it as the execution will be very fast because it simply marks the table to be cached on its first use.

Caching the table consumes a lot of memory. So you should make use of this feature carefully. We have some guidelines:

1. Do not cache tiny tables because tiny table will anyway work faster.
2. Do not cache huge tables that cannot fit in the Spark memory because there is no point in caching a table into memory if it cannot fit in the memory.

It is recommended to cache medium-sized frequently used tables.

So, you should cache a table if you meet the following criteria:

1. Table data is a reasonable size and fits into the memory.
2. Table is frequently used. You should not cache a table that you do not want to use again and again.

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

Databricks Spark Certification and beyond

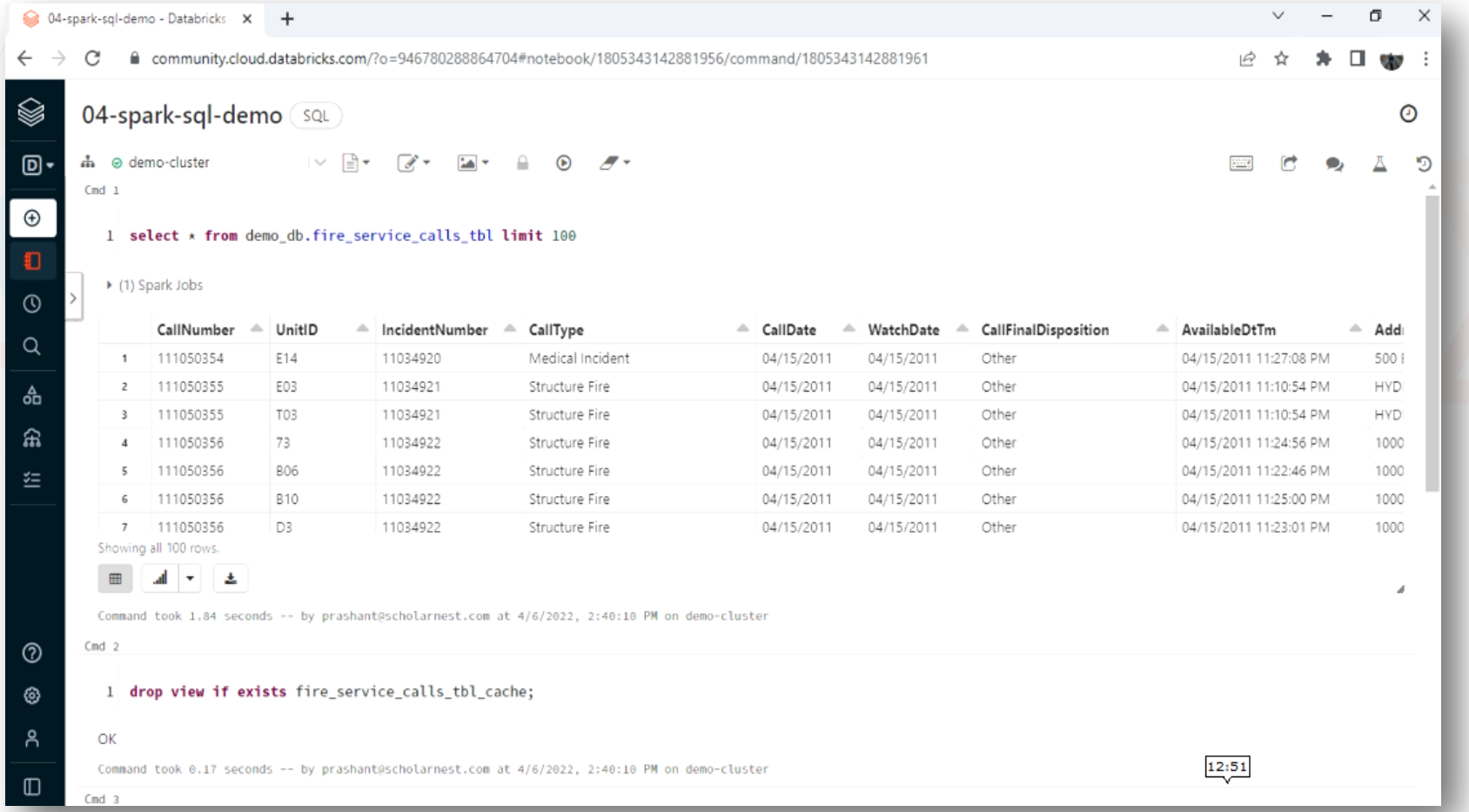
Instructor: Prashant Kumar Pandey

Absolute Beginner to Specialization in Apache Spark and Azure Databricks

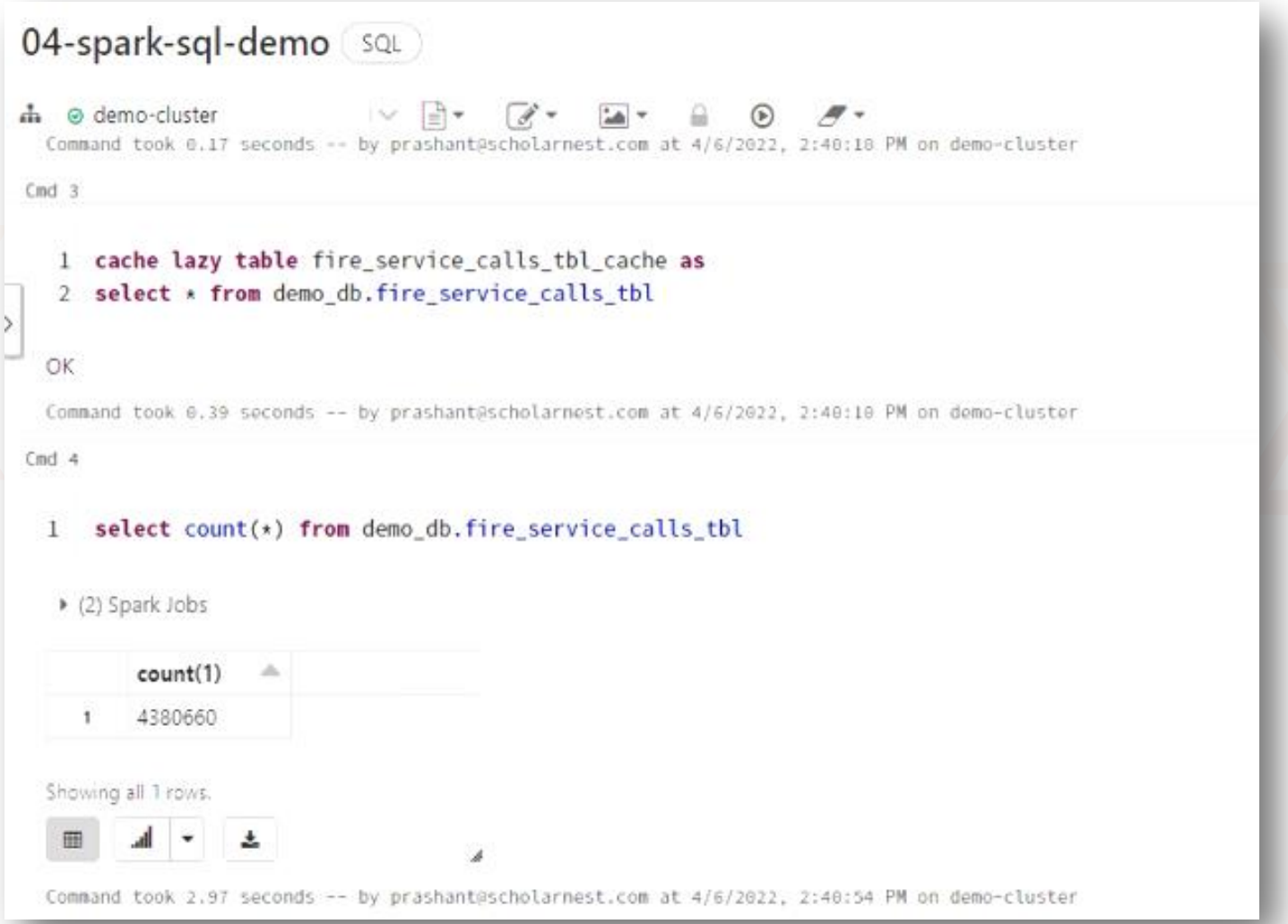
Working with Spark SQL

Go back to your notebook you created in the previous lecture. **(Reference: 04-spark-sql-demo.sql)**

We created a table and also cached it into the memory. Now we are ready to start writing some SQL expressions to answer the 10 micro project questions.



Let us start by taking a simple count from the table. We can use the select count(*) query. If we run it, we can see there are more than 4.3 million records in the table.

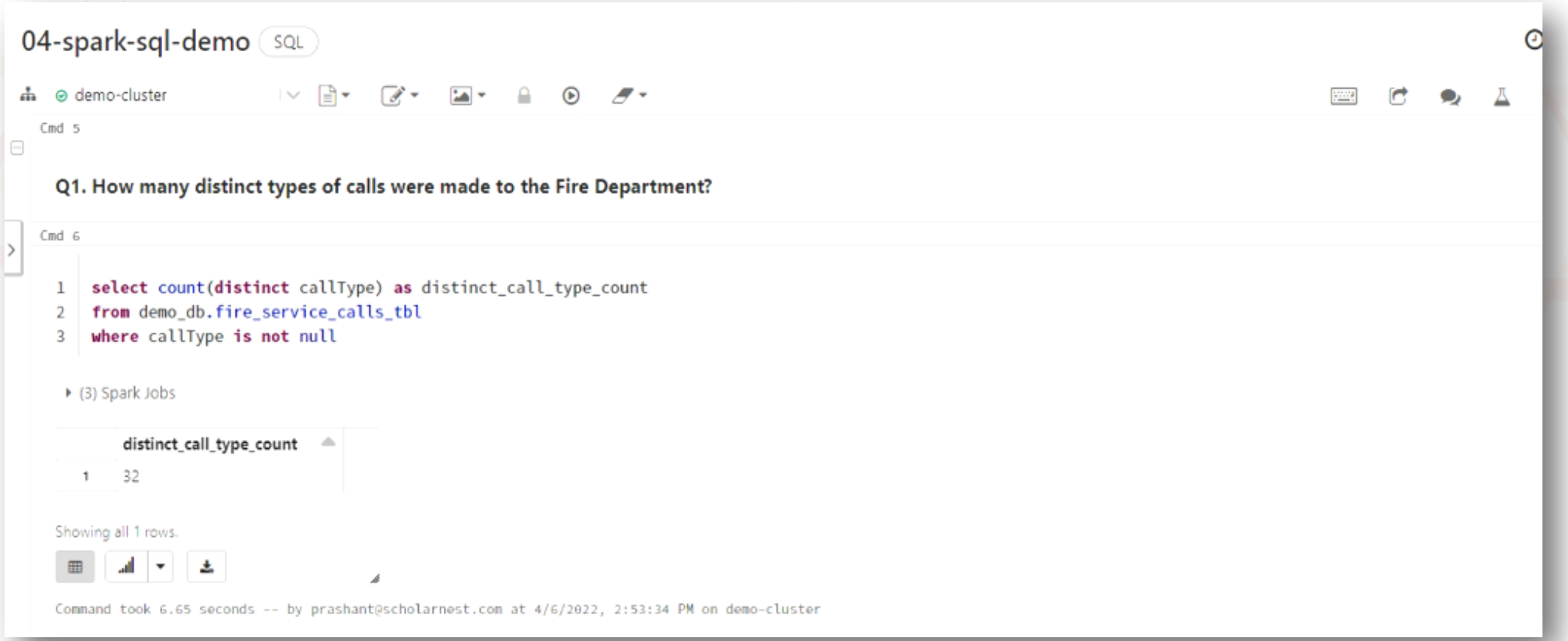


Here are ten questions we want to answer:

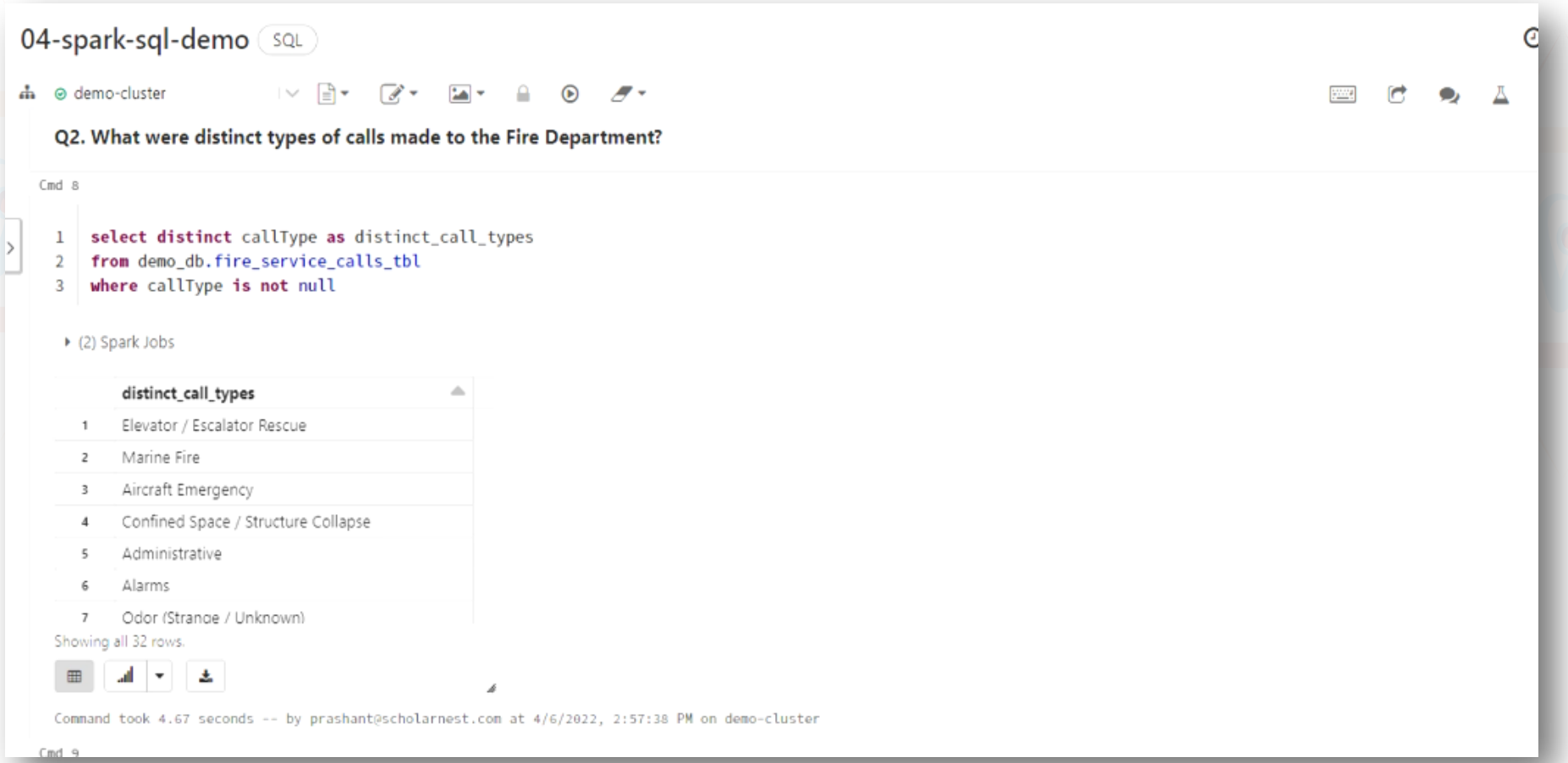
1. How many distinct types of calls were made to the fire department?
2. What are distinct types of calls made to the fire department?
3. Find out all responses or delayed times greater than 5 mins?
4. What were the most common call types?
5. What zip codes accounted for the most common calls?
6. What San Francisco neighbourhoods are in the zip codes 94102 and 94103
7. What was the sum of all calls, average, min, and max of the call response times?
8. How many distinct years of data are in the CSV file?
9. What week of the year in 2018 had the most fire calls?
10. What neighbourhoods in San Francisco had the worst response time in 2018?

Databricks Notebook allows us to document things using markdown language.
You can use the %md command to create a documentation cell, as shown in the image given below.

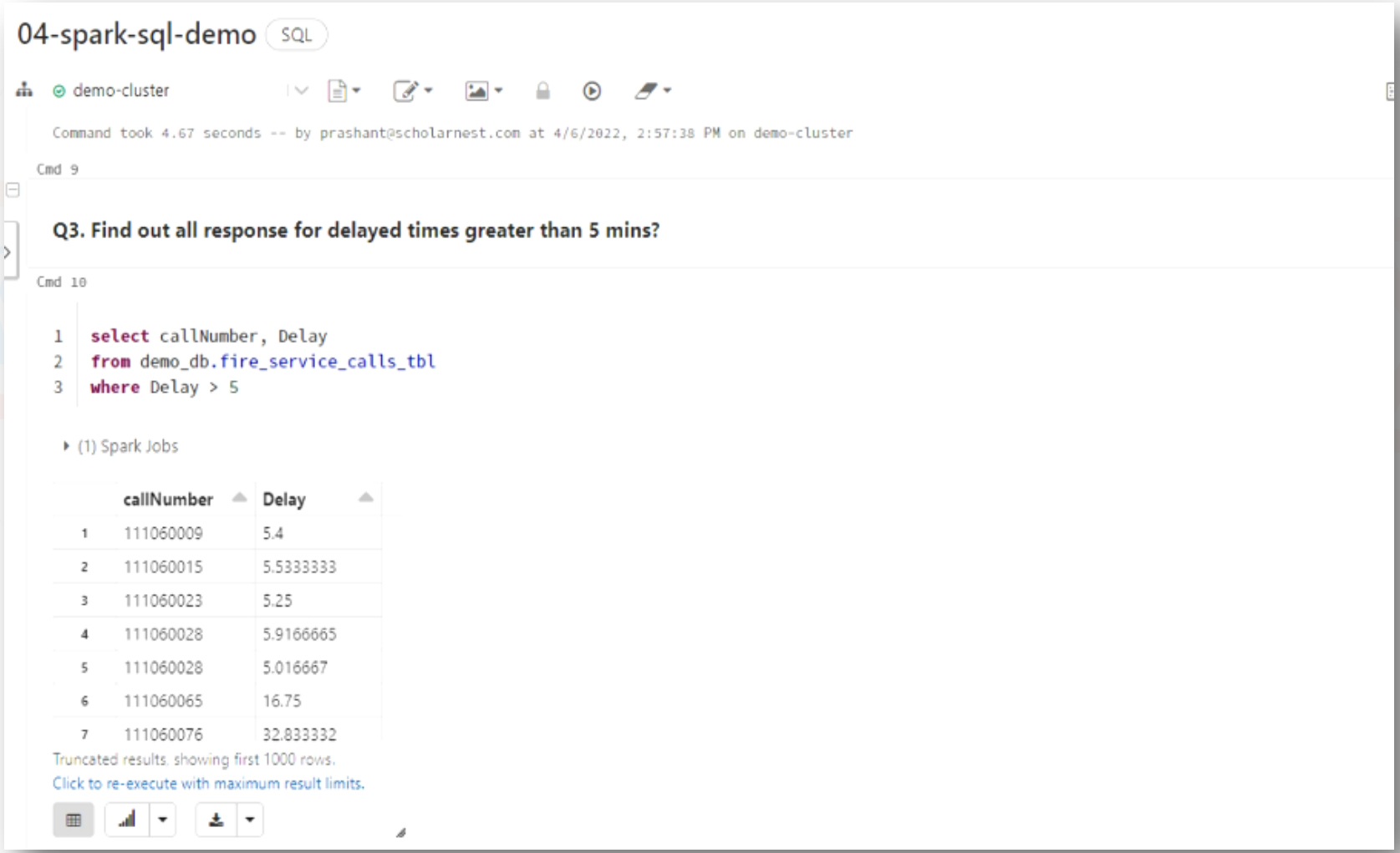
Let us look at the first question.
They are asking for how many distinct types of calls were made. So the solution is to take count of distinct call types. And we have the SQL code which says *select distinct call type* and then take a *count*. I added one extra condition to eliminate null values in the cell type.
And we got the expected output below the cell. We have 32 unique call types made to the San Francisco Fire station.



Let us look at the second question.
So the question is almost the same as the first one, but this time, we need a list of distinct call types.
Earlier, we wanted to count distinct call types, but now we want the list.
We have the code for the same shown below. I removed the count function from the select expression we used in the first question, and we can see the expected results.

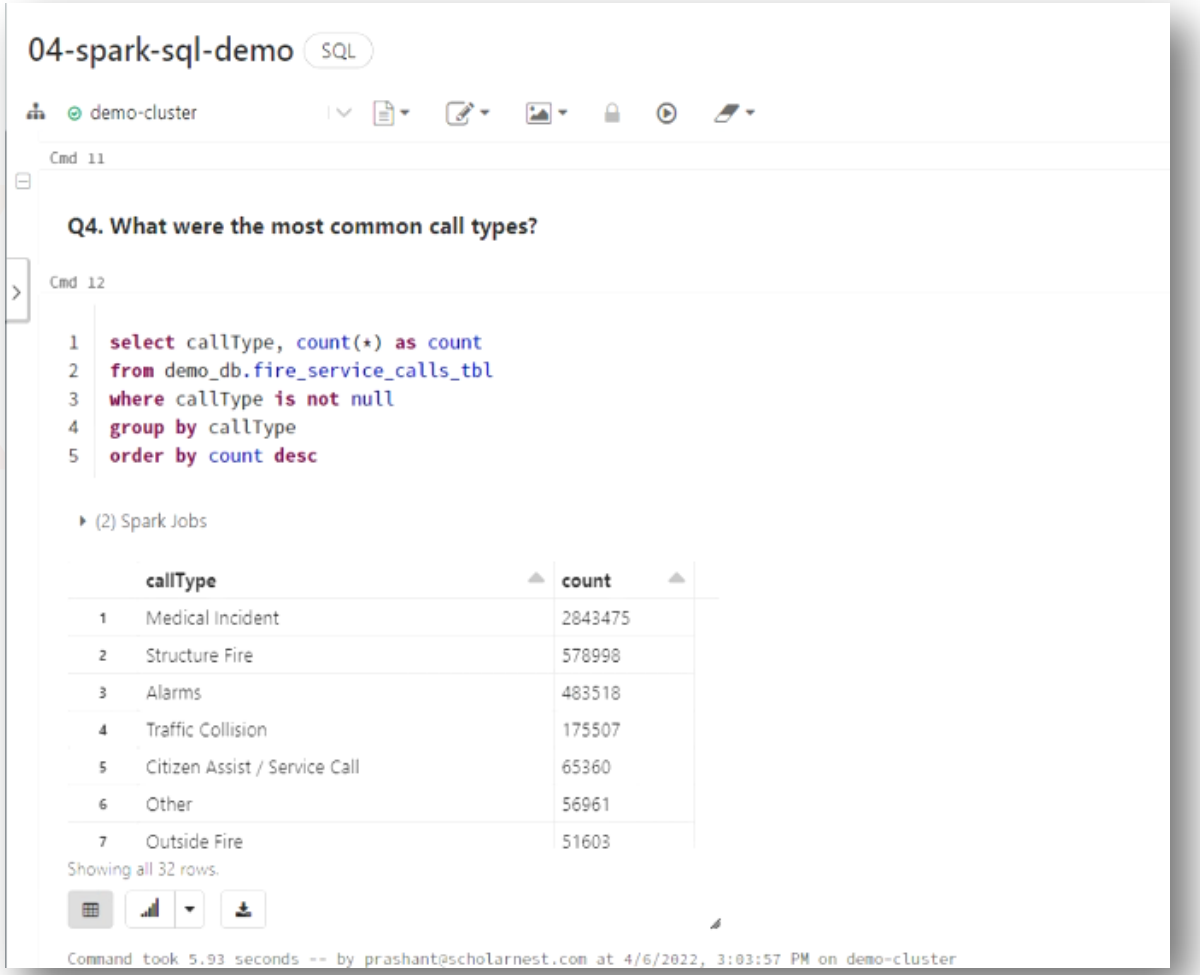


Let us look at the third question.
We want to list the call numbers where the delay is greater than 5 minutes.
We have the code and desired results shown below.

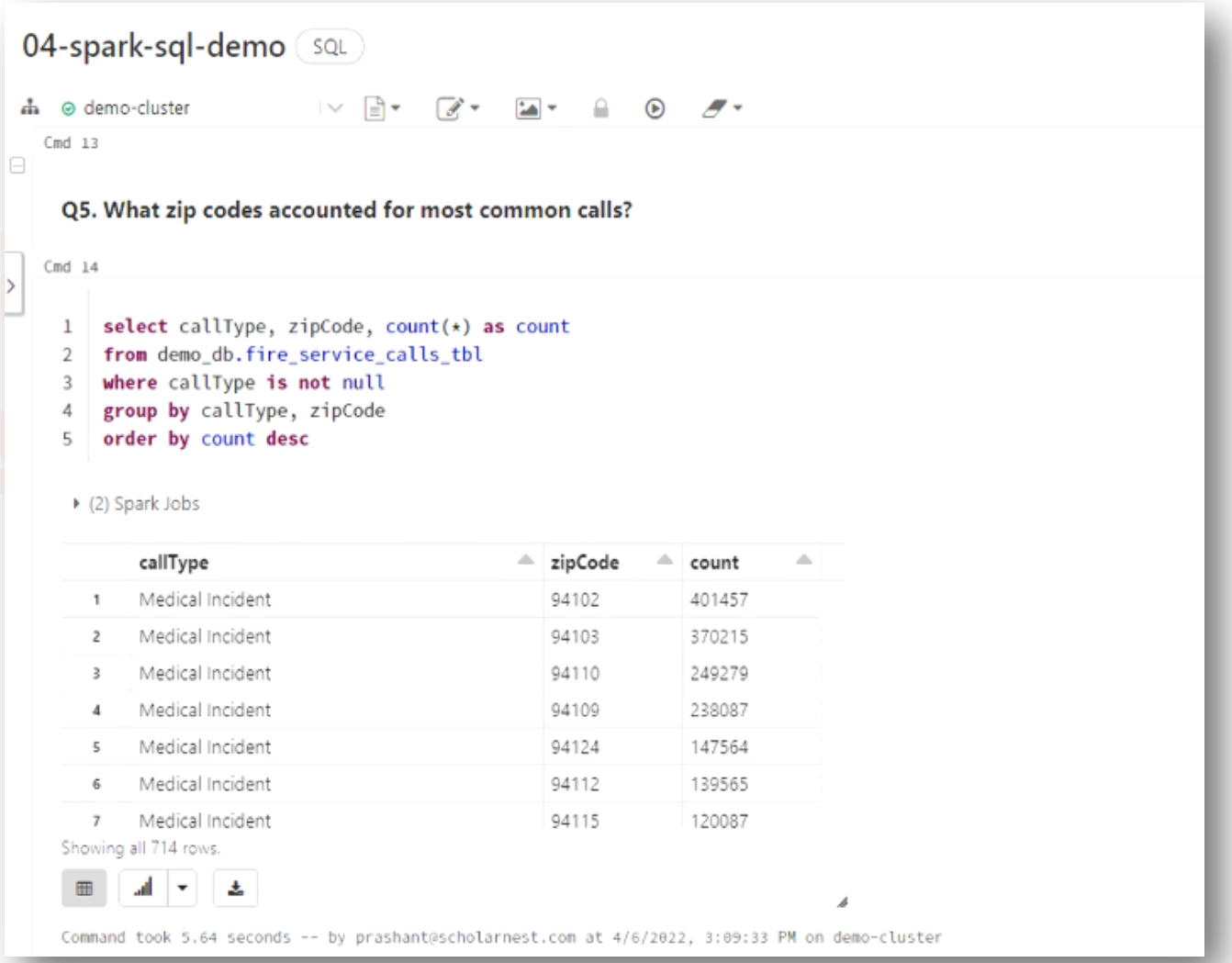


Let us look at the fourth question.

We have to find the most common call types. So, we need to take a count on the call type and sort the result in descending order, and we will get the most common calls at the top. I have written the code for the same and you can see the most common types in sorted order.

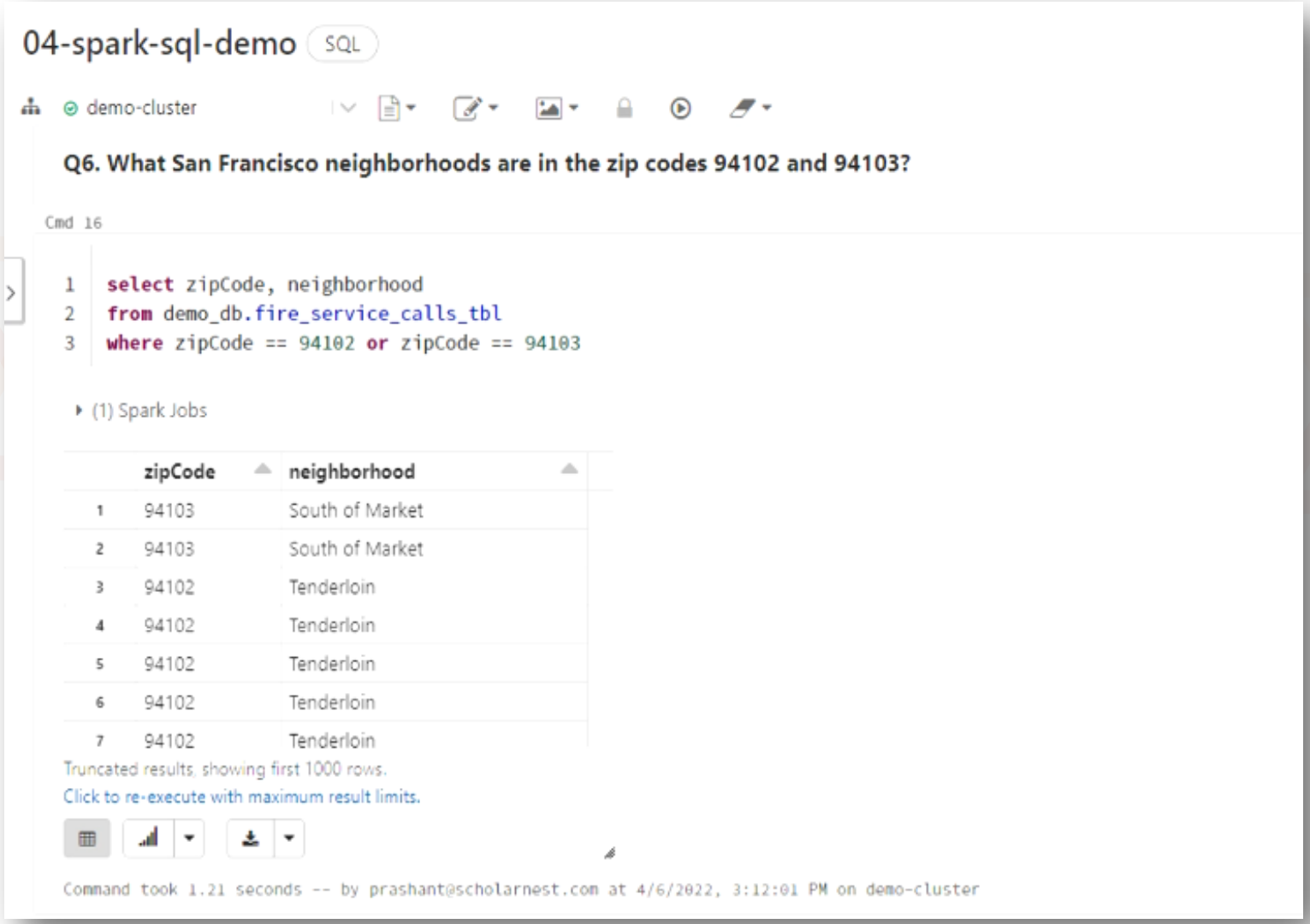


Let us look at the fifth question.
Here we want to find the most common call types, but we also want to see the zip code for the most common call types. I can do that by adding a zip code column in my group by clause.



Let us look at the sixth question.

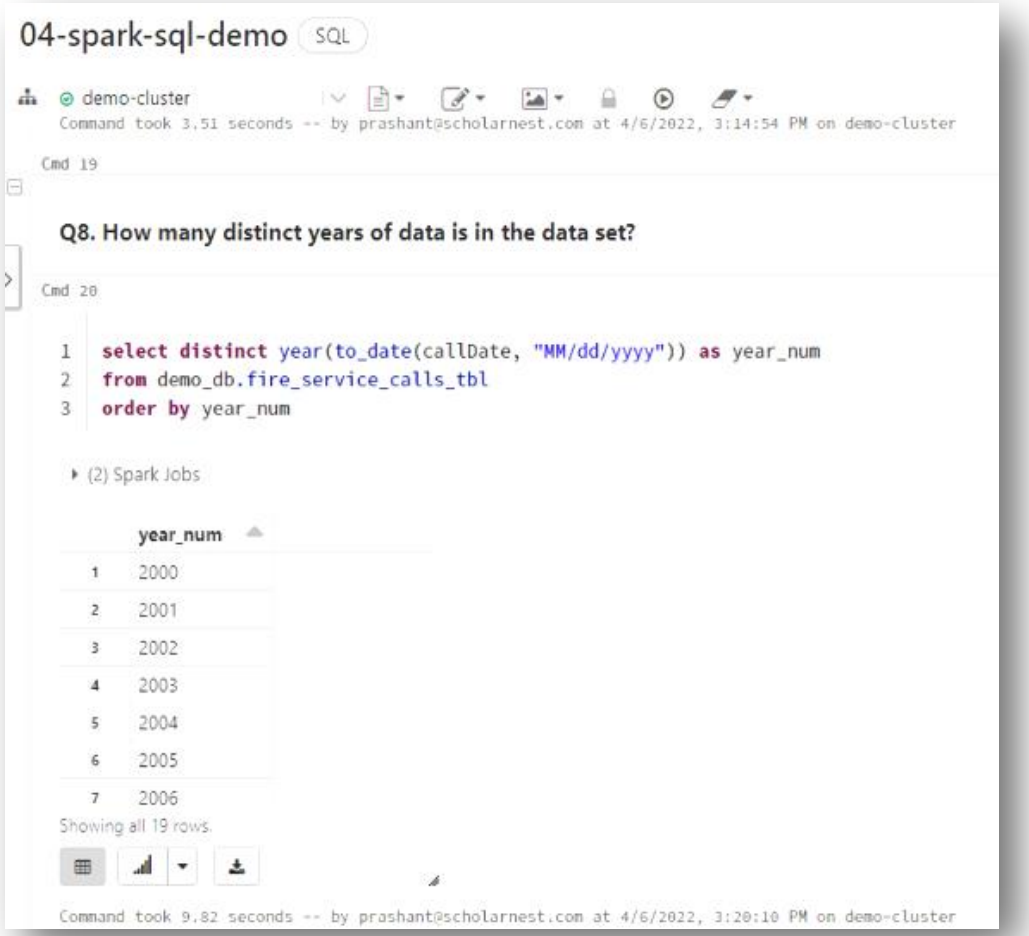
This question wants us to filter the records for two zip codes - 94102 and 94103. Then we want to see the names of unique neighbourhoods in these zip codes.



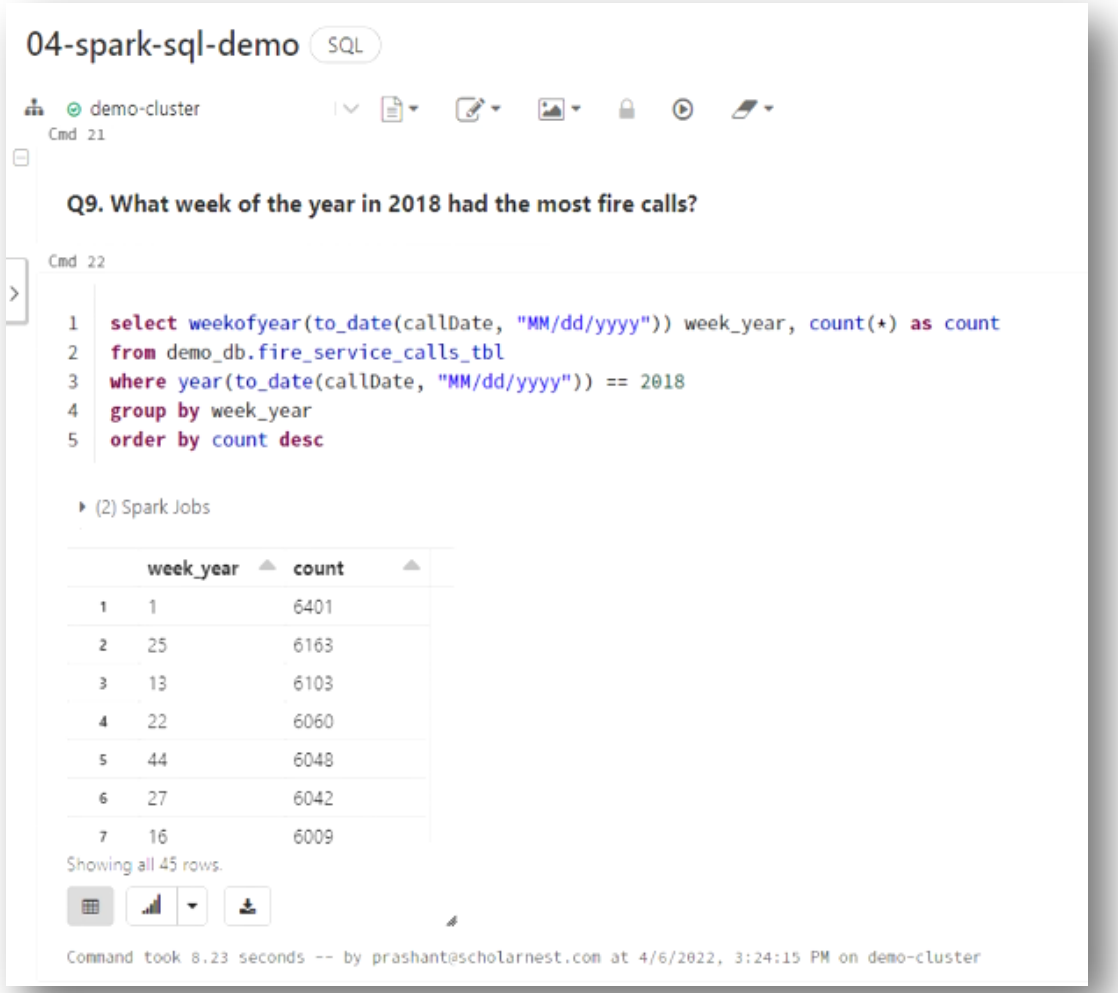
Let us look at the seventh question.

The first part of the question wants us to find the sum of *NumAlarms*. The second part of the question wants us to find avg, min, and max response time delay. And we want to find it for all the calls. So no filter and no grouping.

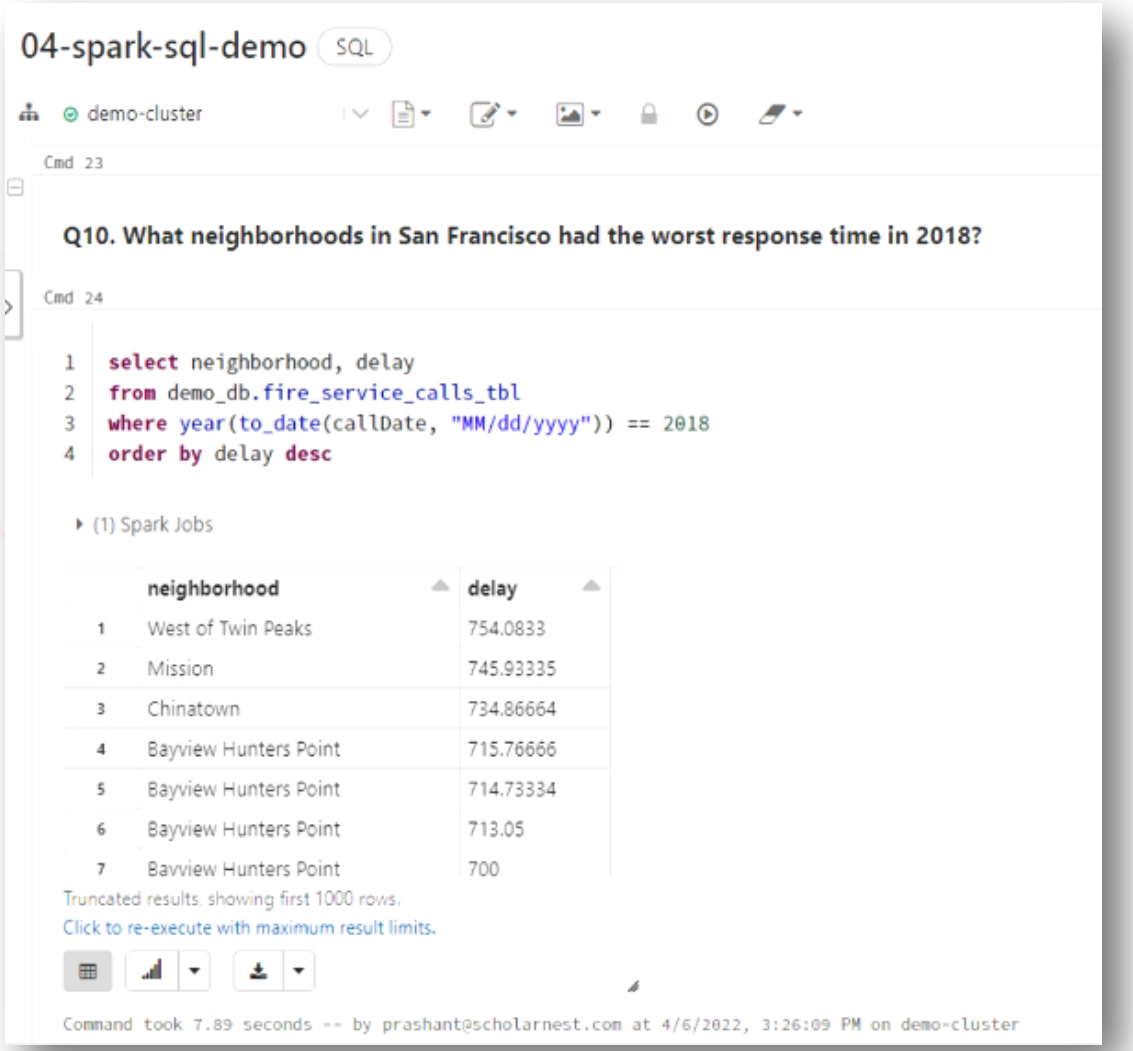
Let us look at the eighth question.
They want us to find the number of distinct years in the data set. We have a Call Date field, and we can take out the year from that field. However, the Call Date is a string field.
But we can convert it to a date field and take out the year. Once we have the year number, we can apply a distinct clause to find unique years.



Let us look at the ninth question.
They want us to filter the table for the year 2018. We can convert the CallDate into a date filed, take out the year and filter it for 2018. Then we can take out the week number from the CallDate and count it by week number. Once counted, you can sort it in descending order to find the most calls.



Let us look at the tenth question.
We just need to filter records for the year 2018. Then look for the neighbourhoods and delay columns. And finally, sort it by delay in descending order.



Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

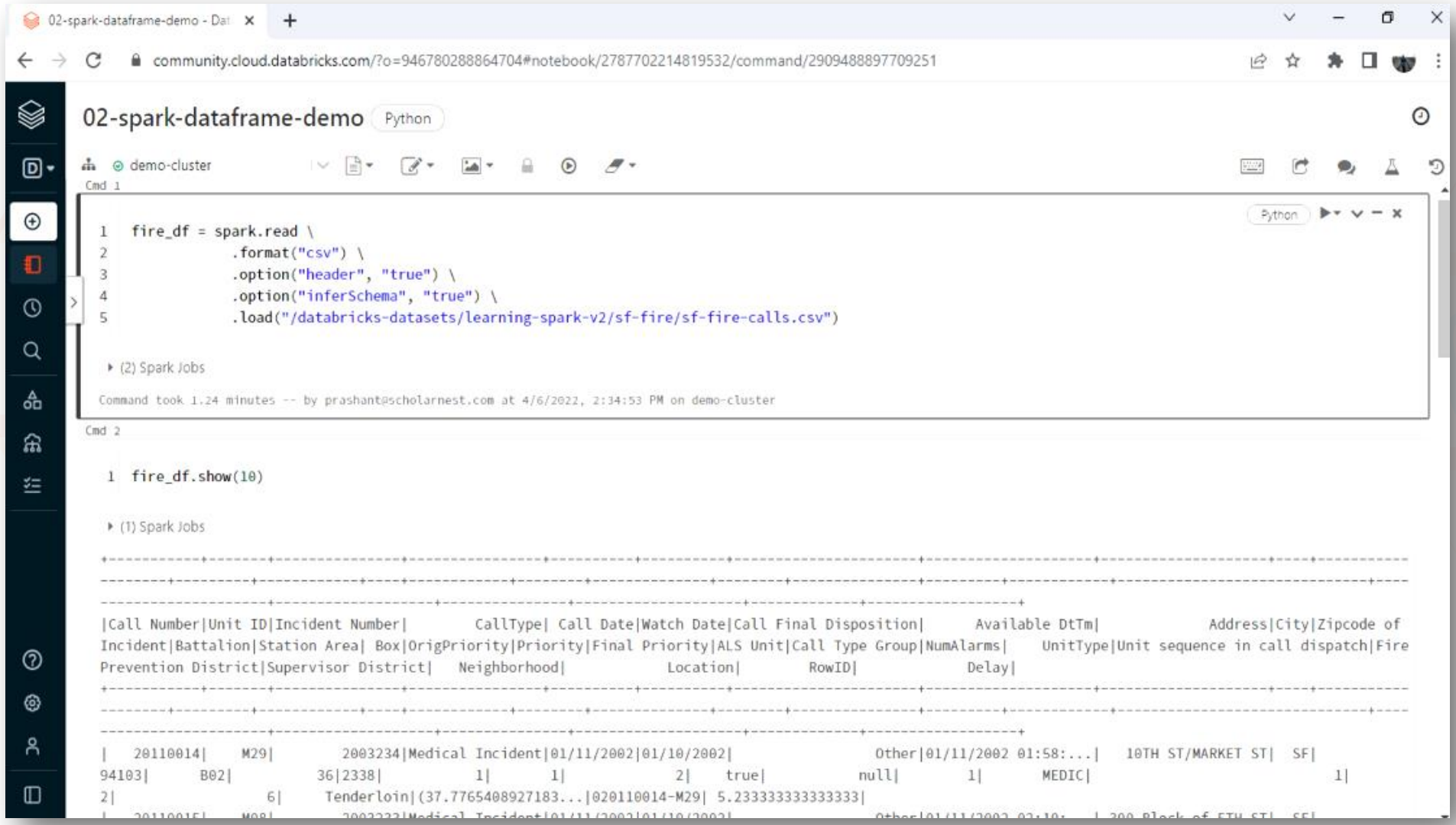
Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

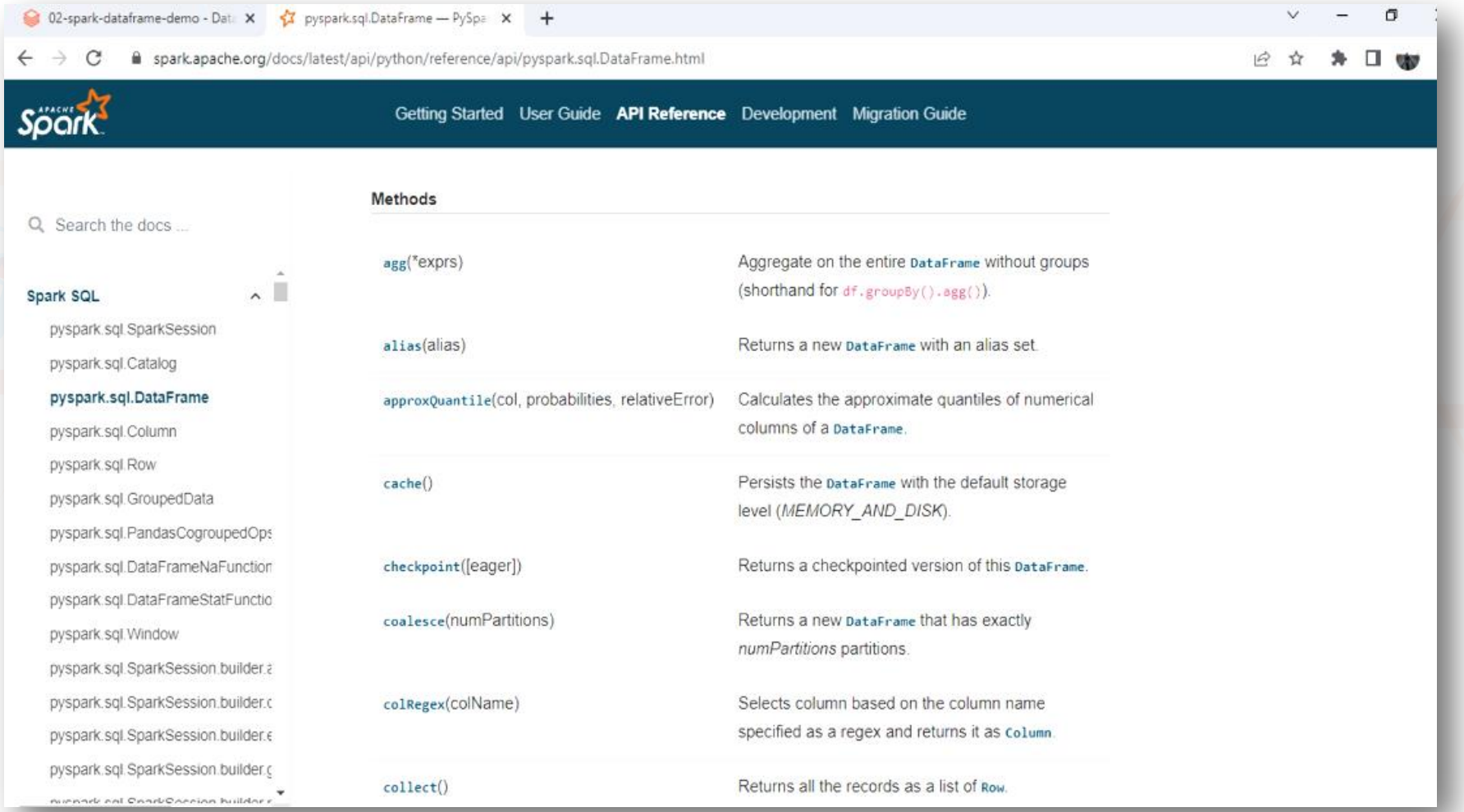
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Dataframe Methods

Go to your Databricks workspace and open spark-dataframe-demo notebook. **(Reference: 02-spark-dataframe-demo.ipynb)**



Now that we have created a Dataframe, we can apply Dataframe methods to process the data. You can visit https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html to get the list of all the Dataframe methods.

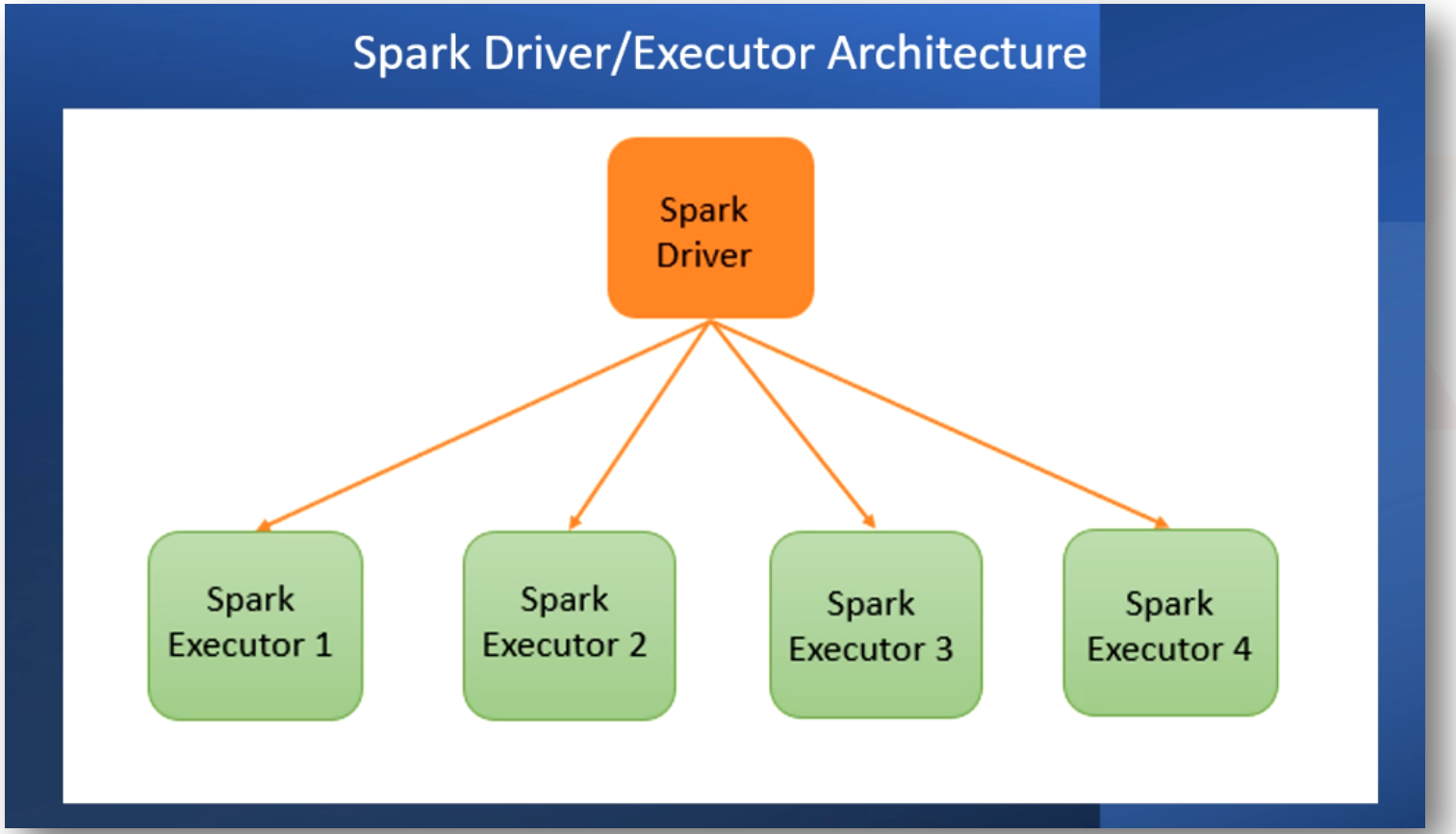


Dataframe offers you a long list of methods. However, we can classify them into three logical groups:

1. Actions
2. Transformations
3. Functions/Methods

Actions: Actions trigger a Spark Job and return to the Spark driver.
Spark is a distributed processing system. So every Spark program runs as one driver and one or more executors
And Actions are Dataframe operations that performs two things:

1. Kick off a Spark Job
2. Return to the Driver



Transformations: Spark Dataframe transformation produces a newly transformed Dataframe.
Transformations are different than actions. They do not kick off a Spark Job, and they do not return to the driver.
They simply create a new Dataframe and return it.

Functions and Methods : These are DataFrame methods or functions which are not categorized into Actions or Transformations.
So everything other than the Actions and Transformations is a method or function.

Here is a list of Dataframe Actions and Dataframe Transformations.
Spark offers you 12 actions as on date. This list may update with a newer version of Spark.
We have 38 transformations.

Actions

1. collect
2. count
3. describe
4. first
5. foreach
6. foreachPartition
7. head
8. show
9. summary
10. tail
11. take
12. toLocalIterator

Transformations

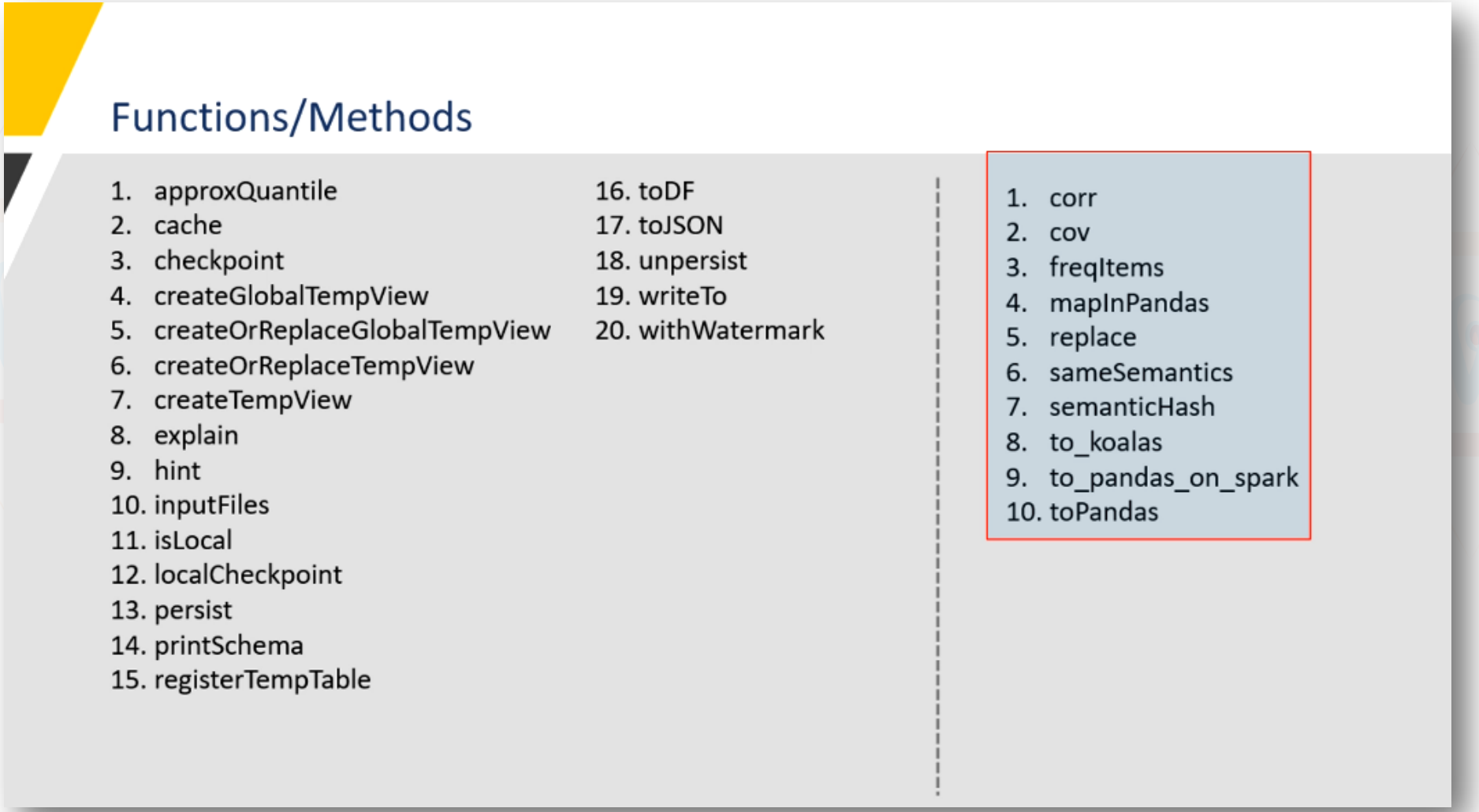
1. agg
2. alias
3. coalesce
4. colRegex
5. crossJoin
6. crosstab
7. cube
8. distinct
9. drop
10. drop_duplicates
11. dropDuplicates
12. dropna
13. exceptAll
14. filter
15. groupby
16. intintersect
17. ersectAll
18. join
19. limit
20. orderBy
21. randomSplit
22. repartition
23. repartitionByRange
24. rollup
25. sample
26. sampleBy
27. select
28. selectExpr
29. sort
30. sortWithinPartitions
31. subtract
32. transform
33. union
34. unionAll
35. unionByName
36. where
37. withColumn
38. withColumnRenamed

Here is a list of Dataframe Functions and Methods.
We have 20 functions and methods. These are utility functions or methods.

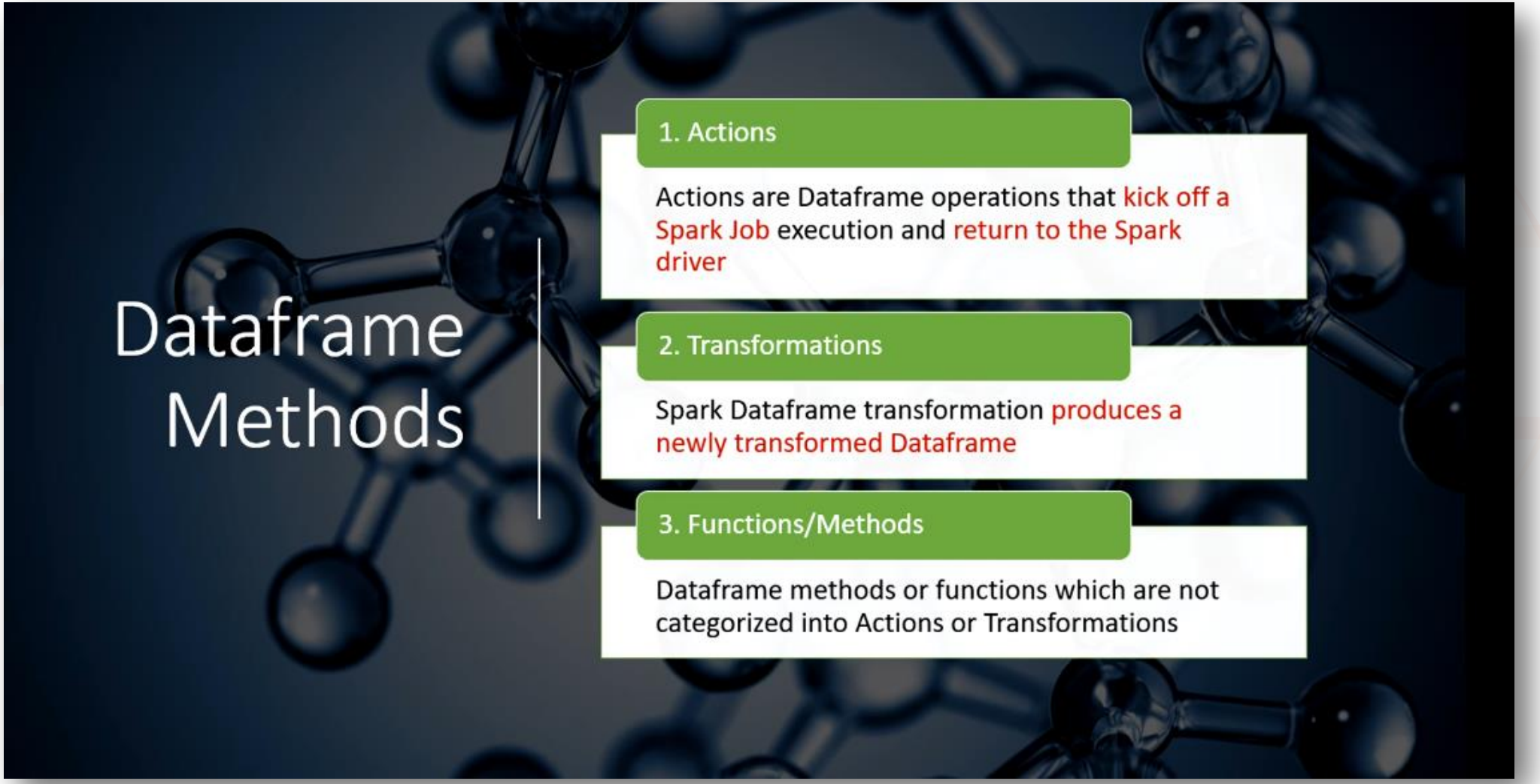
Functions/Methods

1. approxQuantile
2. cache
3. checkpoint
4. createGlobalTempView
5. createOrReplaceGlobalTempView
6. createOrReplaceTempView
7. createTempView
8. explain
9. hint
10. inputFiles
11. isLocal
12. localCheckpoint
13. persist
14. printSchema
15. registerTempTable
16. toDF
17. toJSON
18. unpersist
19. writeTo
20. withWatermark

There are ten more functions that I excluded from the list that I have highlighted in the image below. I have excluded these because those functions are only available in Python and not in other languages, and it is not frequently used.



Finally, here is the definition of all the three types of Dataframe Methods.



Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

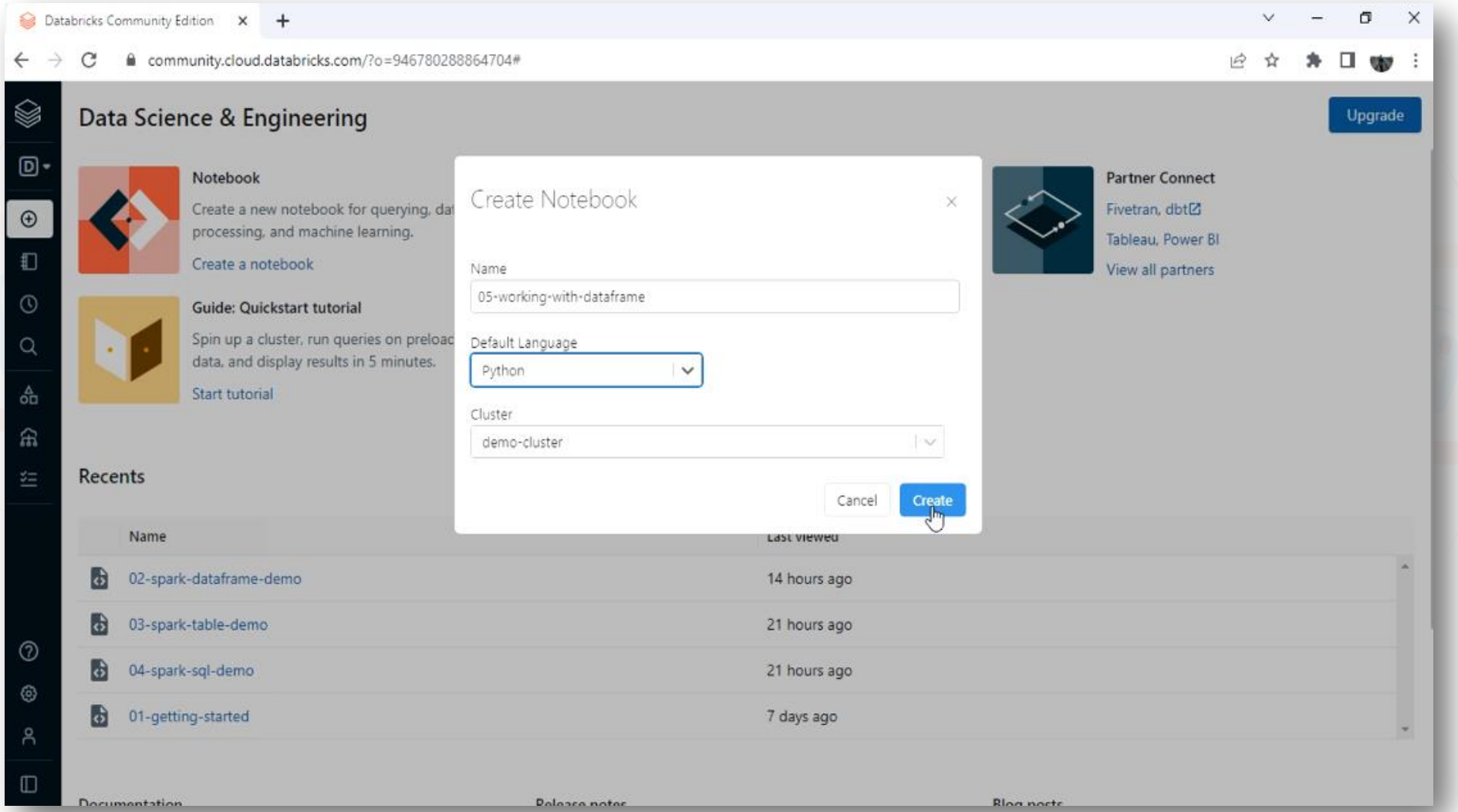
Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

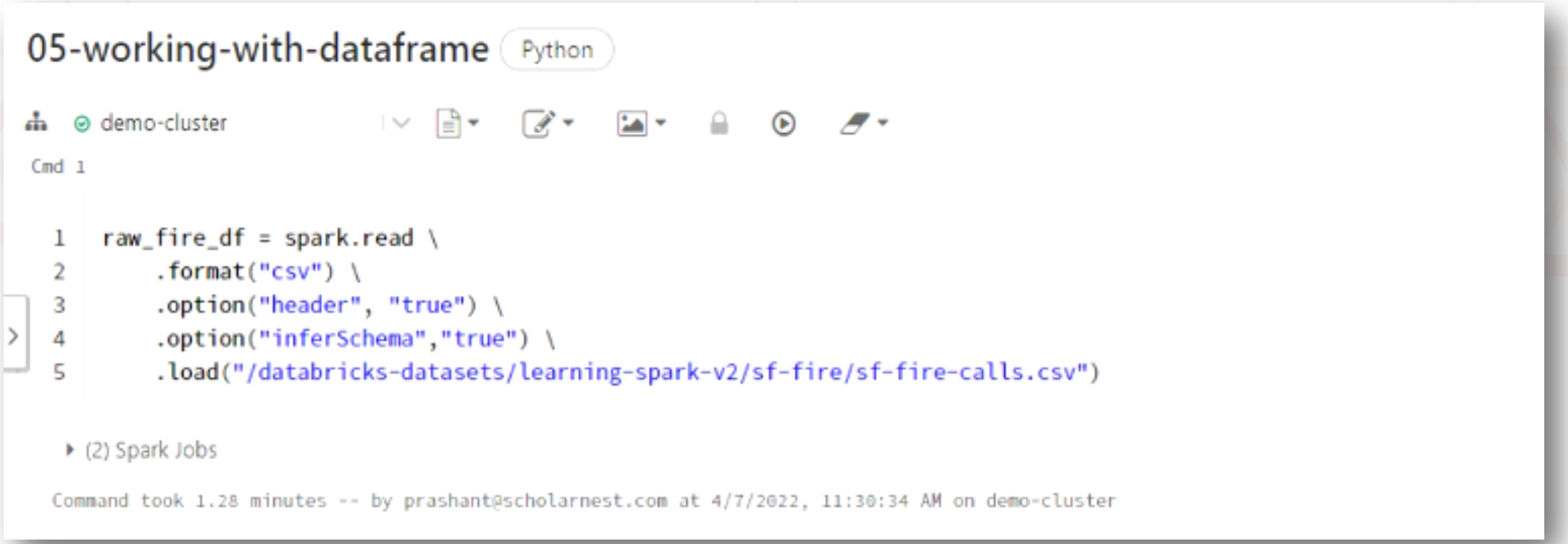
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Applying Dataframe Transformations

Go to your Databricks workspace and create a new notebook as given below. **(Reference: 05-working-with-dataframe.ipynb)**

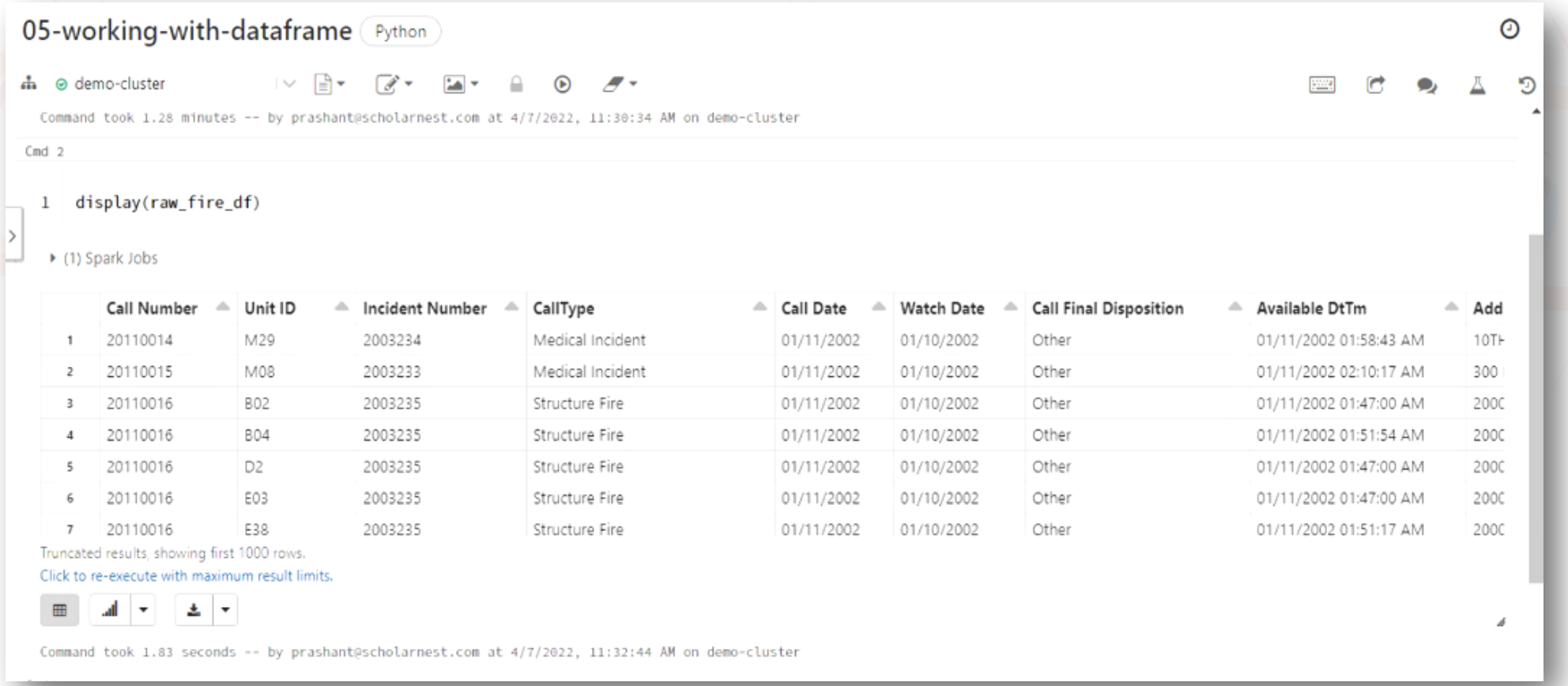


Here we the older code which we used earlier to create a data frame for the fire call response data set.



If we display the older raw_fire_df Dataframe and see the results as given below. We can see there are two problems in this Dataframe:

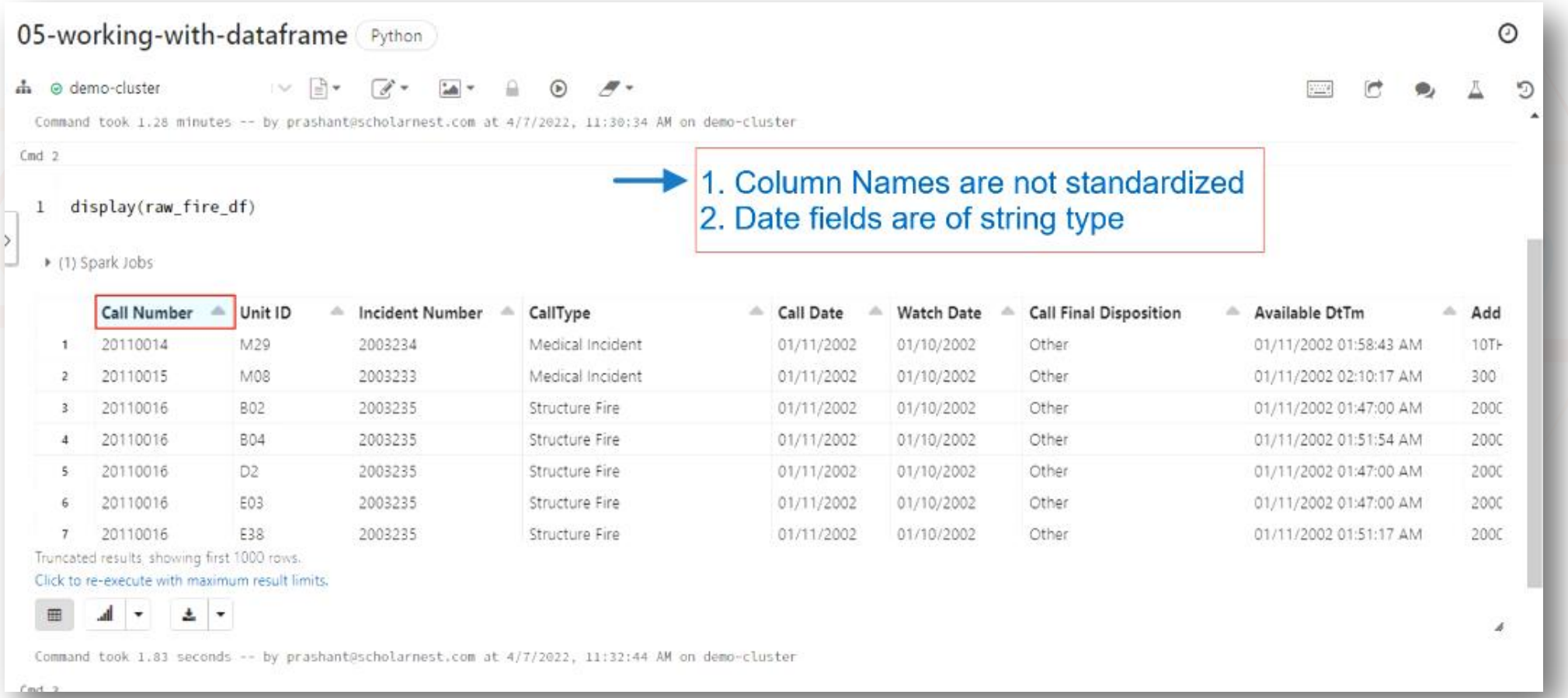
1. Column Names are not standardized
2. Date fields are of string type



	Call Number	Unit ID	Incident Number	CallType	Call Date	Watch Date	Call Final Disposition	Available DtTm	Add
1	20110014	M29	2003234	Medical Incident	01/11/2002	01/10/2002	Other	01/11/2002 01:58:43 AM	10TH
2	20110015	M08	2003233	Medical Incident	01/11/2002	01/10/2002	Other	01/11/2002 02:10:17 AM	300
3	20110016	B02	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	200C
4	20110016	B04	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:51:54 AM	200C
5	20110016	D2	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	200C
6	20110016	E03	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	200C
7	20110016	E38	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:51:17 AM	200C

Column Names are not standardized:

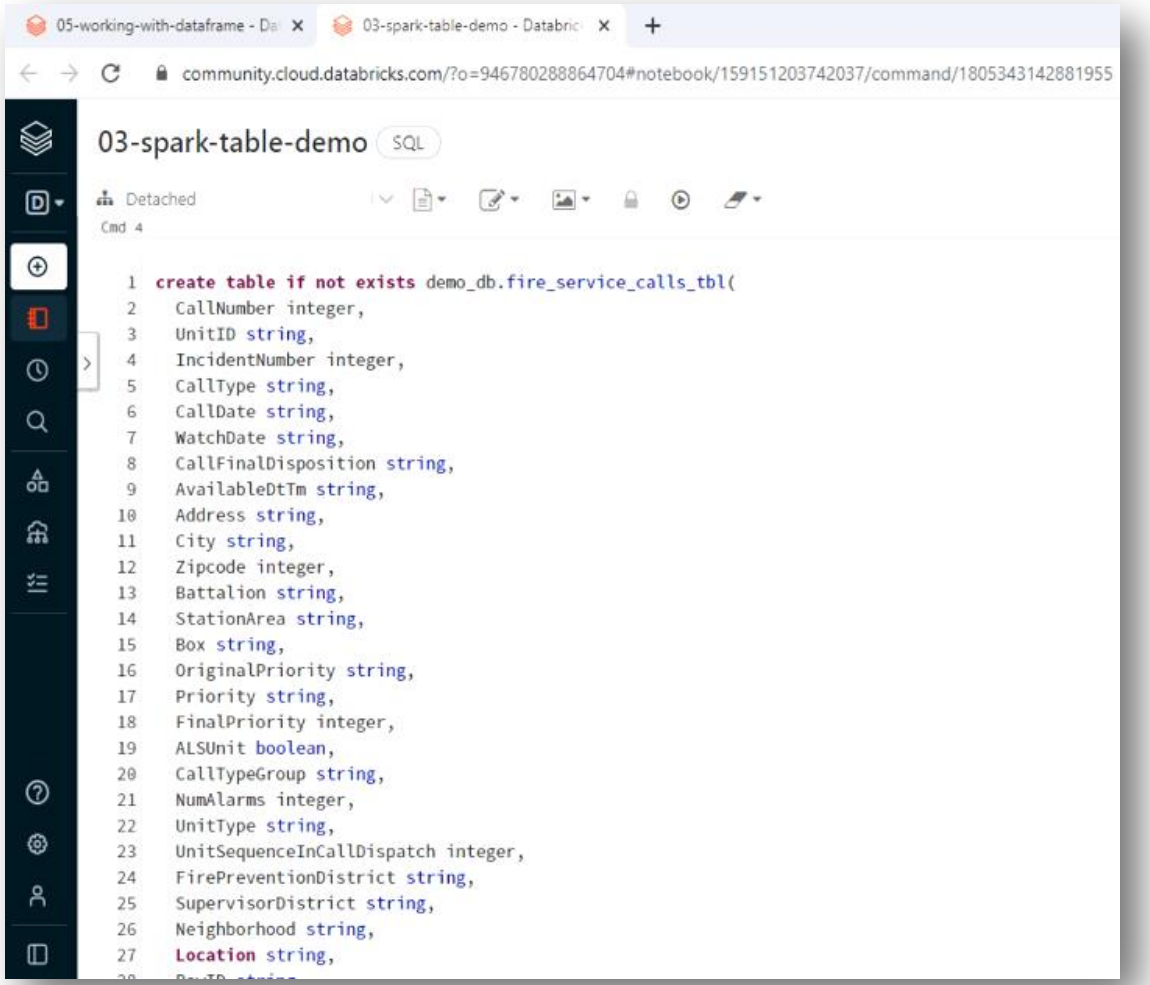
You can see a blank space between call and number. That's not standard. Spark Dataframe column names are case insensitive. However, space in a column name is not a common thing. Let me fix these column names and rename them to remove spaces.



Column Names are not standardized:

We fixed this problem while creating the table. **(Reference: 03-spark-table-demo.sql)**

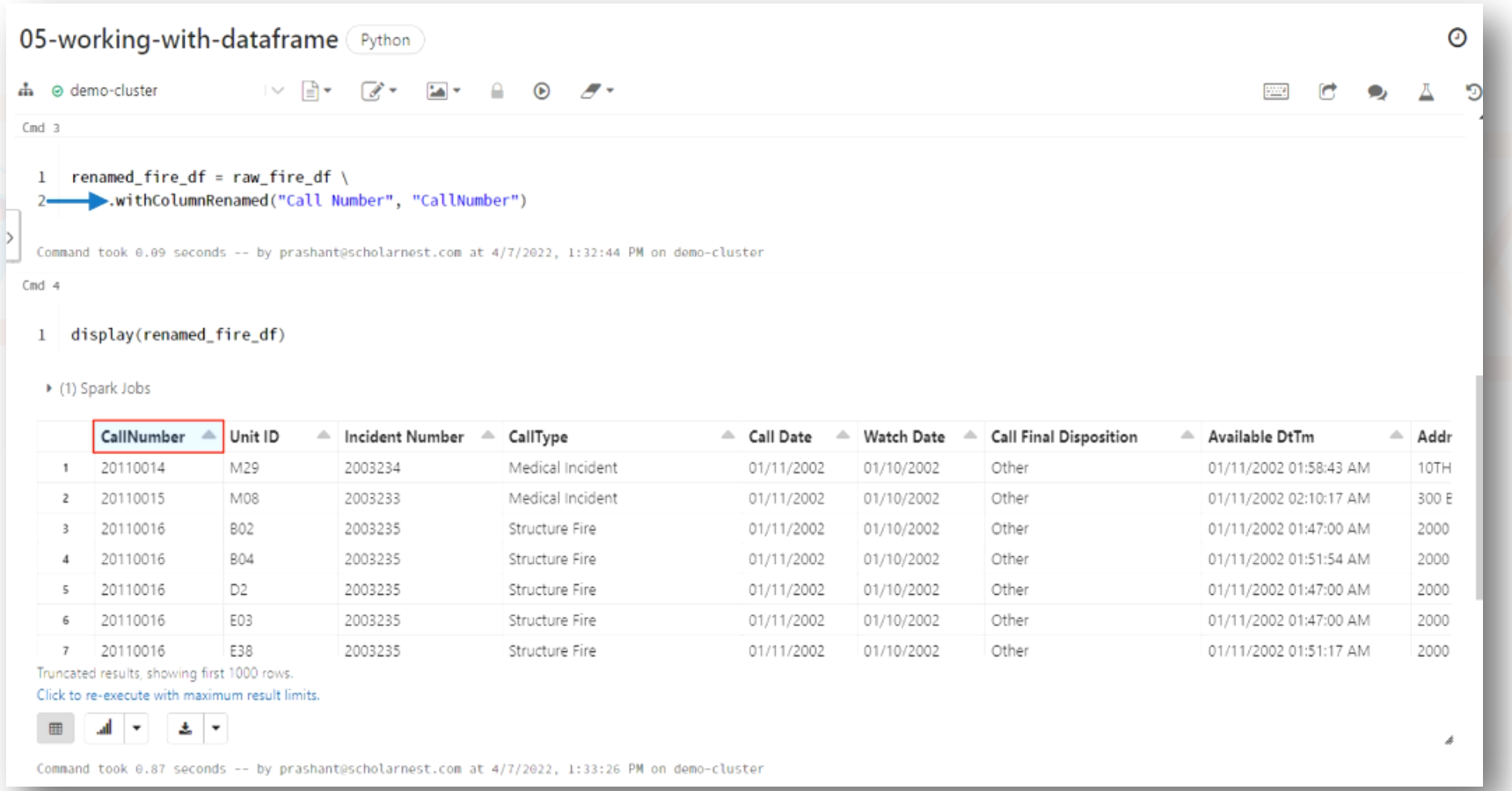
You can see in the image below that we created column names without spaces when we created the table. Then we loaded data into this table. So we already fixed the column names while creating a table.



Column Names are not standardized:

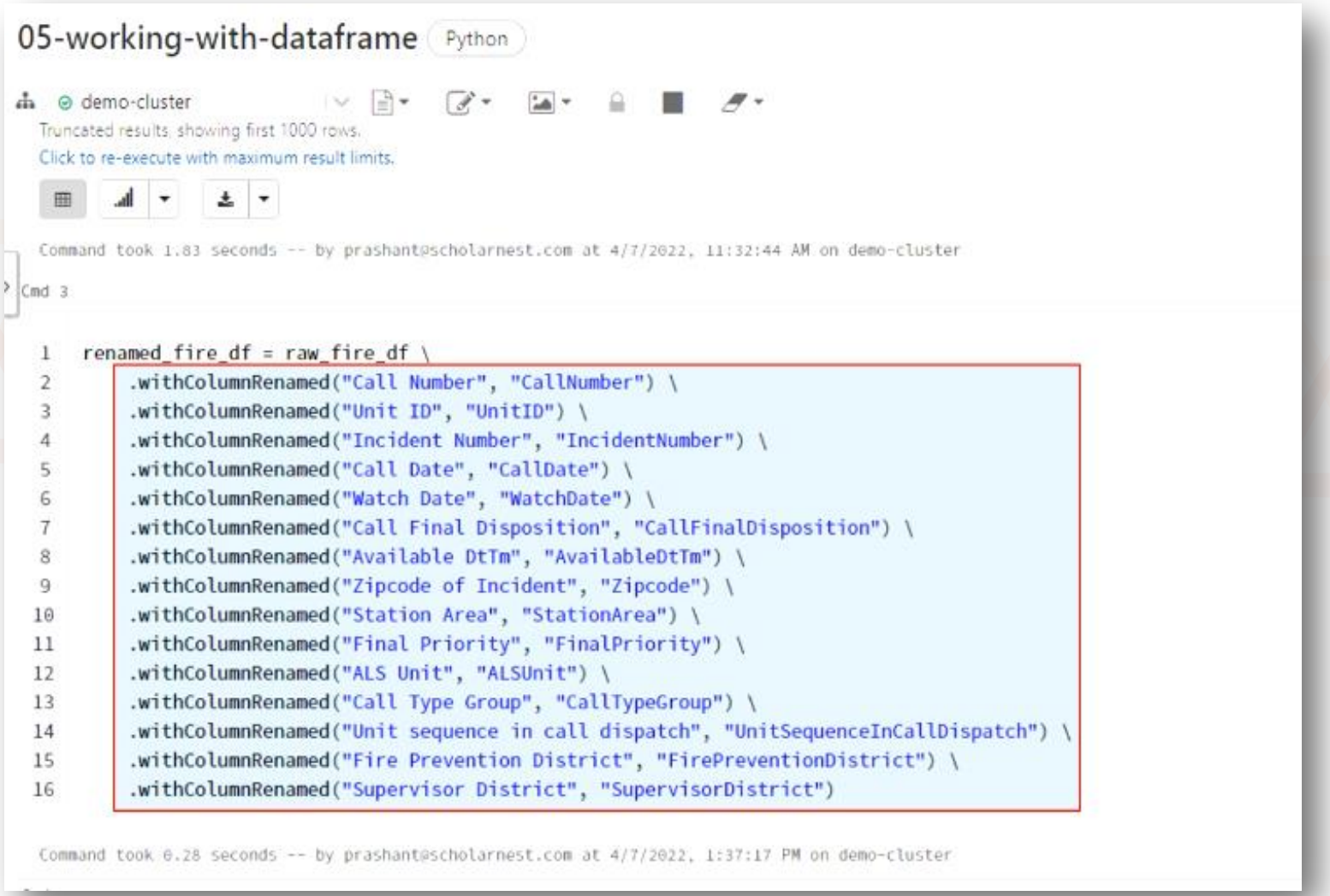
Here is how we can fix the column names while working on Dataframes. **(Reference: 05-working-with-dataframe.ipynb)**

Spark Dataframe offers you withColumnRenamed() transformation to rename a column. You can see in the screenshot below that withColumnRenamed() takes two arguments: Old column name and New column name. And we can see the desired results when we are displaying the Dataframe below.



Column Names are not standardized:

Similarly, we can rename all the problem columns as shown in the code below. If we run this code and display the results, our first problem is resolved and our column names and standardized.



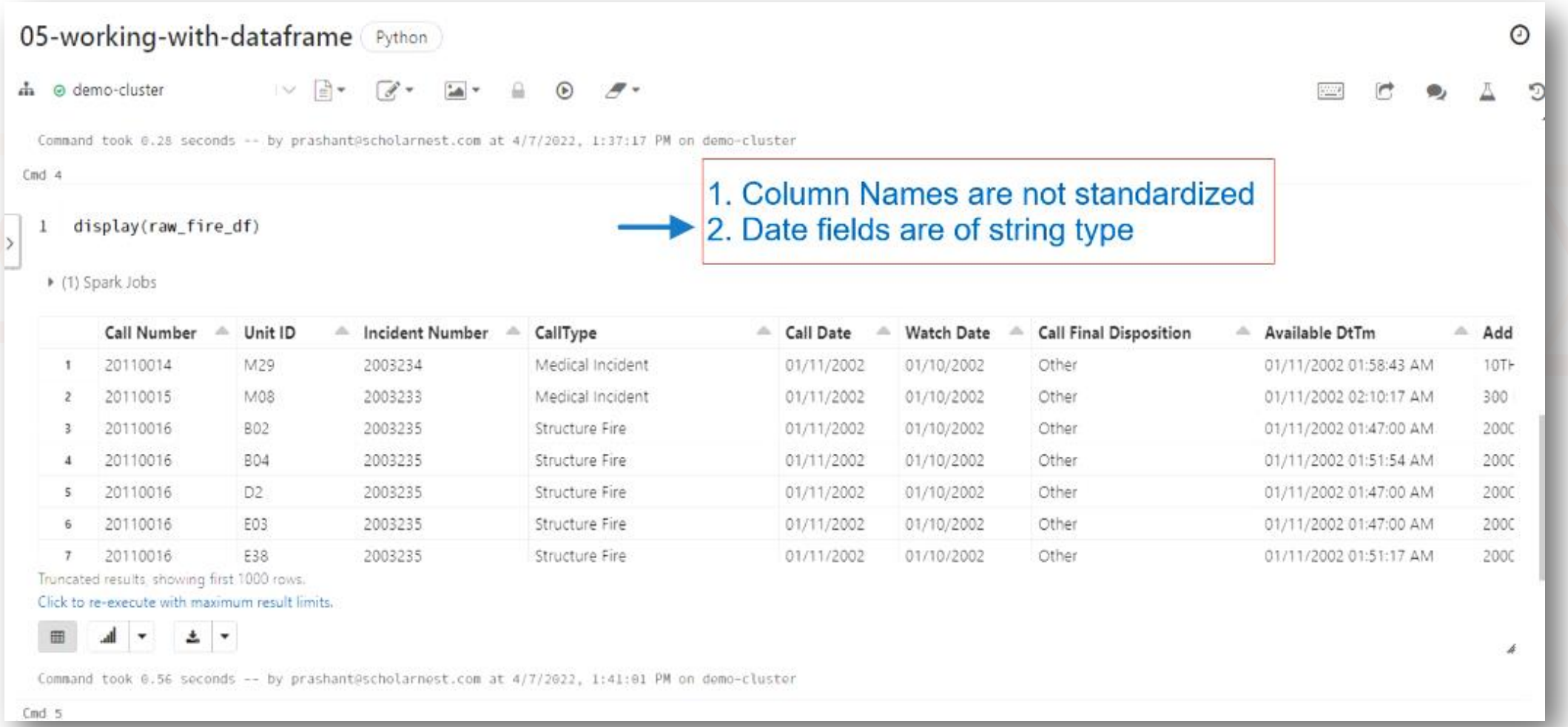
Spark Dataframe programming is all about doing two things:

1. Read your raw data into a Dataframe.
2. Transform your raw Dataframe and create a new transformed Dataframe.

We can also narrow down 3 key observations from what we have learned till now:

1. You can create a chain of Spark transformation methods one after the other.
2. Spark transformation returns a new Dataframe after transforming the old Dataframe.
3. Spark Dataframe is immutable. We cannot change or modify an existing Dataframe. Instead, we transform the existing Dataframe and create a new Dataframe.

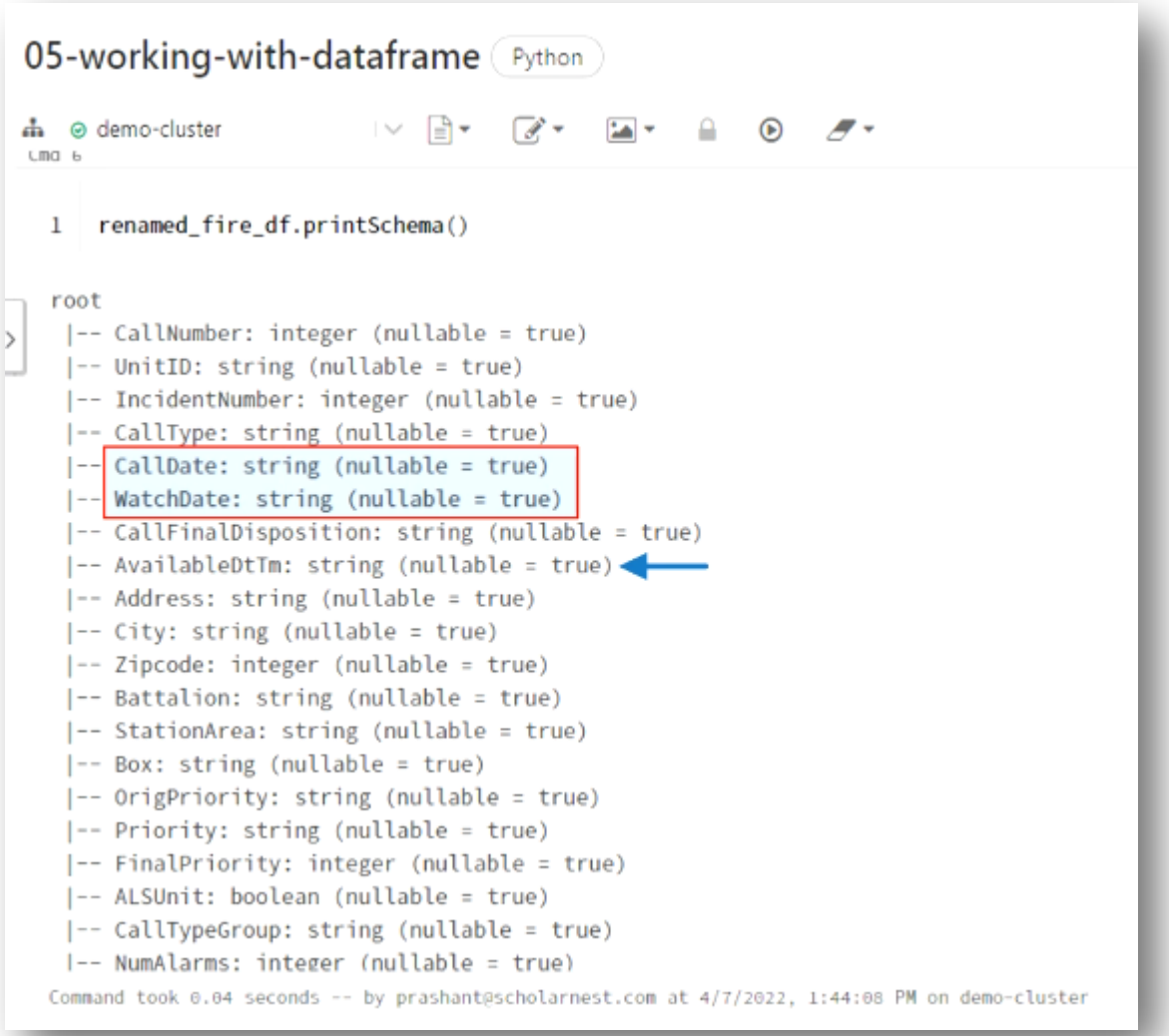
Now let us move ahead to the second problem.
You can see call date, watch date, and AvailableDtTm fields in the data set. The call date and watch date represent a date. And the AvailableDtTm represents a timestamp. However, these fields are of the type string, which is not desirable.



	Call Number	Unit ID	Incident Number	CallType	Call Date	Watch Date	Call Final Disposition	Available DtTm	Add
1	20110014	M29	2003234	Medical Incident	01/11/2002	01/10/2002	Other	01/11/2002 01:58:43 AM	10TH
2	20110015	M08	2003233	Medical Incident	01/11/2002	01/10/2002	Other	01/11/2002 02:10:17 AM	300
3	20110016	B02	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	2000
4	20110016	B04	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:51:54 AM	2000
5	20110016	D2	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	2000
6	20110016	E03	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:47:00 AM	2000
7	20110016	E38	2003235	Structure Fire	01/11/2002	01/10/2002	Other	01/11/2002 01:51:17 AM	2000

Date fields are of string type:

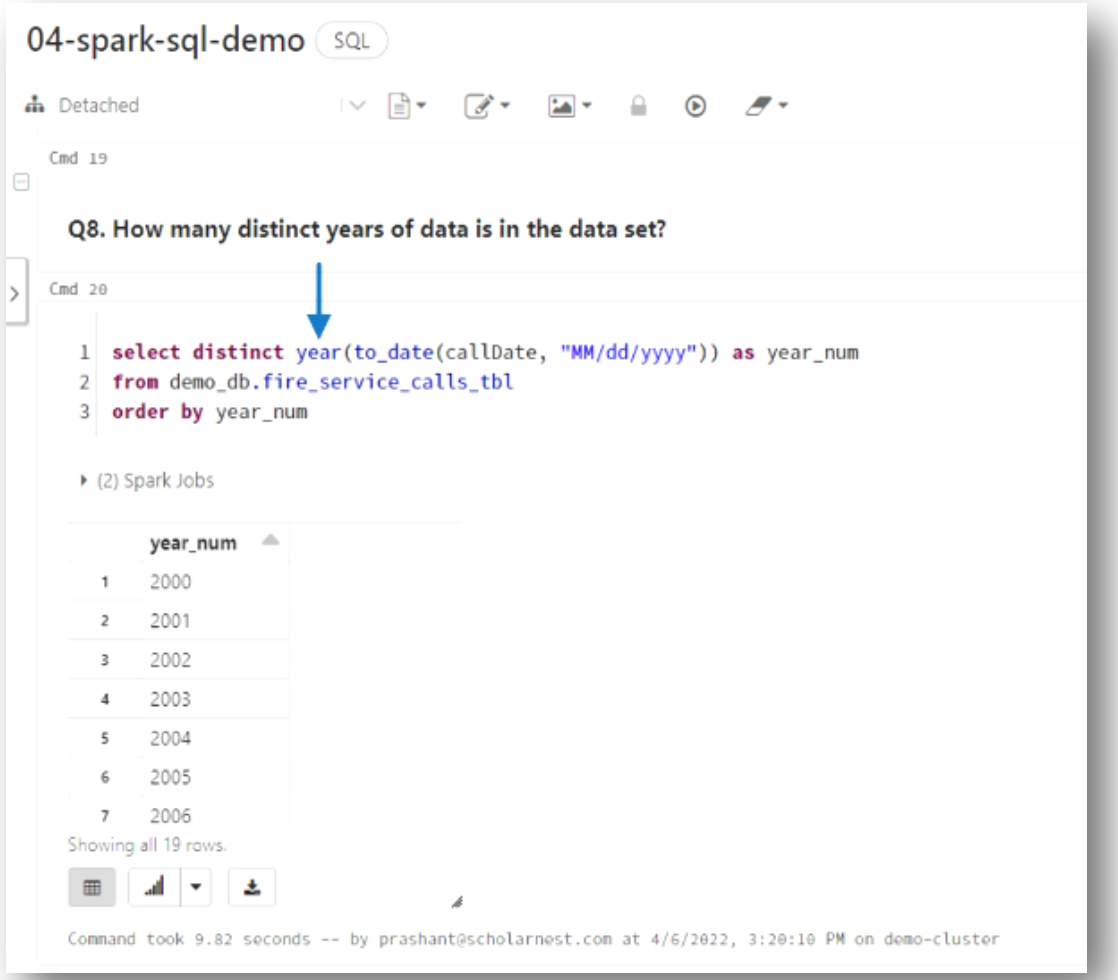
You can see the data type information using the printSchema() utility method. We have used this method on remaned_fire_df as we have transformed the older Dataframe into this new Dataframe by removing the problem column names. You can see that all the three fields are of the string type. Let me fix this problem.



Date fields are of string type:

We had this problem while working on spark-SQL-demo notebook. **(Reference: 04-spark-sql-demo.sql)**

You can see the solution of question number 8, that we converted the Call Date to a date filed. Then we applied the year() function. So, if your date and timestamp fields are stored in a string column, you may have to convert it to date on almost every use of the column.



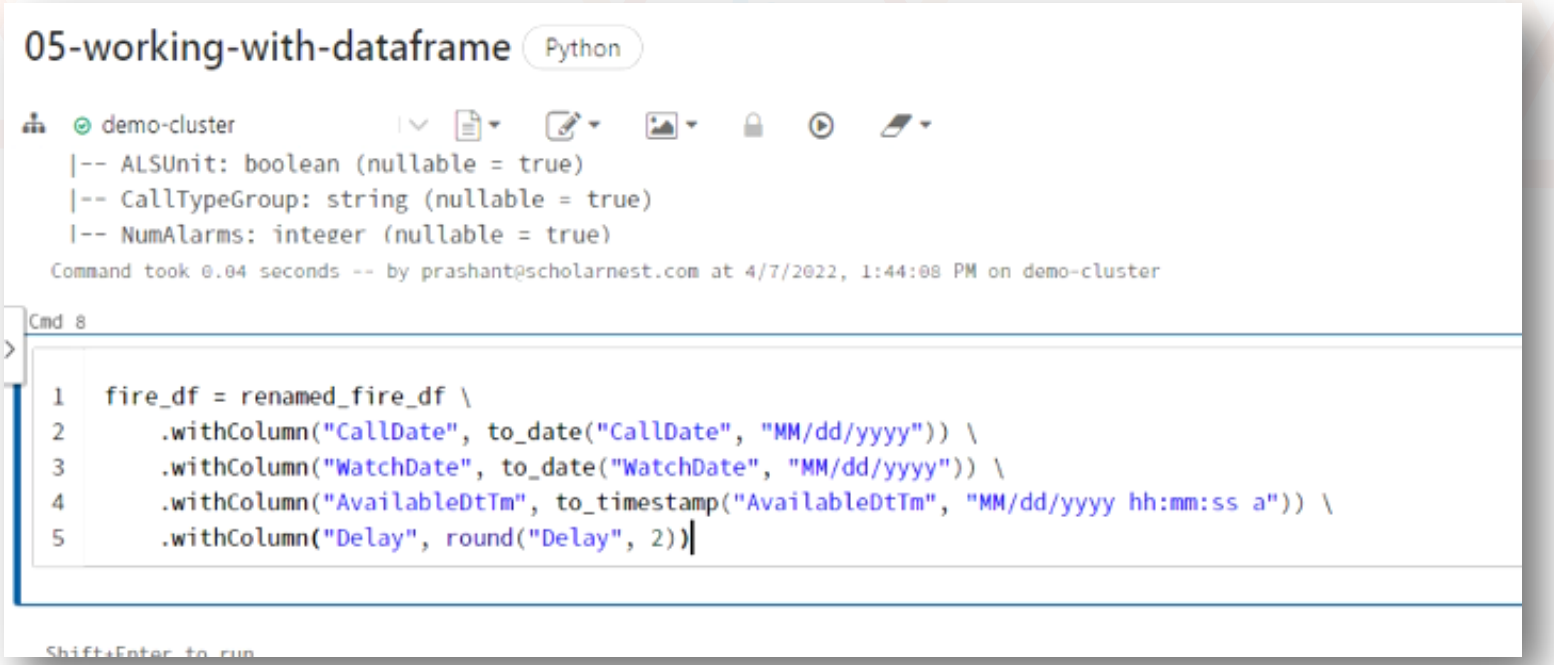
Date fields are of string type:

You can rectify this issue by transforming the *renamed_fire_df* to a new Dataframe *fire_df* by applying the *.withColumn()* transformation. The withColumn method takes two arguments:

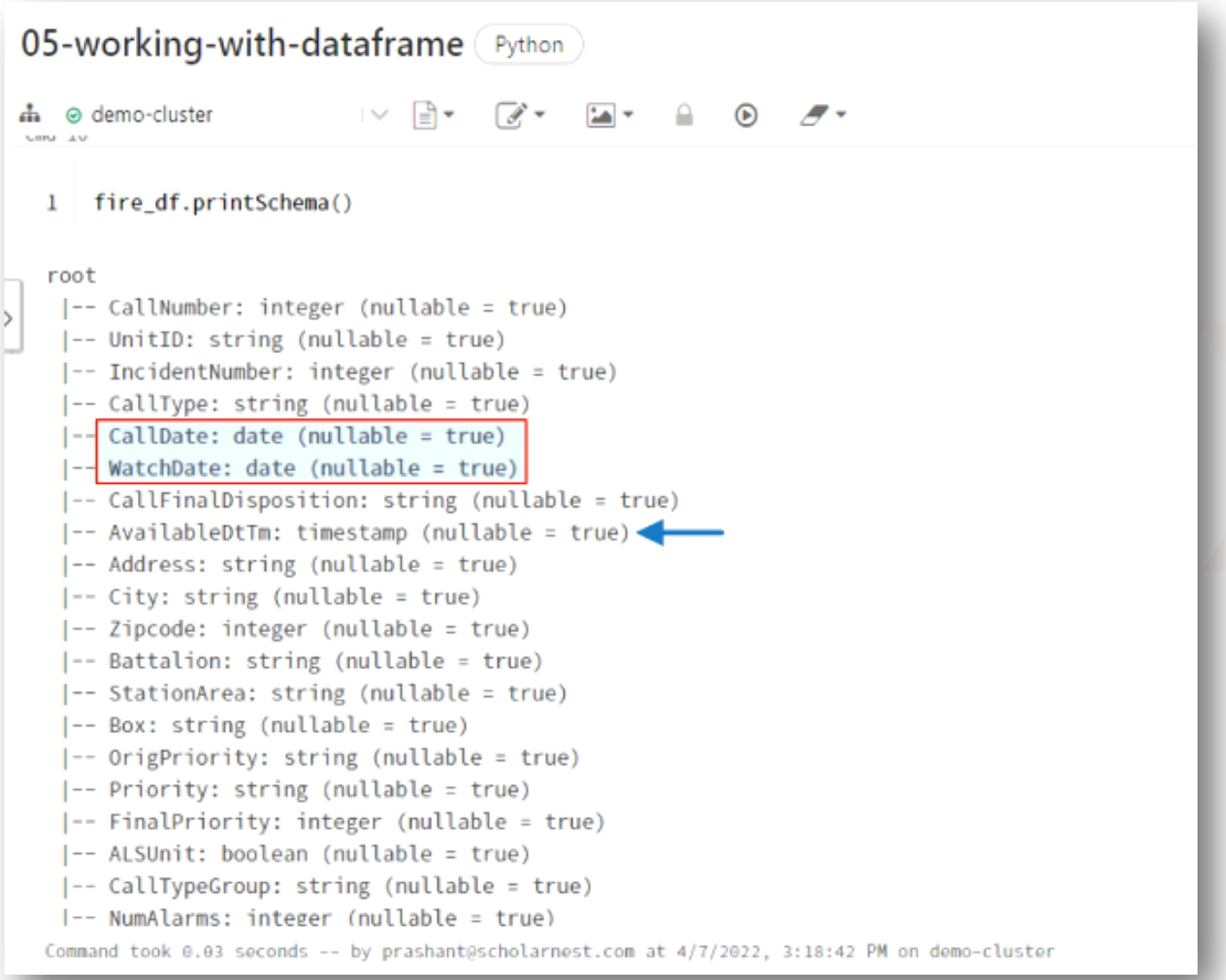
1. The first column is the one you want to transform. I want to transform the Call Date column, so I give the column name here.
2. The second argument is the logic to transform the given column. We want to convert the Call Date to a Date. So I am using the to_date() function. And the Call Date column in the remaned_fire_df is represented using MM/dd/yyyy string. Similarly, we have the corresponding logic for AvailableDtTm.

Make sure you import the to_date() and to_timestamp() function from the pyspark.sql.functions package.

We are also transforming the Delay column to round the Delay column for two digits.



We can print the schema of the new fire_df Dataframe and see the data types.
And we see the desired results in the output, the date and timestamp transformation is successful.



We learned about three new methods in this lecture.

Column Transformation Methods:

1. withColumnRenamed
2. withColumn

Utility Method:

1. printSchema()

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

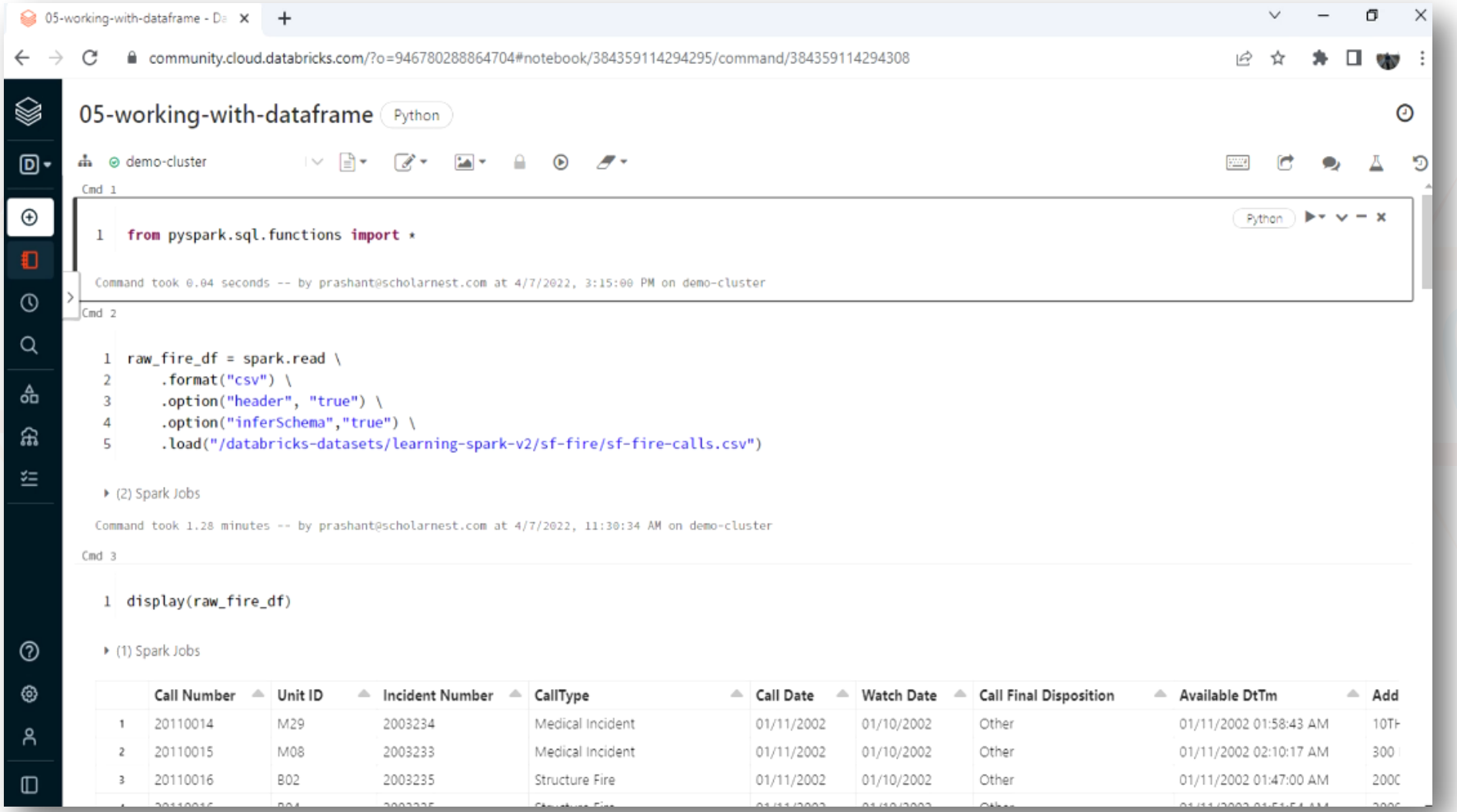
Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

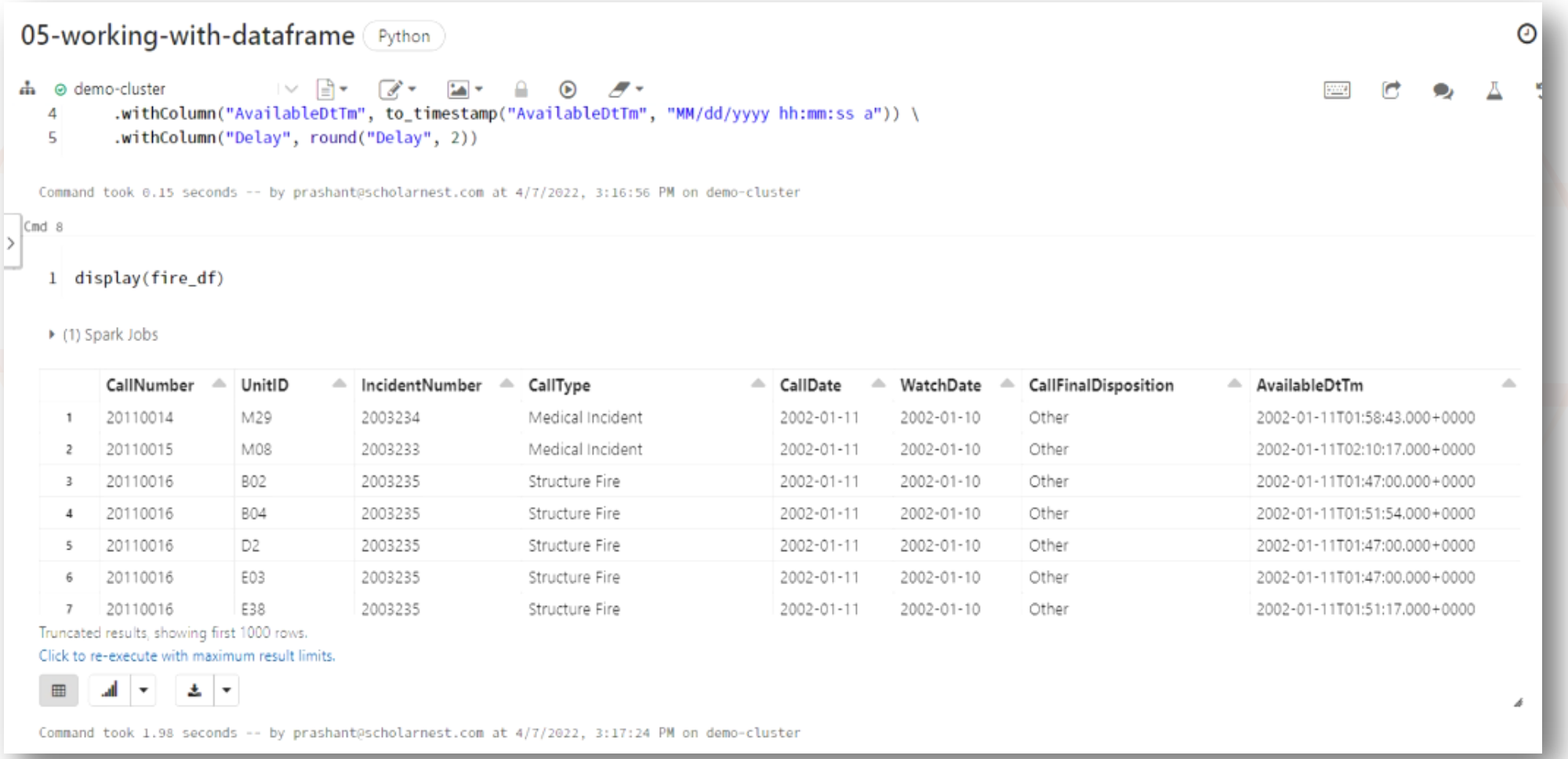
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Working with Spark Dataframe – Part I

Go to your Databricks workspace and open the notebook we created in the previous lecture. **(Reference: 05-working-with-dataframe.ipynb)**

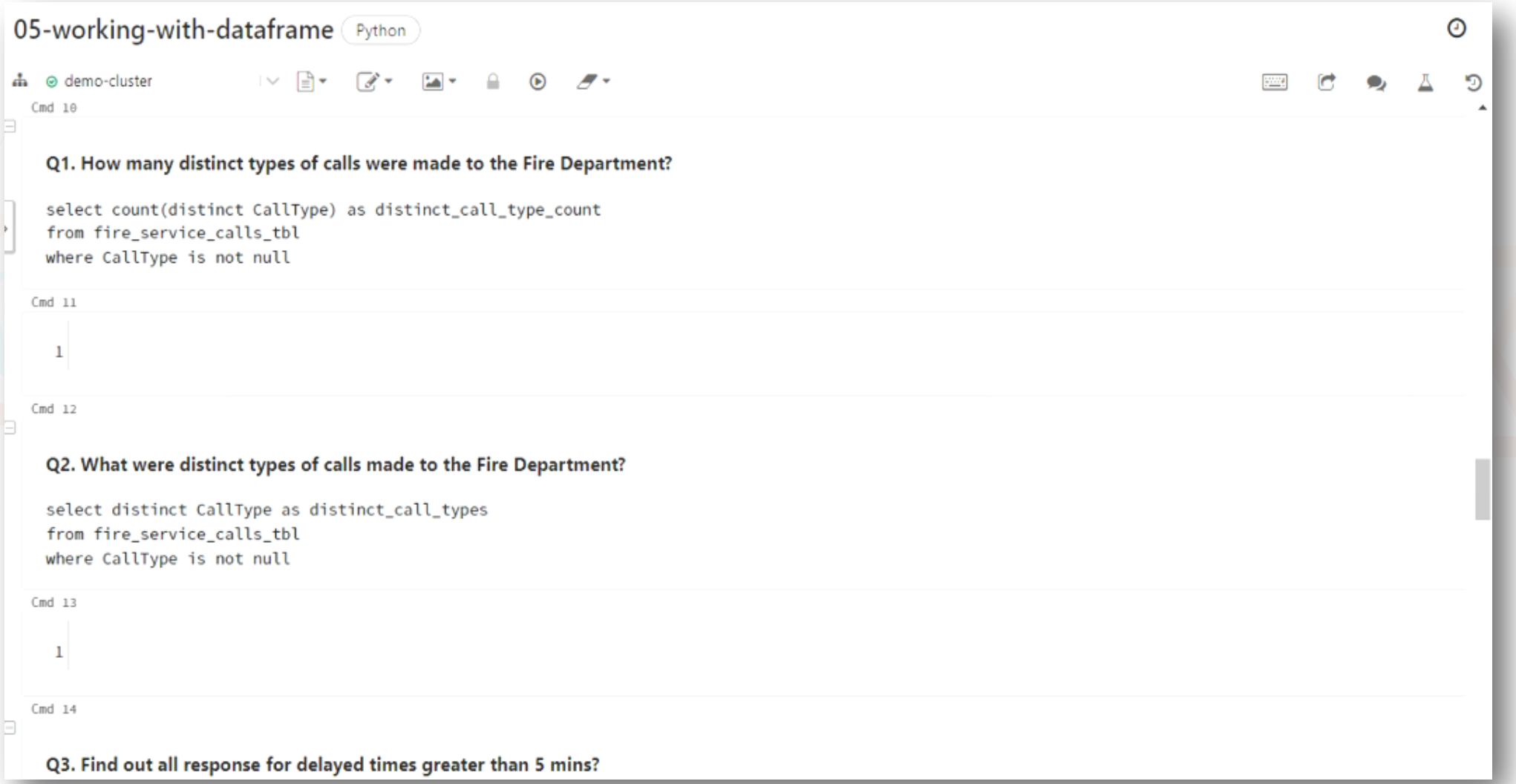


We have transformed the created a Dataframe fire_df which is ready for analysis. You can see the fire_df Dataframe here in the screenshot below.



	CallNumber	UnitID	IncidentNumber	CallType	CallDate	WatchDate	CallFinalDisposition	AvailableDtTm
1	20110014	M29	2003234	Medical Incident	2002-01-11	2002-01-10	Other	2002-01-11T01:58:43.000+0000
2	20110015	M08	2003233	Medical Incident	2002-01-11	2002-01-10	Other	2002-01-11T02:10:17.000+0000
3	20110016	B02	2003235	Structure Fire	2002-01-11	2002-01-10	Other	2002-01-11T01:47:00.000+0000
4	20110016	B04	2003235	Structure Fire	2002-01-11	2002-01-10	Other	2002-01-11T01:51:54.000+0000
5	20110016	D2	2003235	Structure Fire	2002-01-11	2002-01-10	Other	2002-01-11T01:47:00.000+0000
6	20110016	E03	2003235	Structure Fire	2002-01-11	2002-01-10	Other	2002-01-11T01:47:00.000+0000
7	20110016	E38	2003235	Structure Fire	2002-01-11	2002-01-10	Other	2002-01-11T01:51:17.000+0000

We can document all the 10 micro project question in this notebook along with it's solution SQL expression which we have already seen in the earlier lectures.



Before I start analysis, let me cache the Dataframe. You can use the cache() utility method to cache the Dataframe into memory. Because I want to run ten analysis transformations on the same Dataframe.

We already know the logic behind the 10 question as we implemented the same using SQL. But SQL runs on a table. And we want to learn how to run those same 10 logic on a Dataframe.

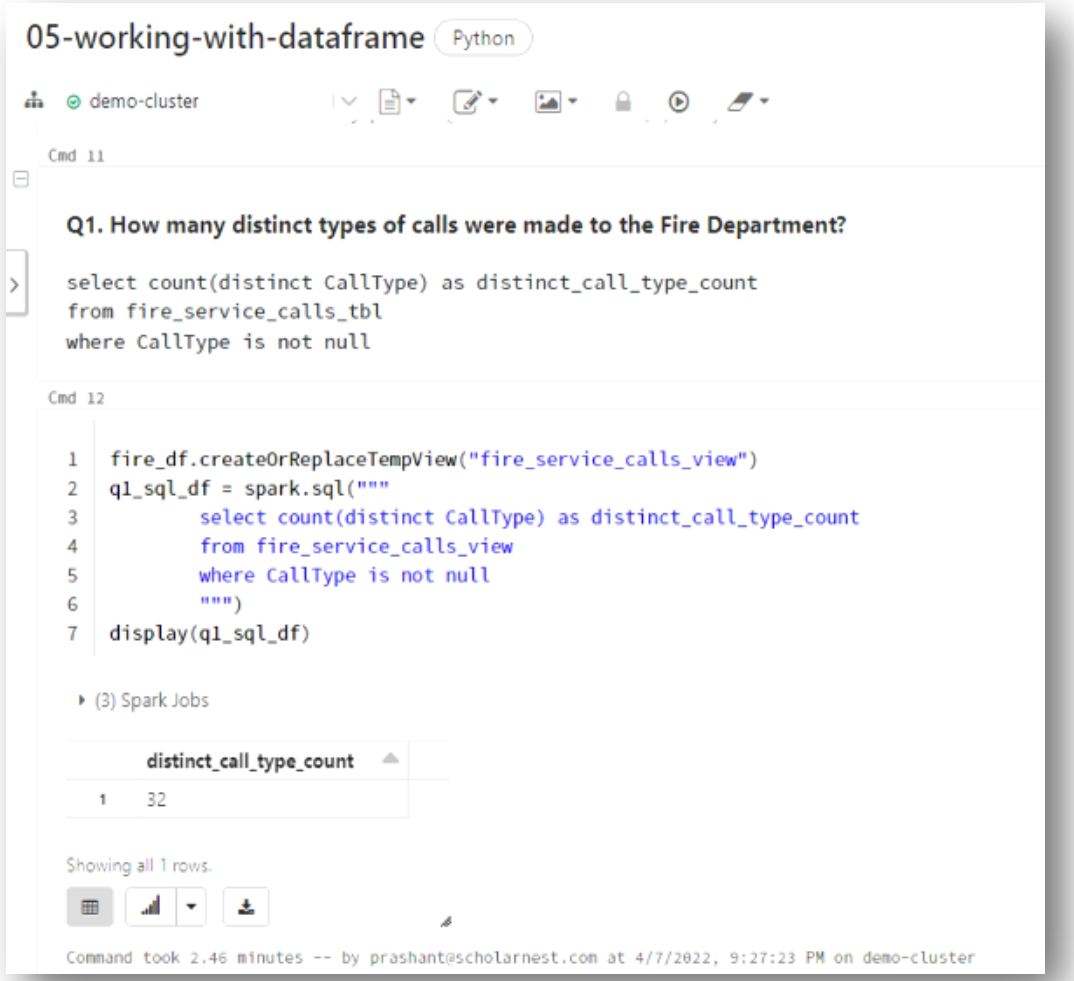
We have two approaches:

1. SQL Approach
2. Dataframe Transformation Approach

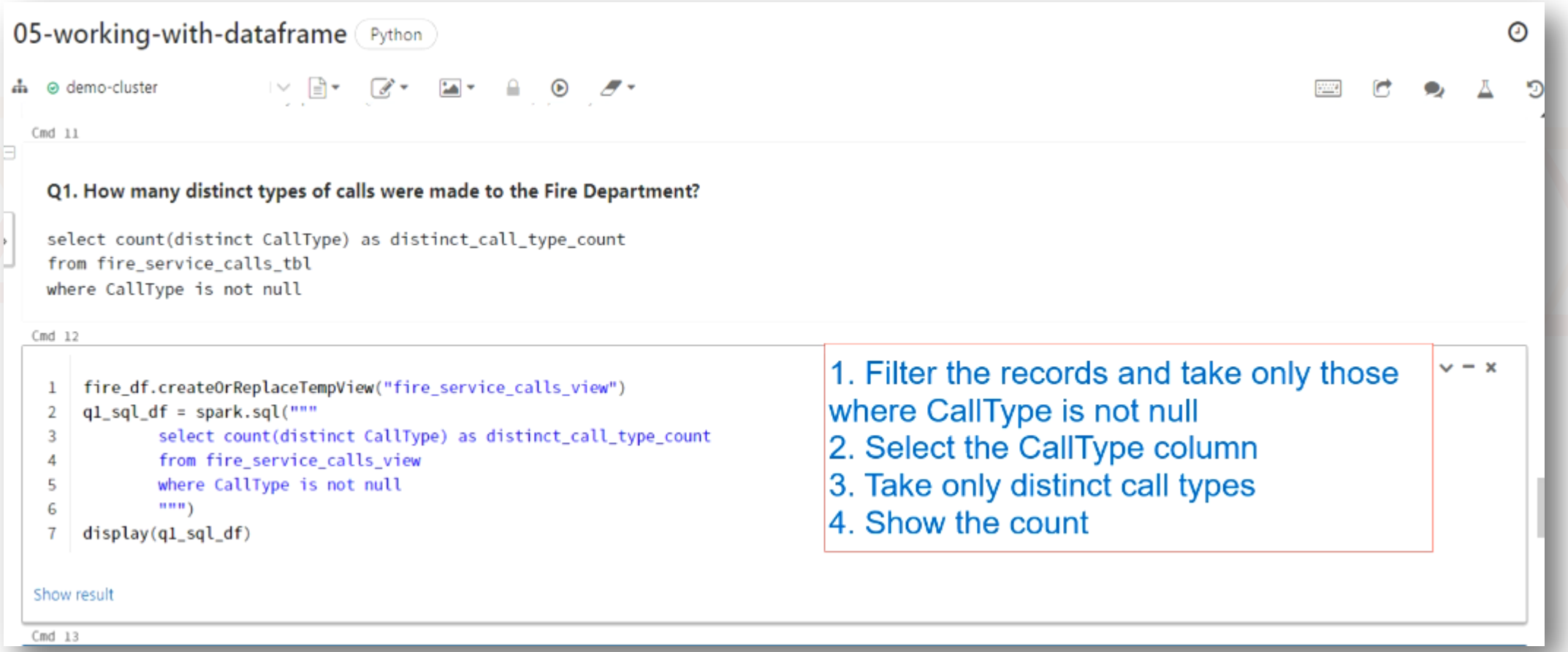
Further, the SQL approach is a two-step process:

1. Convert your Dataframe to a temporary view
2. Run your SQL on the view

Let me proceed with the SQL approach, the first thing is to convert my Dataframe to a temporary view. So I am using the same SQL that I used on the table. But I changed the table name to the view name. Now, the second step is to run SQL queries on the view. The second step will give you a Dataframe.



Now let us see the Dataframe Transformation approach. Here we have the logic for this approach given below. We can build the logic easily just by looking at SQL solutions.



Here is the code for the Dataframe Transformation approach. We got the same desired output with this approach as well.

We can also write the code for the Dataframe approach in the manner shown below. Both the approach will give you the same answer, but it is recommended to follow the first approach shown in the image.

We can also write the code for the Dataframe approach in the manner shown below. Both the approach will give you the same answer, but it is recommended to follow the first approach shown in the image.

Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com

ScholarNest

Spark Azure Databricks

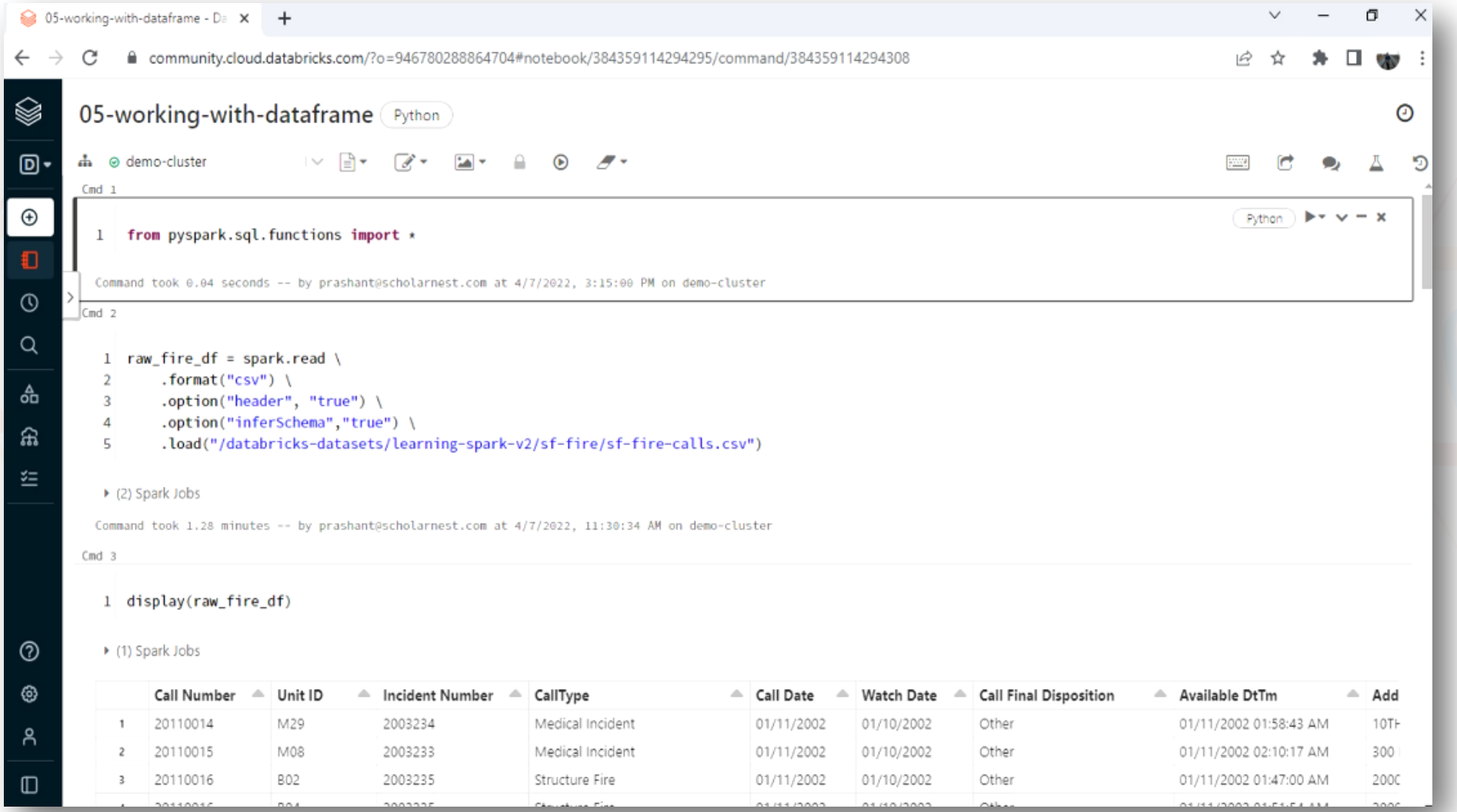
Databricks Spark Certification and beyond

Instructor: Prashant Kumar Pandey

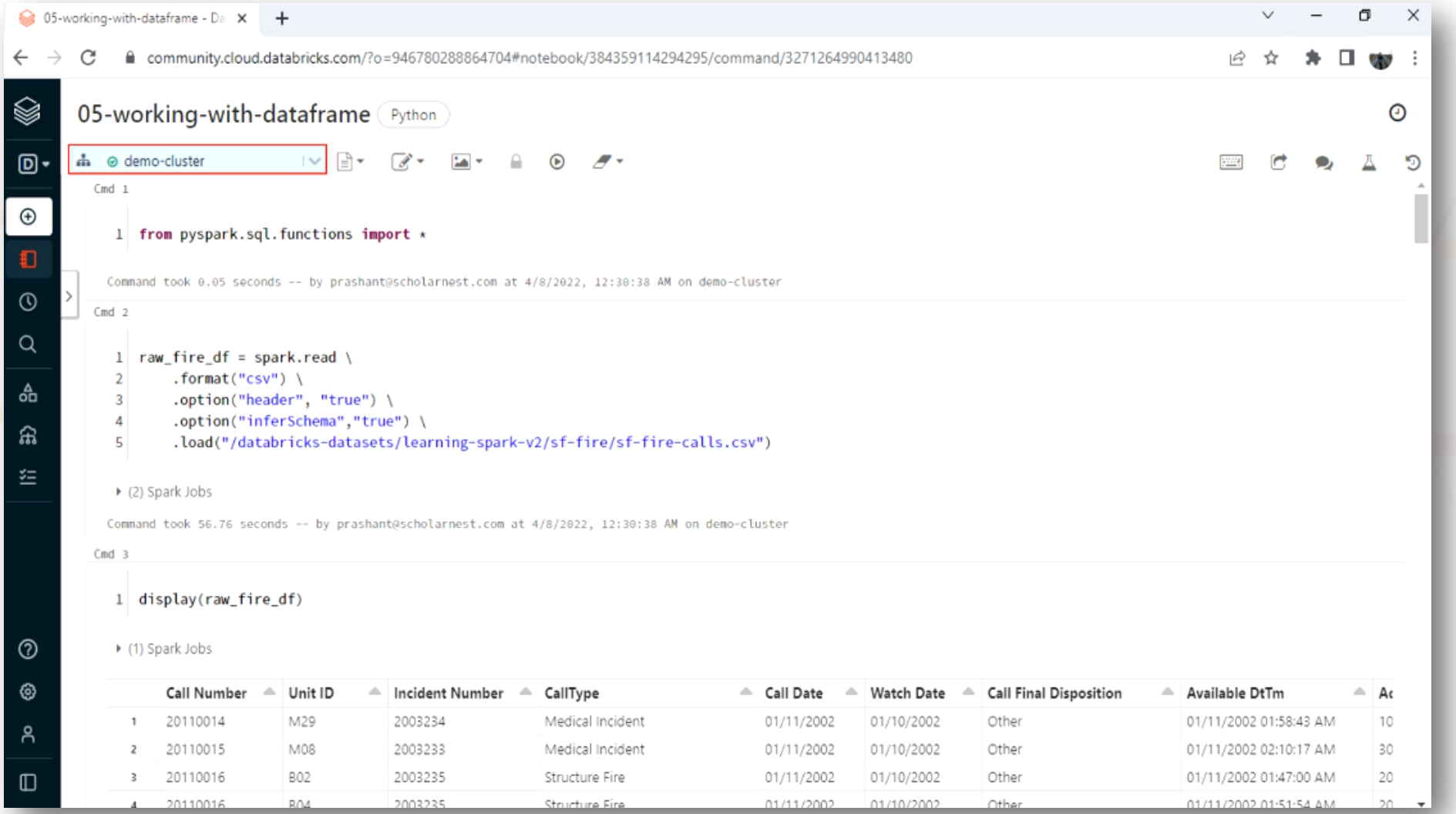
Absolute Beginner to Specialization in Apache Spark and Azure Databricks

Working with Spark Dataframe – Part II

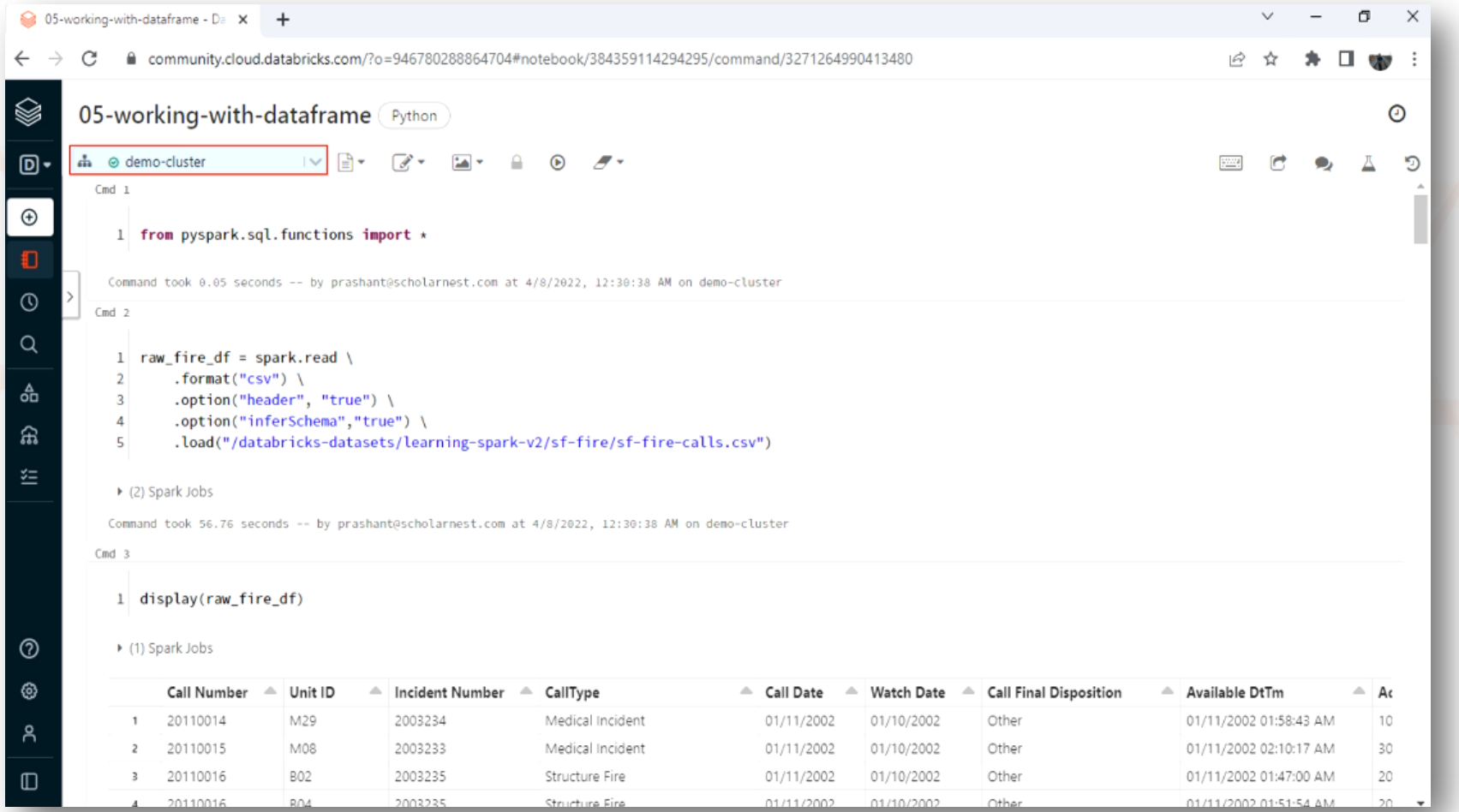
Go to your Databricks workspace and open the notebook we worked with in the previous lecture. **(Reference: 05-working-with-dataframe.ipynb)**



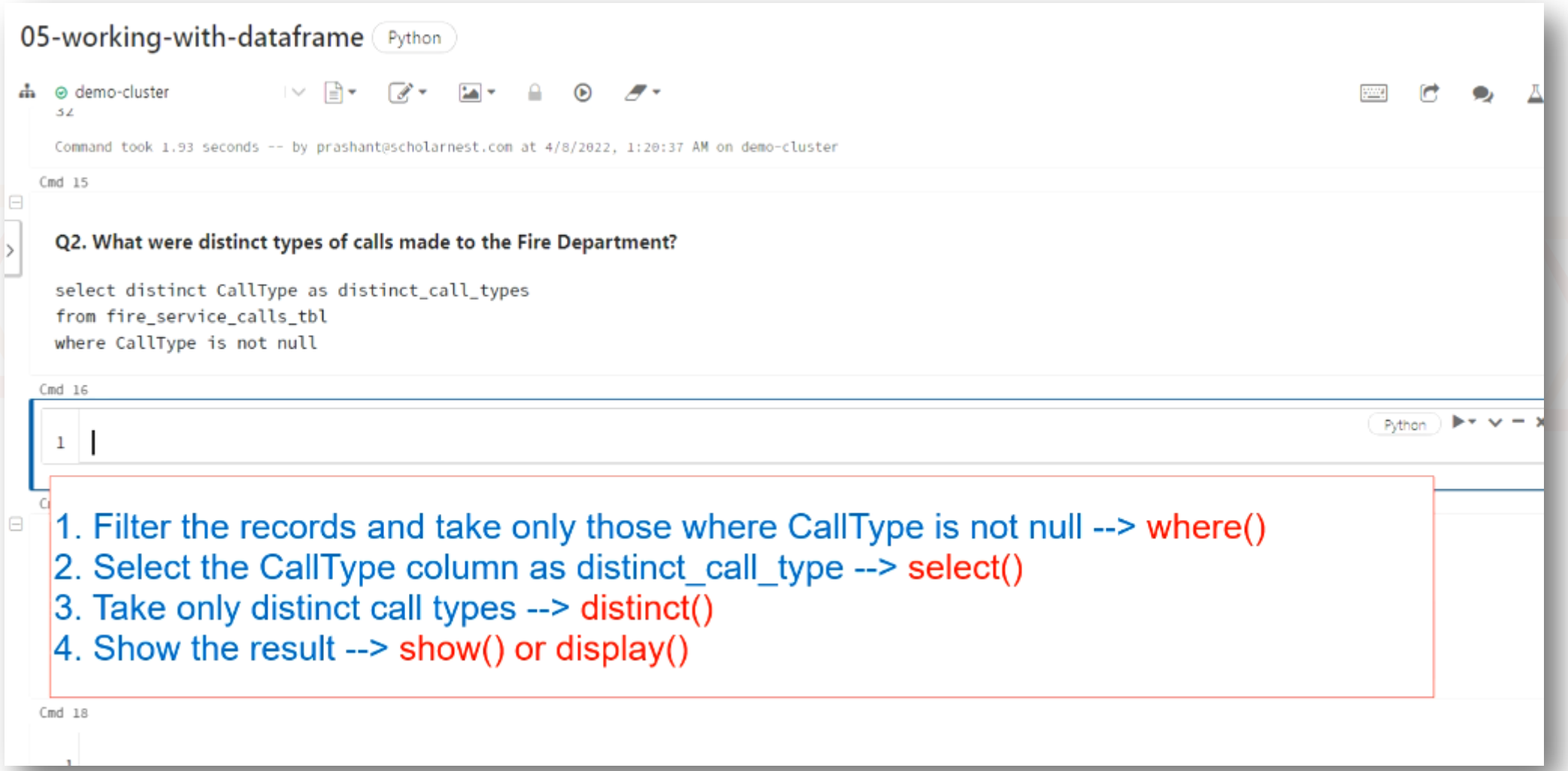
Make sure you have attached a running cluster to your notebook, and run the entire notebook once before you start working on this notebook further.



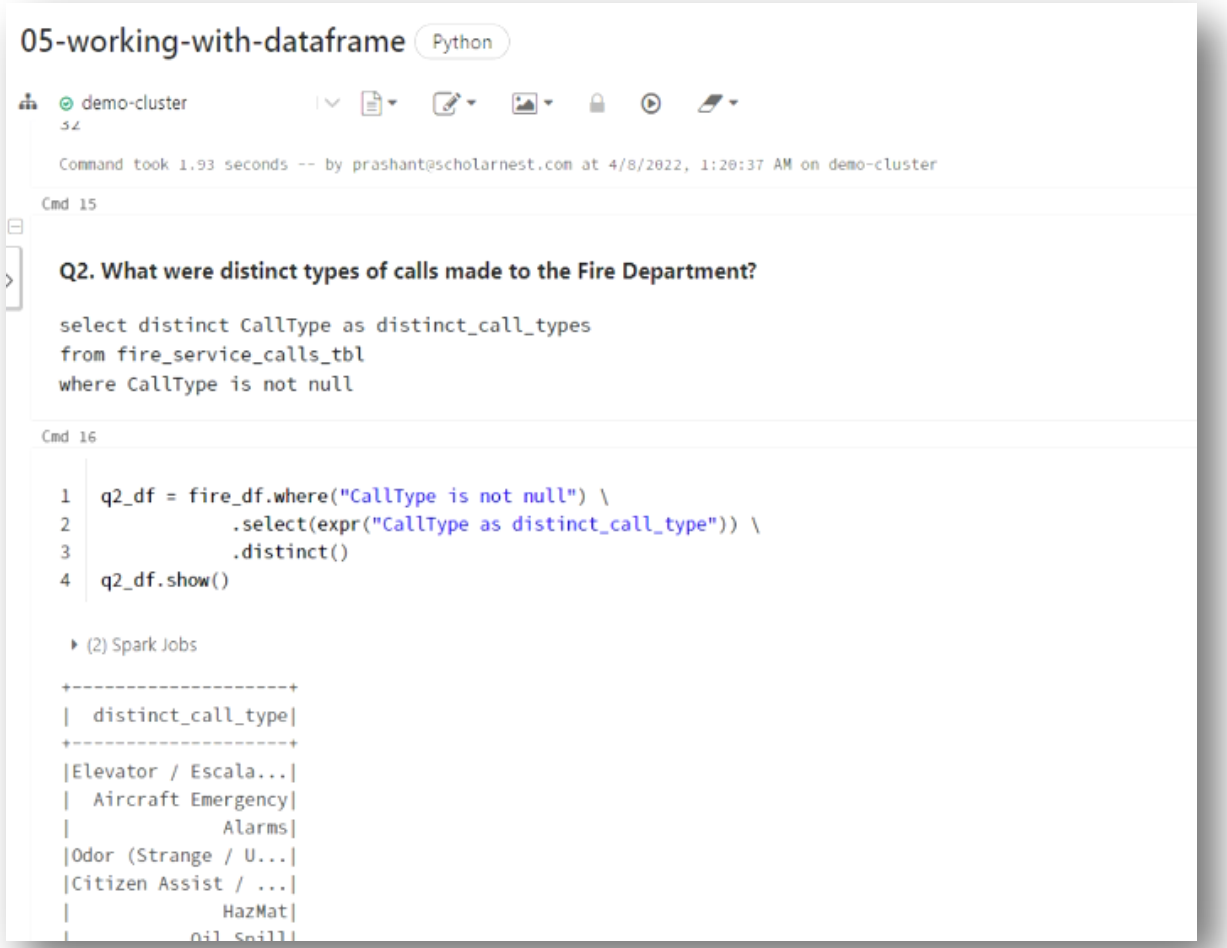
Make sure you have attached a running cluster to your notebook, and run the entire notebook once before you start working on this notebook further. Whenever you stop working on the Databricks cloud and want to return, you must start a new cluster, attach it to your notebook and recreate all the objects you want to use.



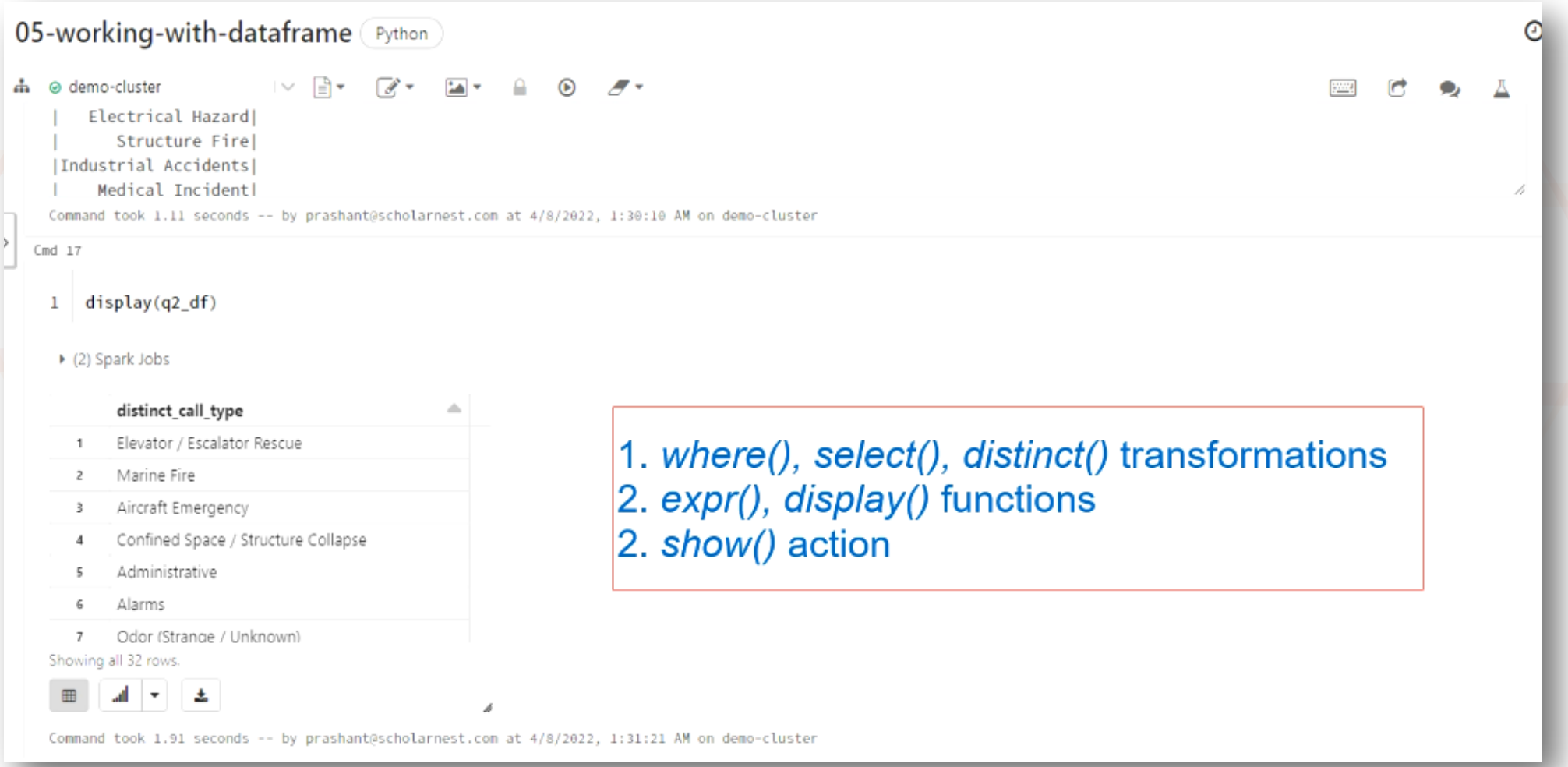
We already solved the first question in the previous lecture. Now, let us look at the second question. We have the logic framed for this question in the image below.



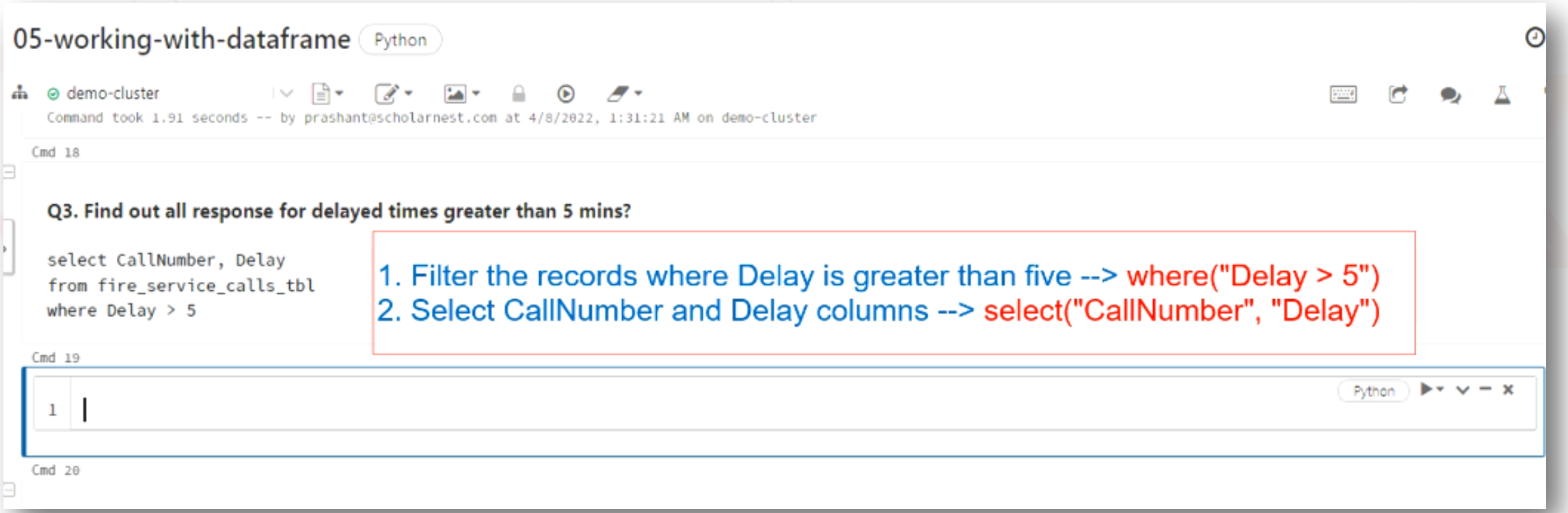
Here is the code for the logic we framed in the second question.
I am using fire_df and applying where transformation. The filter condition takes only those records where CallType is not null. Then I am chaining the select transformation which takes a column name. However, I wanted to give an expression. So, I must use the expr() function to evaluate my expression. Then I am chaining the distinct() transformation. And last step is to take action and show the results.



You can also use to the display() function depicts the formatted results. The display() function is not part of Apache Spark. It is a utility function added by Databricks. And the show() method is a commonly used Spark action which shows a plain text output.
So, we learned the following Dataframe methods from the second question.



Now let us look at the 3rd question.
The question asks for all responses for delayed times greater than 5 mins. We have framed the logic for the same in the image shown below.

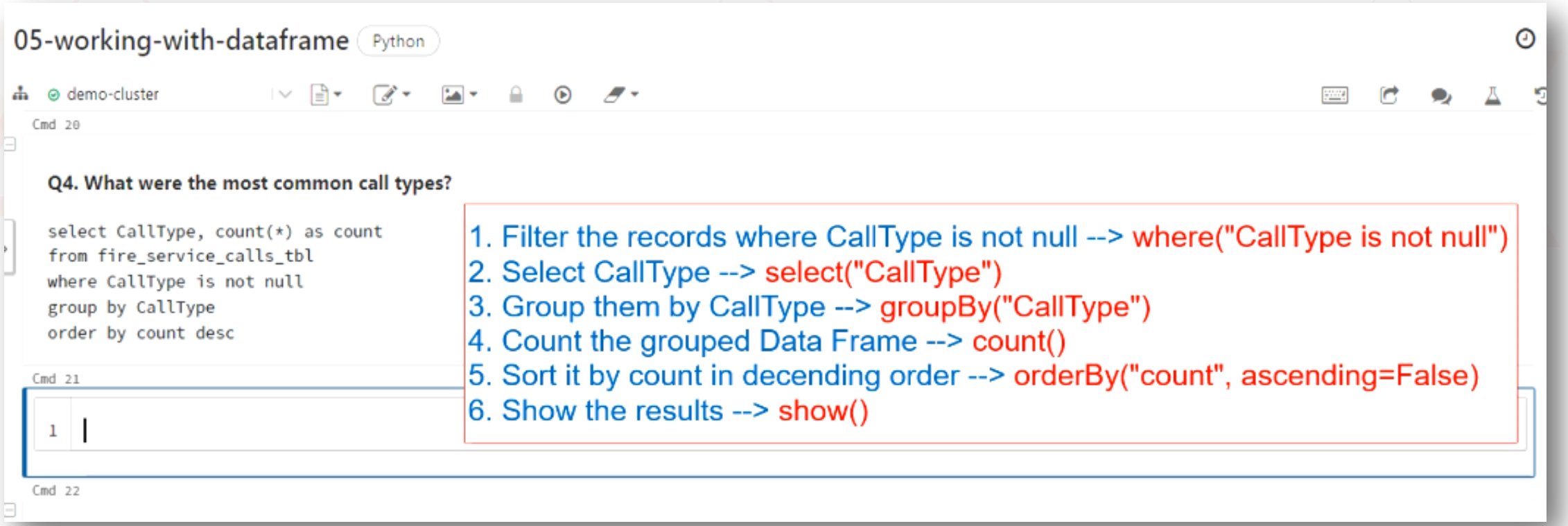


We have implemented the logic into code and we can see the desired output below.

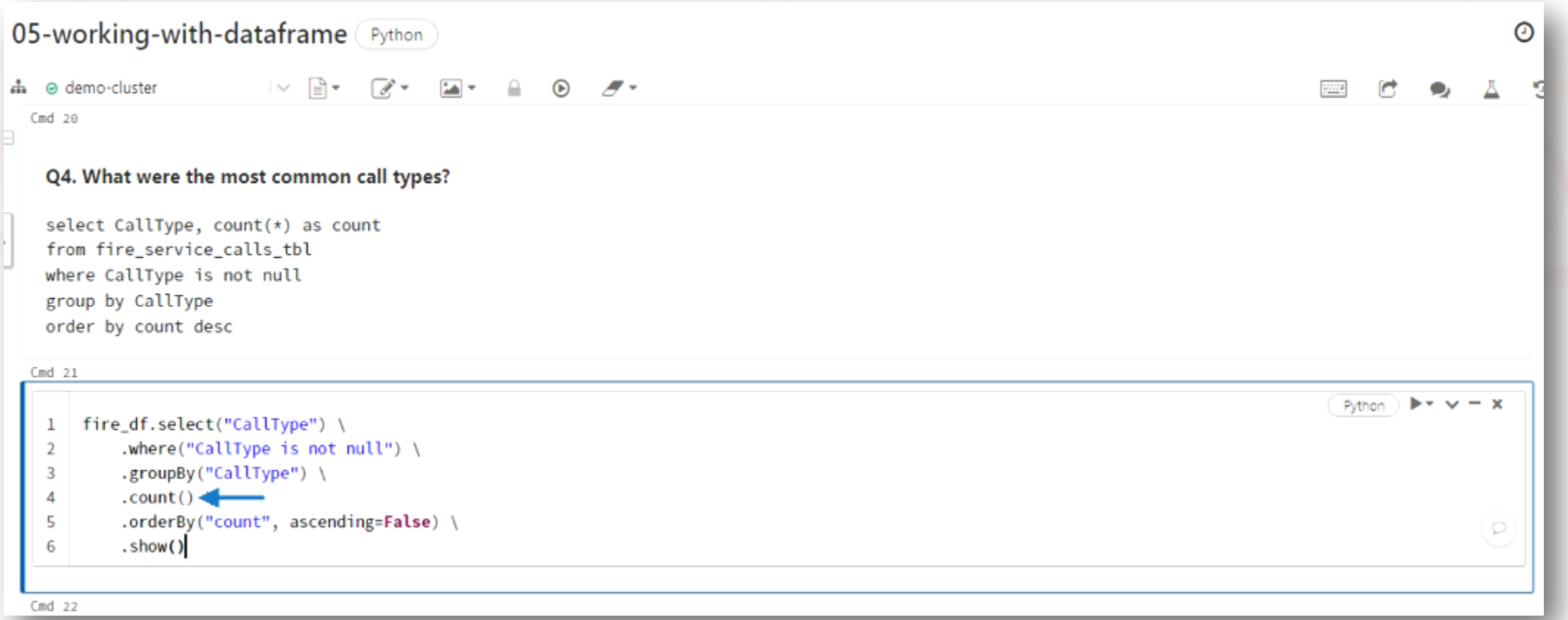
We are doing two new things in this code:

1. We are selecting two columns in the select() method. The select() method takes a comma-separated list of columns.
2. We chained the show() action. Remember that you can chain one and only one Action, and that must be the last method in the chain. So I cannot chain anything else after the show() method.

Here we have the next question and it's logic along side.
We want to filter records where the call type is not null and select the CallType. But we want to find the most common CallType. So we must count the calls for each CallType. Hence, we will group by the CallType and then count for each group. The last step is to sort the record by the count in descending order to know which ones are the most common.



Here is the code for the fourth question.
But I want to highlight one thing here, I am using count() transformation in this example. But we already learned that the count() is an action.



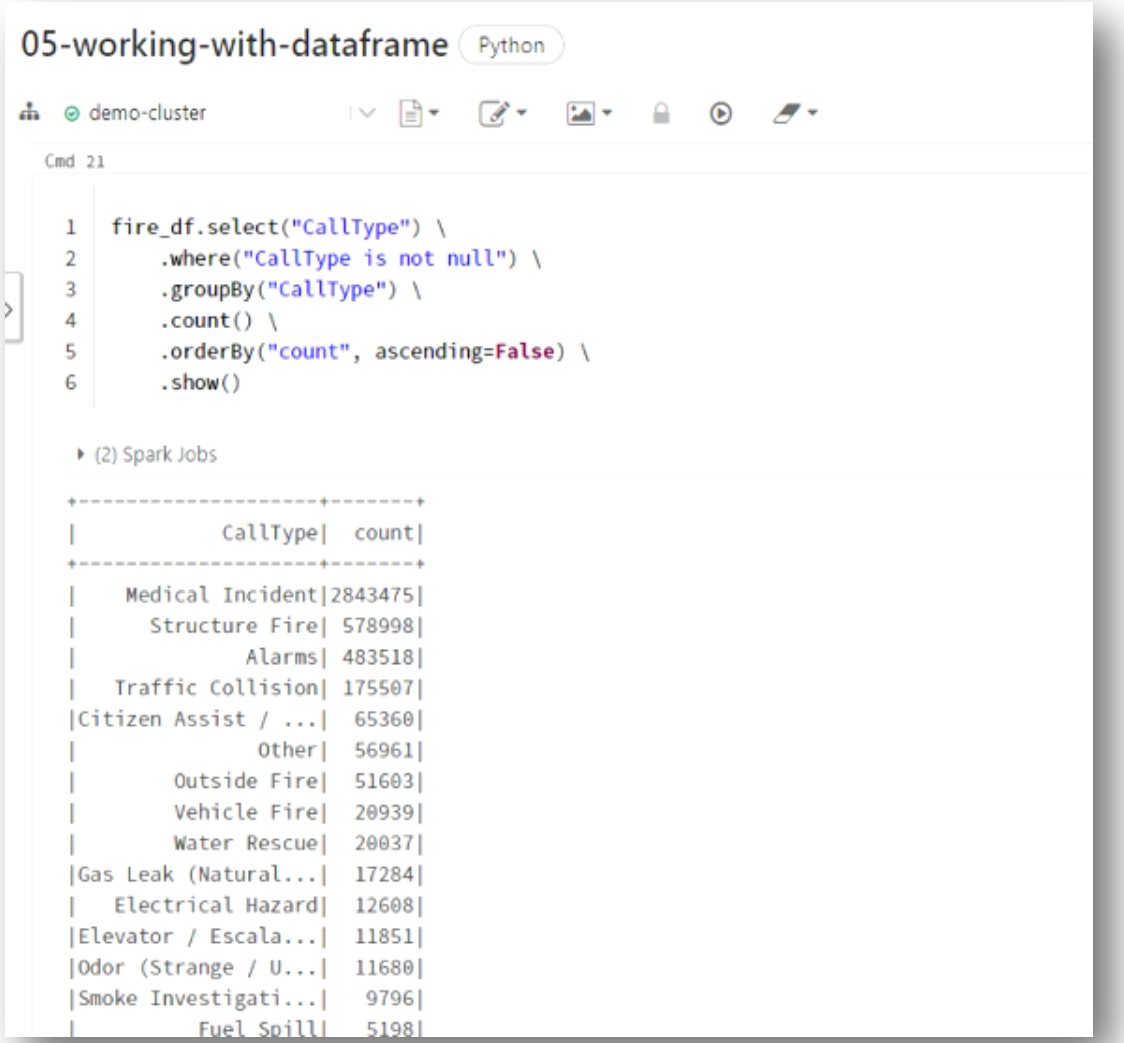
So we have two count() methods in Spark:

1. Dataframe.count()
2. GroupedData.count()

If you are using a count before the groupBy(), it will be an action, and you cannot chain any other transformation after the count () Action.

However, if you use a count() method after the groupBy(), it will be a transformation, and you can chain further transformations.

We can see the desired output for the fourth question as well.
The remaining 6 questions are given for practice, you can compare your answer with the solutions in the material provided.



Thank You

ScholarNest Technologies Pvt Ltd.

www.scholarnest.com