ConverSight

Data Loading through Notebook

Deep Dive into Data Loading Techniques and Analysis.

Datasheet

Table of Contents

1. Introduction 02

2. Creation of Dataset within ConverSight 02

3. Using Jupyter Notebook for Data Loading 08

4. Analyzing Data in Jupyter Notebook 13

5. Smart Analytics Integration 17

- 5.1 Create Method 17
- 5.2 Update Method 19
- 5.3 Publish Method 20
- 5.4 Delete Method 22

6. Conclusion 23

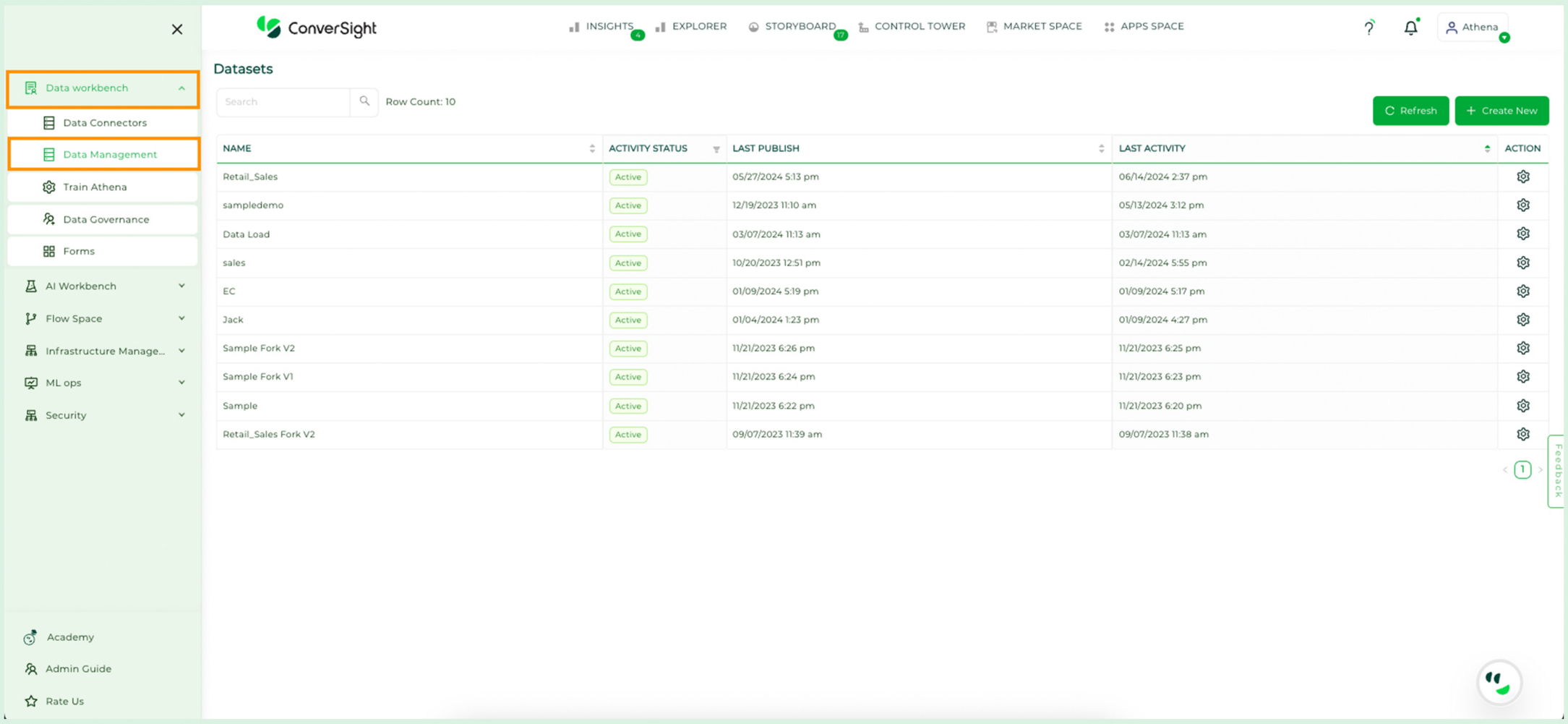
1. Introduction

ConverSight provides a seamless experience for loading and managing data directly within the platform. Additionally, you can also enhance your data management capabilities by leveraging Jupyter Notebook. This approach aligns with modern data-driven practices, offering solutions for complex data analytics and machine learning endeavors. By integrating Jupyter Notebook with ConverSight's Smart Analytics, you access an interactive, flexible and powerful toolkit for data loading and analysis. Dive into this datasheet to explore the enriched capabilities offered through Jupyter Notebook for loading the data.

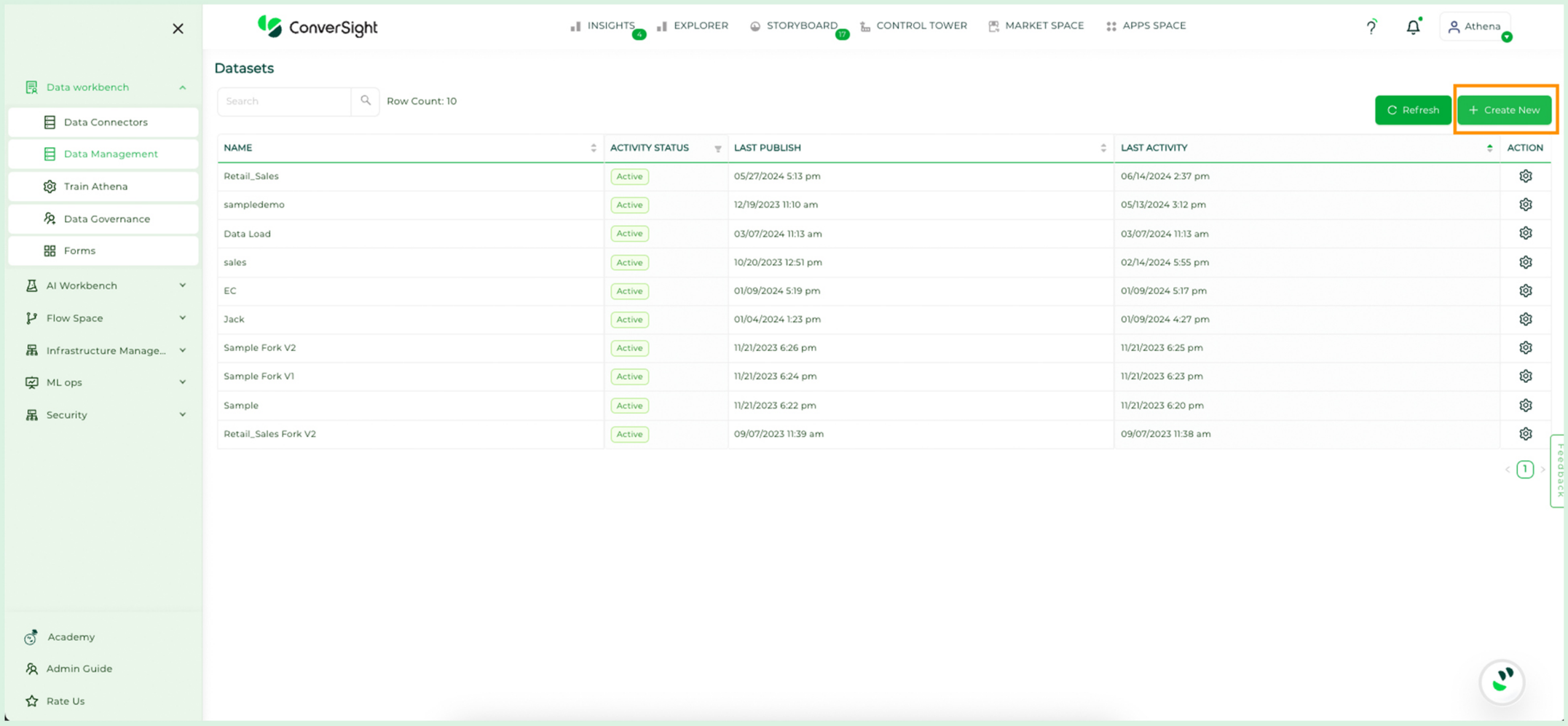
2. Creation of Dataset within ConverSight

To initiate the data upload process in Juypter Notebook, you must first create a Dataset, within the ConverSight platform. Subsequently, the creation of a new dataset is performed by the following steps:

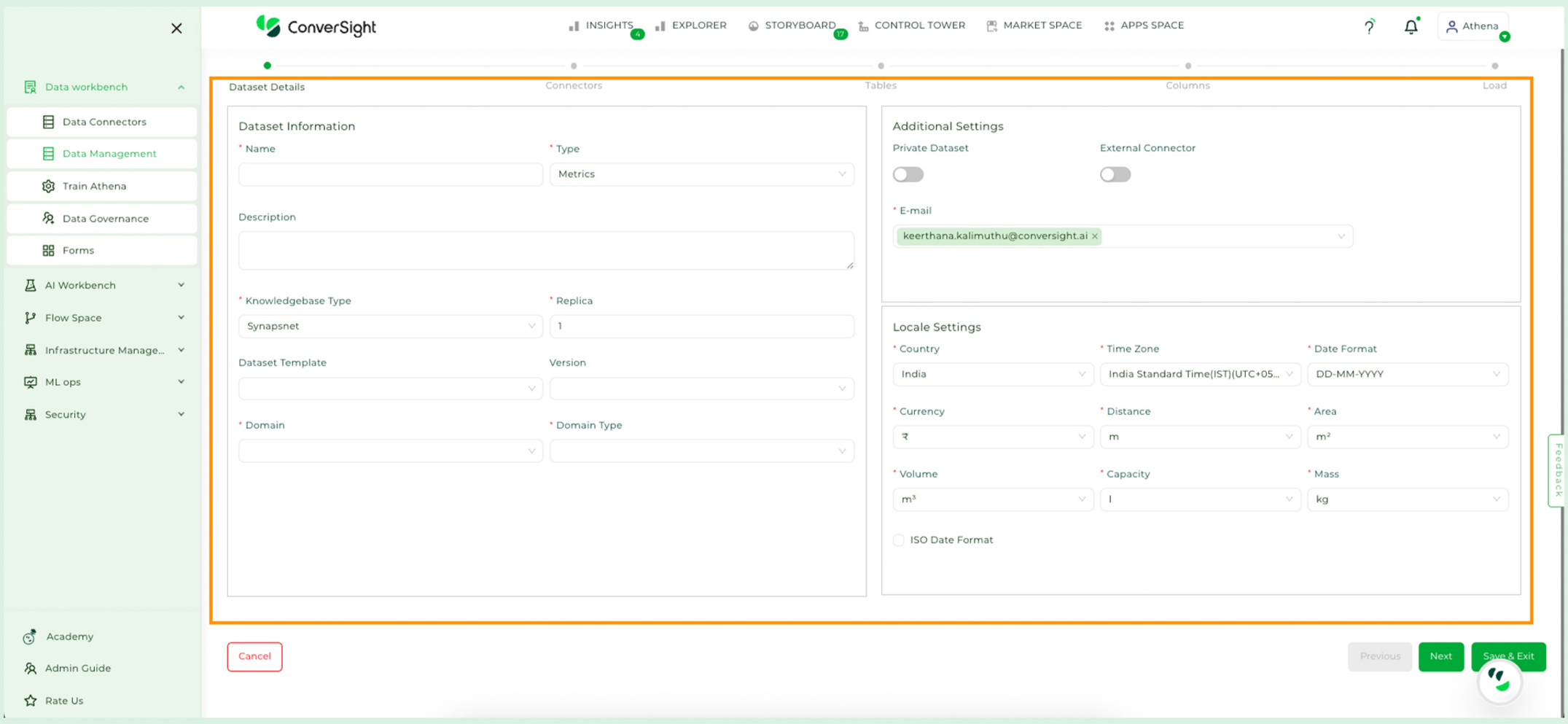
Step 1: Navigate to the **'Data Management'** section under the **'Data Workbench'** menu.



Step 2: Select the **'Create New'** option to initiate the dataset processing in ConverSight.



Step 3: Fill in the Dataset Details, providing essential information about the dataset, locale settings and additional configurations necessary for comprehensive dataset configuration.



Dataset Information

Attribute	Description
Name	Provide a name for your dataset.
Type	Define the dataset type based on its characteristics, such as metrics or documents, for effective organization and utilization in later stages.
Description	Provide a description for enhanced comprehension, as per your preference.
Knowledge Base Type	Choose between SynapsNet and KBNet for dataset engine, favoring SynapsNet for improved natural language understanding.
Replica	Determine the number of replicas to maintain duplicates of data. Recommended to generate at least two replicas, especially for critical datasets. (Note: By default, a single replica is provided.)
Dataset Template	Select a template matching your dataset.
Version	Choose a version corresponding to the selected dataset template for data integrity and compatibility.
Domain	Specify the dataset's primary category or domain, for effective utilization in targeted analyses or applications.

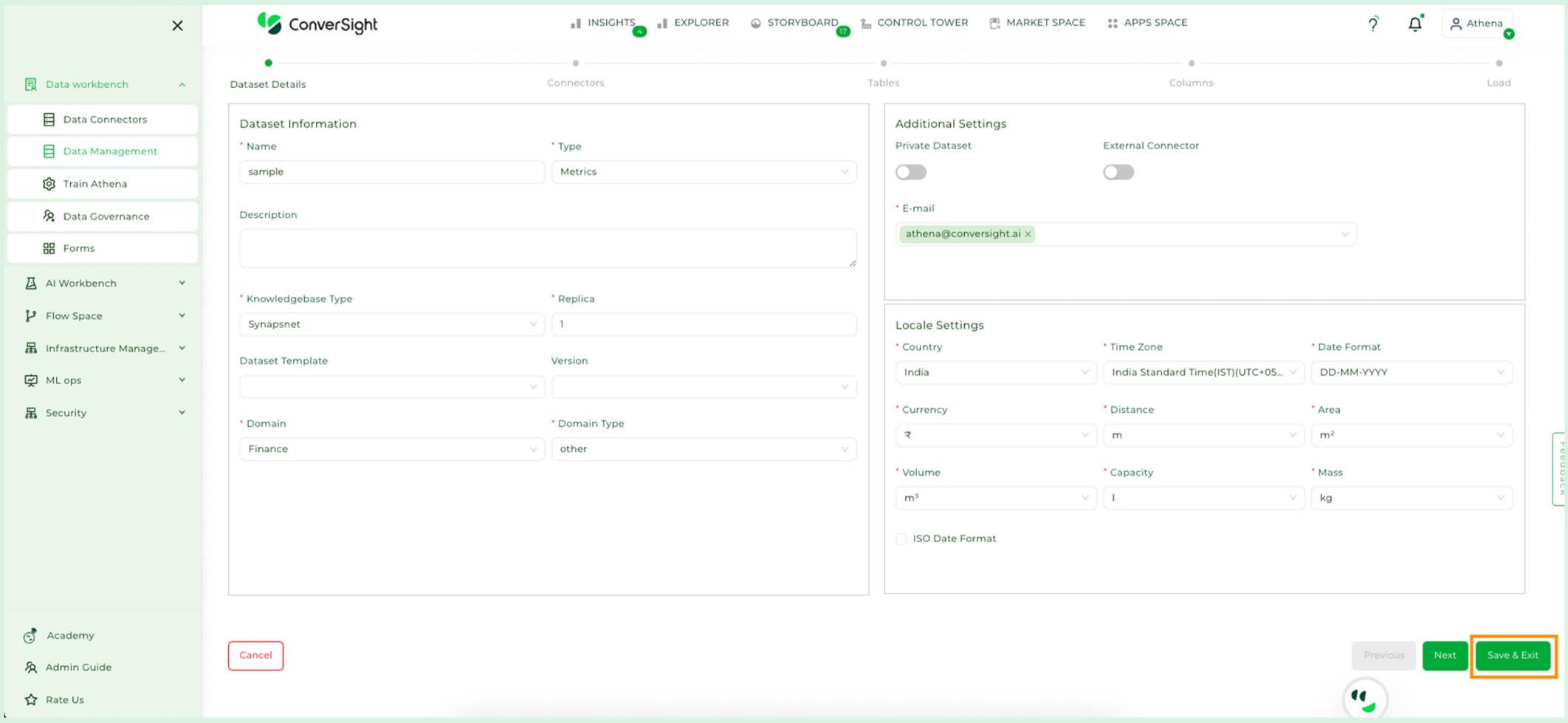
Locale Settings

Attribute	Description
Locale Settings	Customizable configurations based on your specific location. Selection impacts metric options, including currency, volume and distance, adapting to the chosen locale.
ISO Format	Checkbox selected when data loaded in one time zone is used across various locations, preventing time lag discrepancies in data calculations.

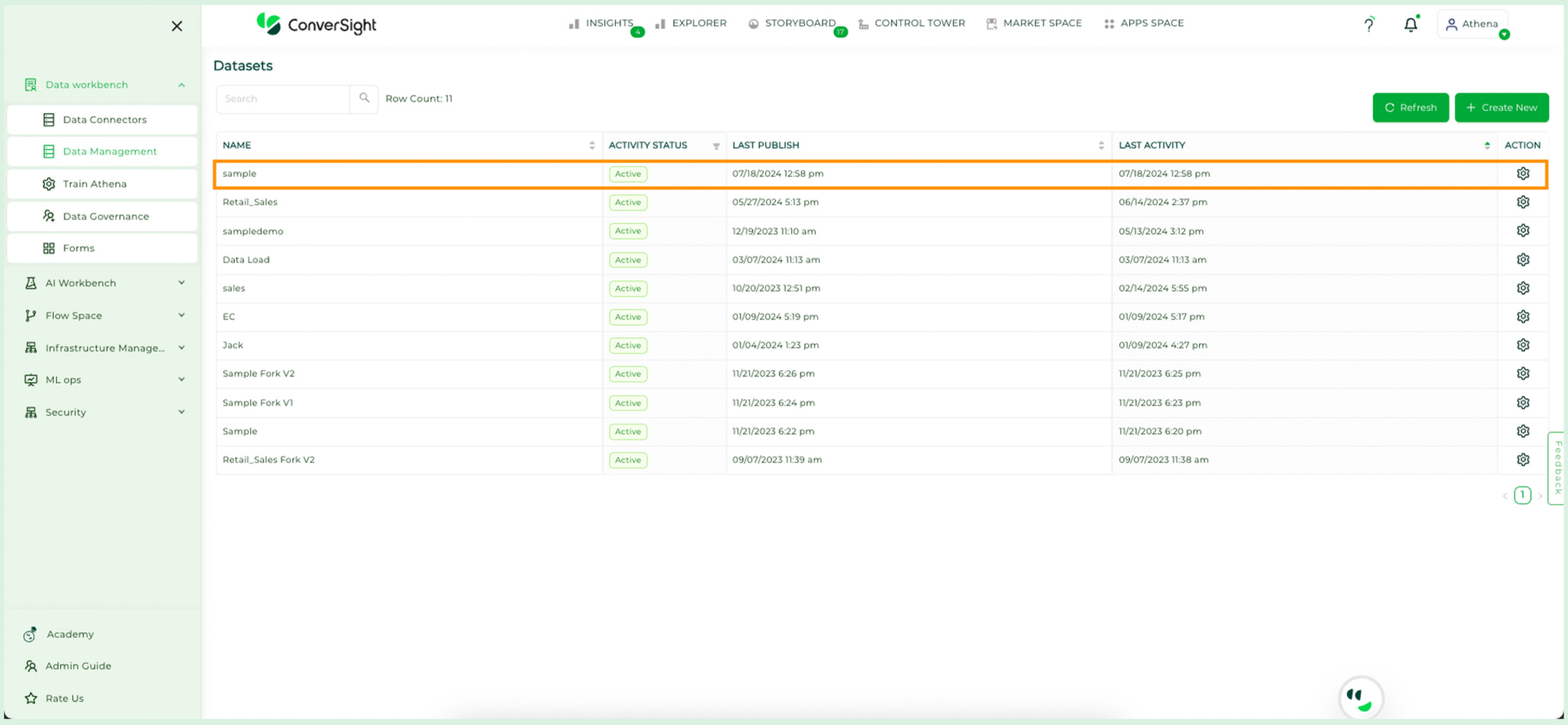
Additional Settings

Attribute	Description
Private Dataset	Enabling ensures the security of your dataset with exclusive access restricted solely to you.
External Connector	Enabling allows direct connection to the dataset, retrieving data from the source database through metadata. Currently accessible only for Snowflake. Note: Users are advised not to select the External Connector option for Data Loading through notebook.
Email	Includes your email address. Additional email addresses can be appended if necessary.

Step 4: Once you have completed entering all the necessary details, proceed by clicking the **'Save & Exit'** button.



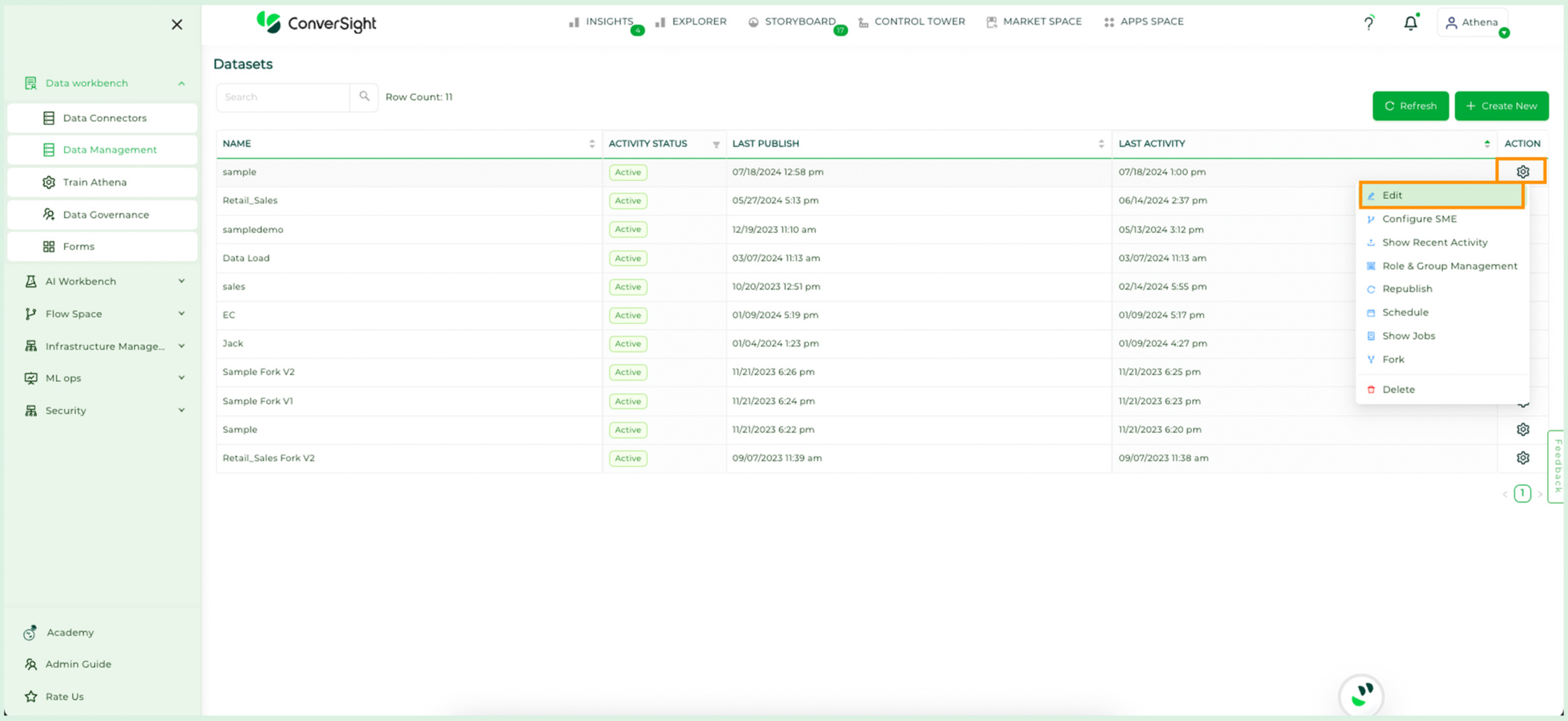
Step 5: Executing these steps results in the creation of a new dataset with an **'Active'** status, initially without pre-loaded data. This status allows you to upload your data through the Notebook in ConverSight.



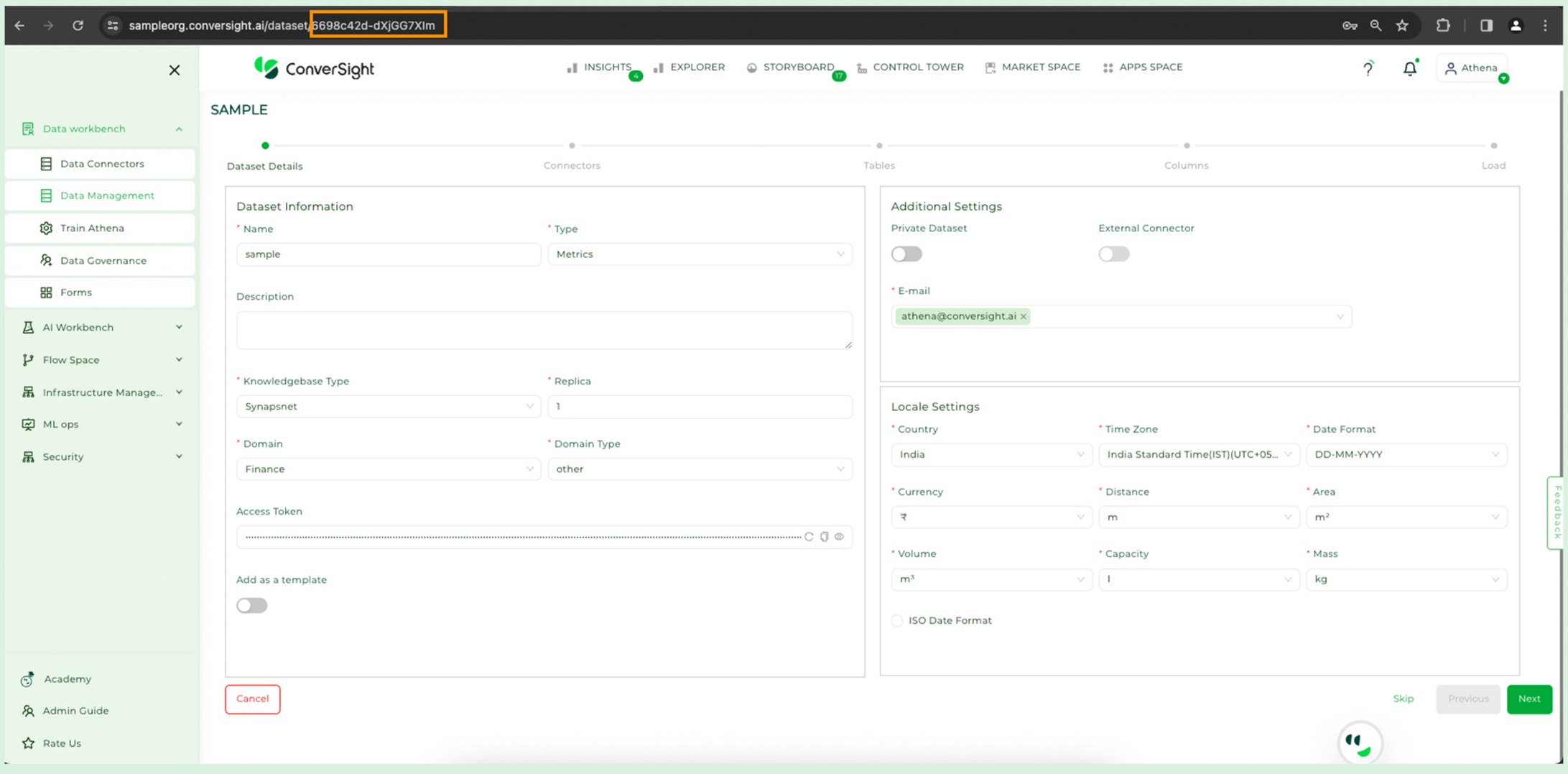
Note

In ConverSight, the option to load data through the notebook is limited to Excel and CSV file formats.

Step 6: To connect your dataset, you'll require the Dataset ID. Click on the **'Settings'** icon located in the Action column and select the **'Edit'** option.



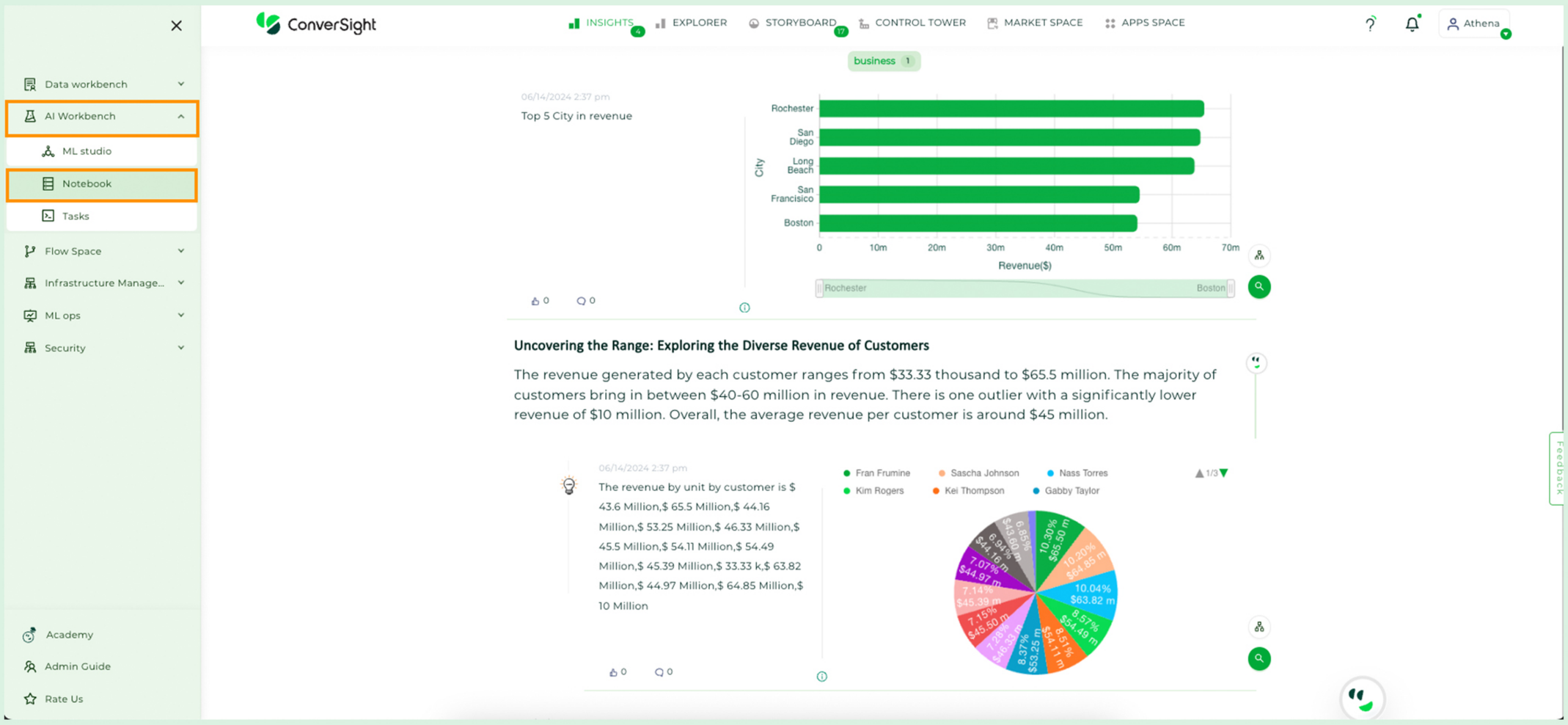
From the Edit Dataset page, copy the last segment of the URL, which represents your dataset ID.



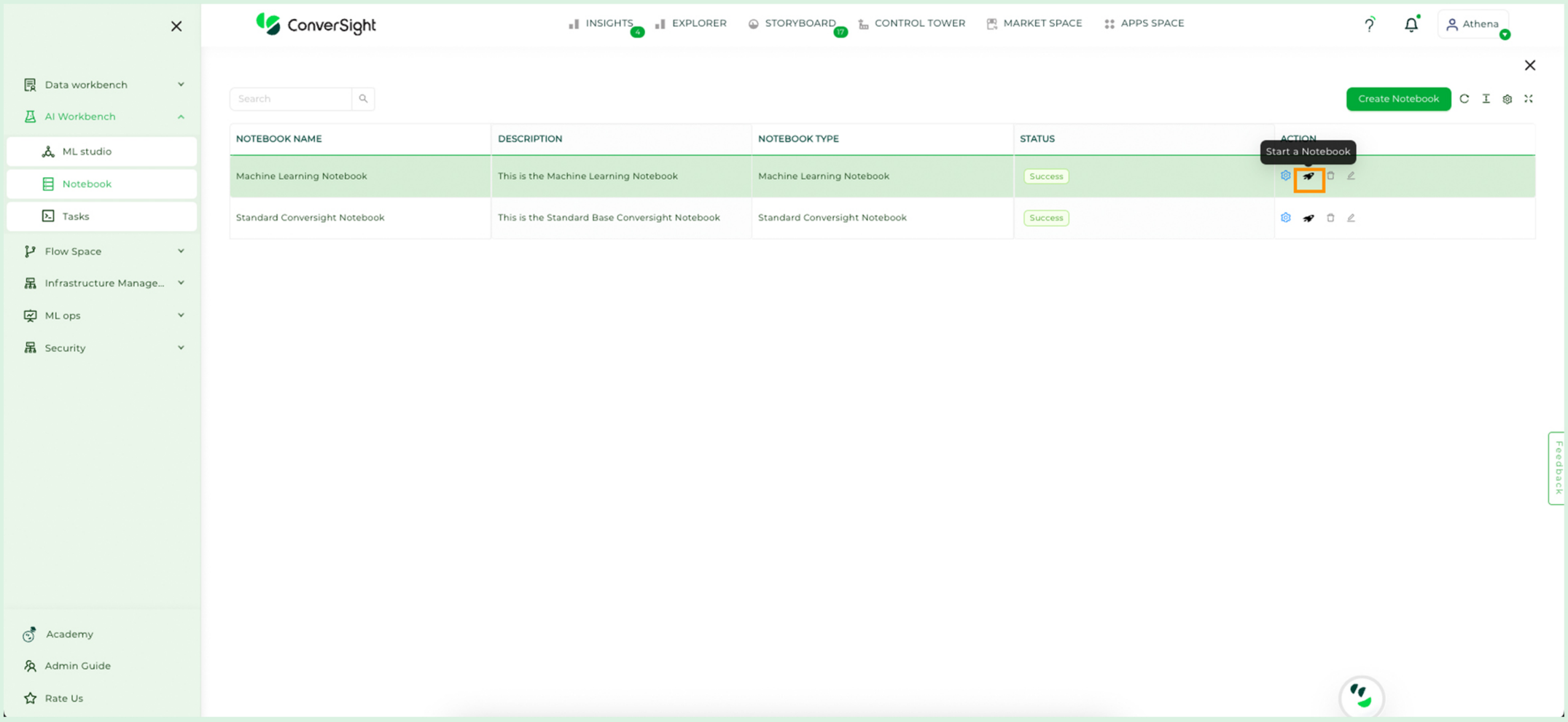
3. Data Loading through Jupyter Notebook

In ConverSight, Jupyter Notebook serves as a cornerstone, in data analytics and machine learning workflows. It acts as the primary interface providing a dynamic online hub for exploring and analyzing data. The interactive features of Jupyter Notebook facilitate real-time code executions and visualization of results, providing a user-friendly environment for experimentation and idea iteration. By guiding to the Jupyter Notebook server, ConverSight offers an immersive environment for data science tasks. The following are the steps to load your data through Jupyter Notebook,

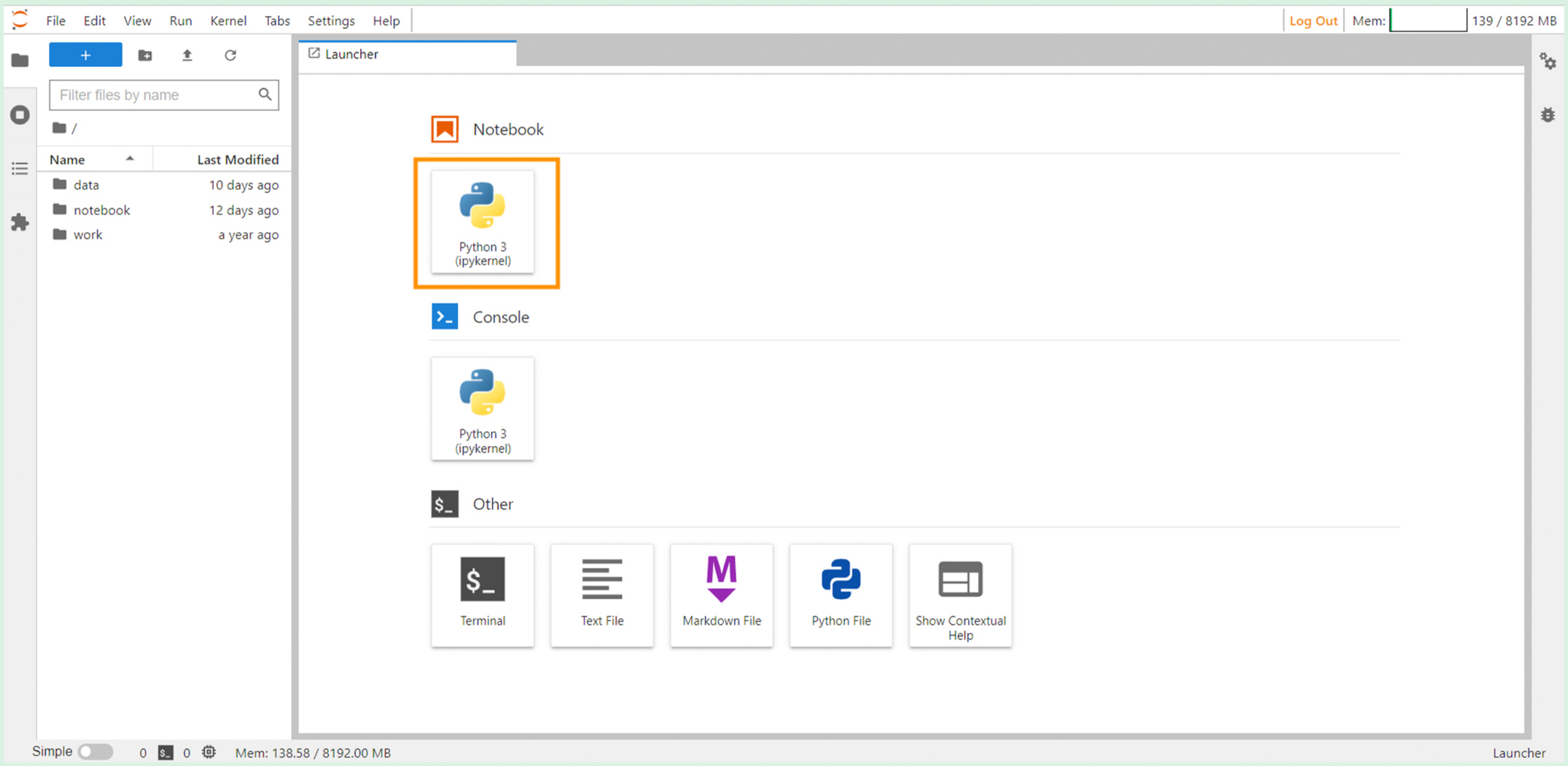
Step 1: Navigate to the **'AI Workbench'** menu in the configuration sidebar and select the **'Notebook'** option. .



Step 2: ConverSight offers two types of Notebooks: The **Machine Learning Notebook,** designed for advanced analytics and machine learning tasks with specialized tools and algorithms and the **Standard ConverSight Notebook**, which serves as a foundational tool with essential packages for general data manipulation and analysis. Choose a **'Notebook'** and initiate by clicking on the **'Rocket'** icon in the Action column.



Step 3: Choose the preferred **'kernel'**.



Step 4: Import the necessary libraries, to access the necessary code encapsulated within the classes provided by the library.

```
import pandas as pd
from conversight import Dataset, SmartAnalytics
```

The table below highlights key code snippets used in ConverSight, demonstrating the import of libraries and specific classes.

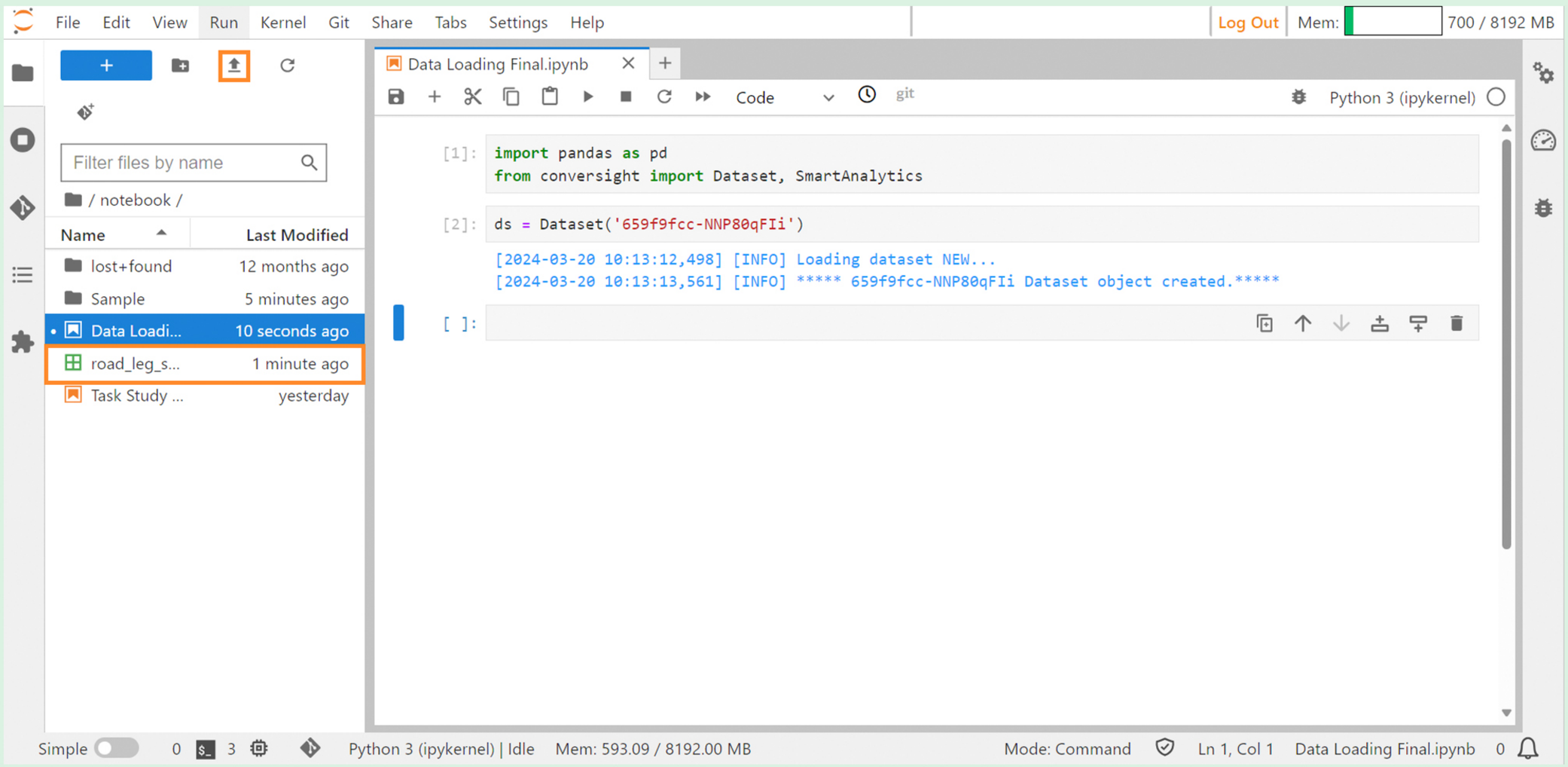
Code	Description
import pandas as pd	Imports the pandas library, allowing for easier reference to its functions and classes.
from conversight import Dataset, SmartAnalytics	Imports specific classes and functions from the conversight library, including Dataset and SmartAnalytics. These are likely related to dataset management.

Step 5: Enter the **'DatasetID'** from the Edit Dataset page for the dataset you intend to load.

```
ds = Dataset('659f9fcc-NNP80qFIi')
```

```
[2024-01-11 11:04:14,615] [INFO] Loading dataset NEW...
[2024-01-11 11:04:15,019] [INFO] ***** 659f9fcc-NNP80qFIi Dataset object created.*****
```

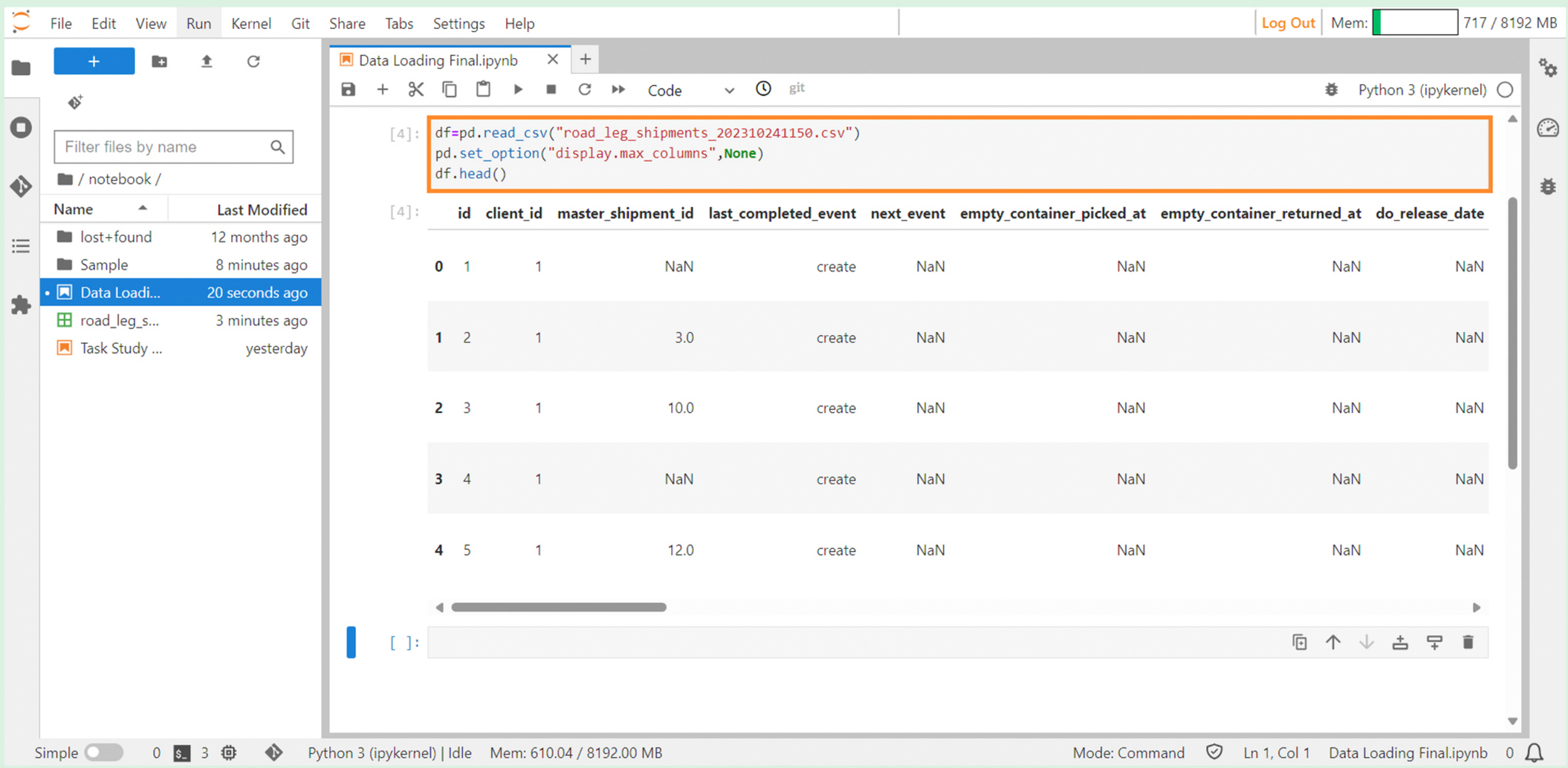
Step 6: Use the **'Upload Files'** icon within ConverSight to bring in your desired file, whether it's in Excel or CSV format.



Step 7: Upon uploading, you can employ the provided commands to read the complete data.

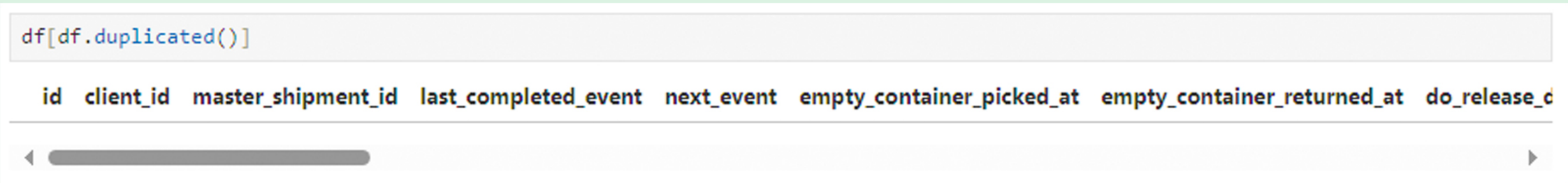
Command	Action
df = pd.read_csv("Filename.csv")	Read complete data from a CSV file
df = pd.read_excel("Filename.xlsx")	Read complete data from an Excel file.
pd.set_option("display.max_columns", None)	Display all columns of the data in the dataframe (df).

df.head()	View entire dataset without column restrictions



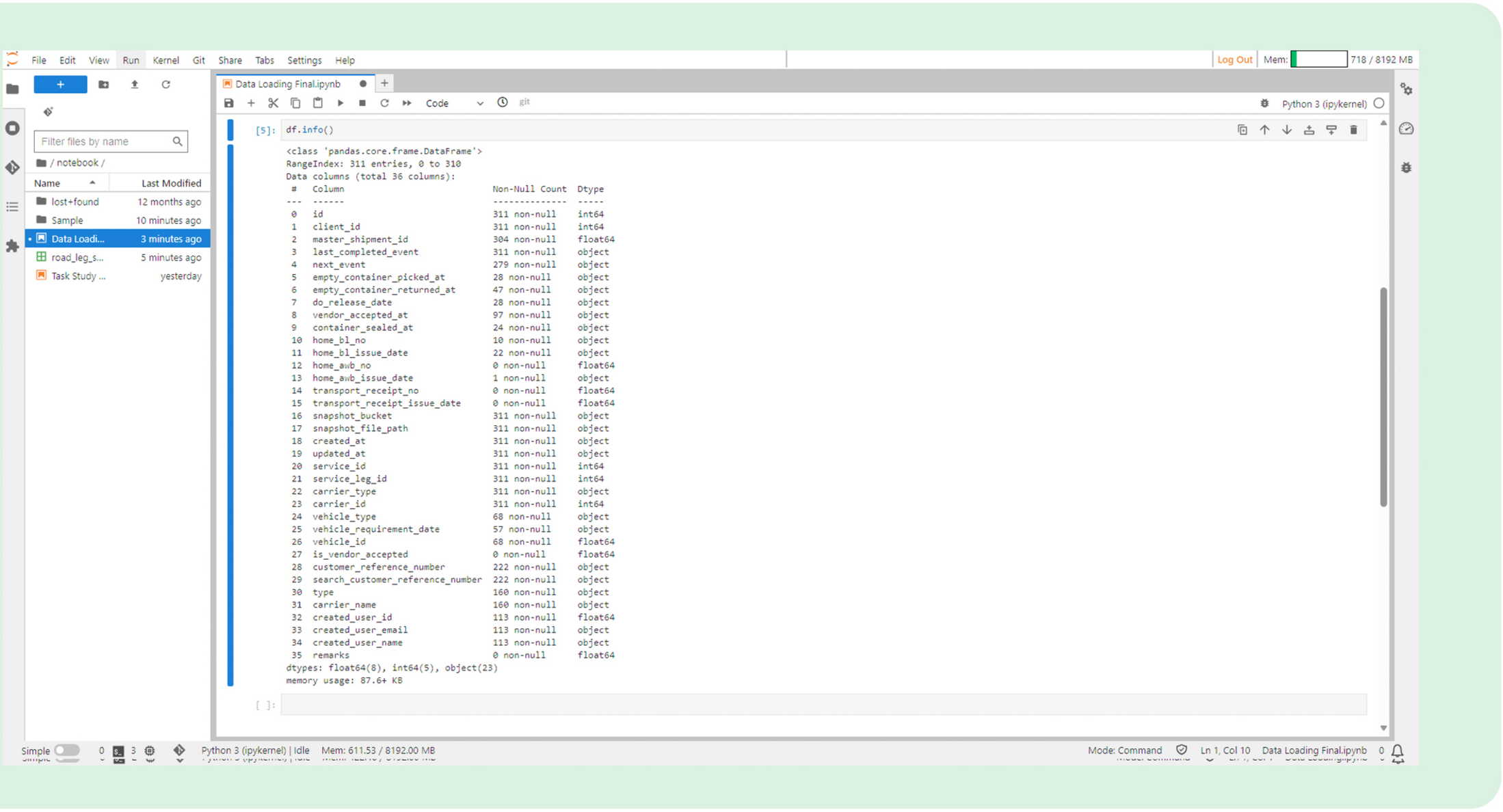
Upon reading the file, you can utilize the **'df[df.duplicated()]'** command to identify duplicate rows within the dataset.

Command	Action
df[df.duplicated()]	It identifies rows that are identical to previous rows, considering all columns. This procedure entails recognizing and removing columns containing identical data, thereby enhancing the cleanliness and accuracy of the dataset for future operations.



Step 8: To view the summary of your dataset, use the **'df.info()'** command.

Command	Action
df.info()	This method provides a summary of the DataFrame. It includes details such as the data types of each column, the number of non-null values and offering insights into the DataFrame's structure.



4. Data Analysis in Jupyter Notebook

After setting up your data in Jupyter Notebook, you can now delve into detailed analysis and exploration of your data. In ConverSight there are four categories of columns:

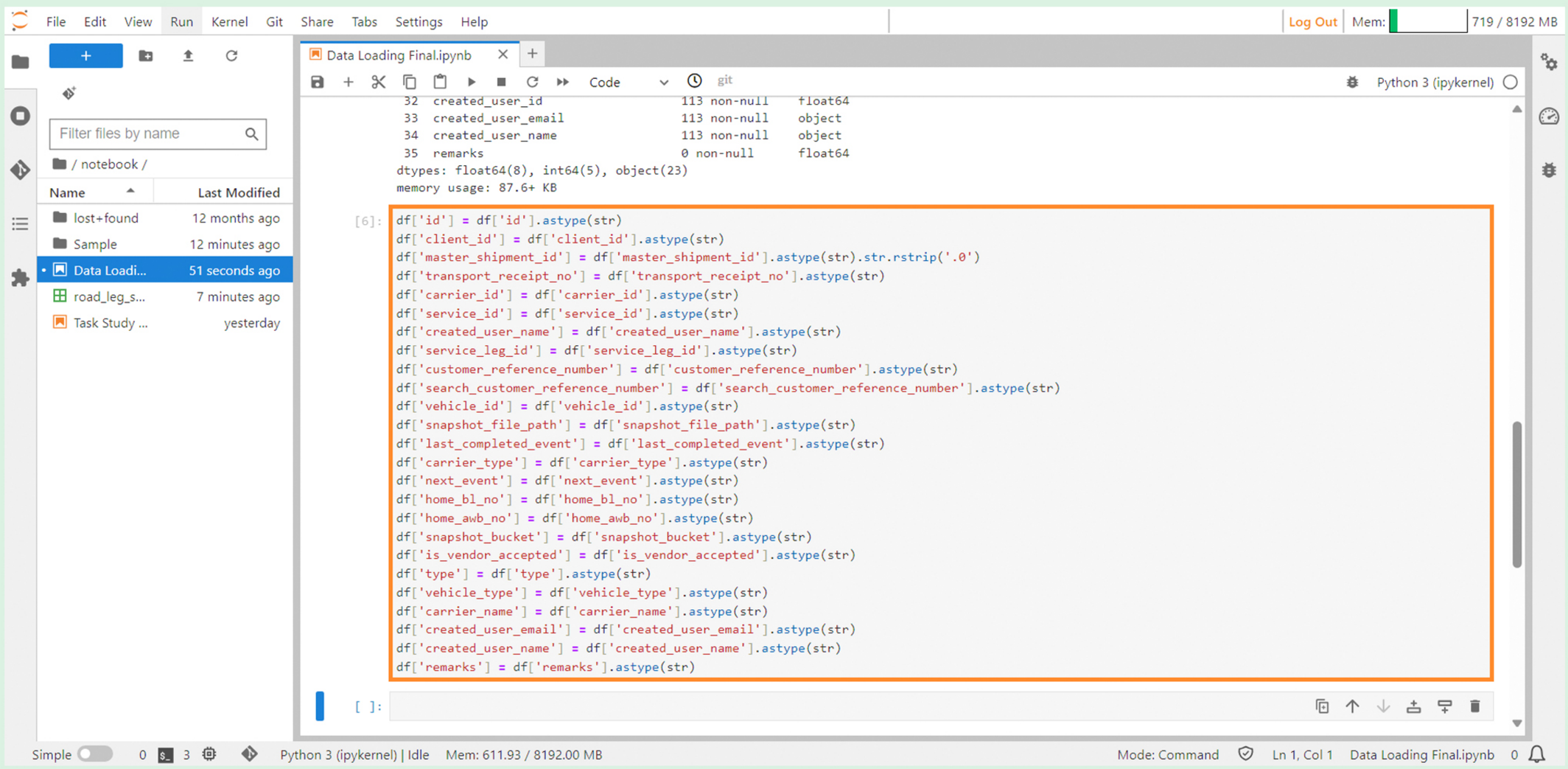
Command	Datatype Requirement	Description
Dimensions	**str or object**	Columns in this category store qualitative data or categorical variables, represented as text.
Metrics	**int or float**	Metrics category is for columns holding quantitative data or numerical values, allowing for mathematical calculations.
Date	**datetime**	Date category is specifically for columns that store date and time information, enabling time-based analysis.
Flag	**binary or boolean**	Flag columns represent binary indicators or Boolean values, often used for marking specific conditions or states.

Adjusting data types using `**astype**` ensures that the data, matches the specific categories required by ConverSight.

After making these adjustments, you can validate that your changes are correct and that the data types are as expected by calling `**ds.info()**`, which provides a summary of the DataFrame's structure including the data types of each column. This validation step helps ensure that your data is formatted correctly before further processing or analysis within ConverSight.

Sample Code

```
df['id'] = df['id'].astype(str)
df['client_id'] = df['client_id'].astype(str)
df['master_shipment_id'] = df['master_shipment_id'].astype(str).str.rstrip('.0')
df['transport_receipt_no'] = df['transport_receipt_no'].astype(str)
df['carrier_id'] = df['carrier_id'].astype(str)
df['service_id'] = df['service_id'].astype(str)
df['created_user_name'] = df['created_user_name'].astype(str)
df['service_leg_id'] = df['service_leg_id'].astype(str)
df['customer_reference_number'] = df['customer_reference_number'].astype(str)
df['search_customer_reference_number'] =
df['search_customer_reference_number'].astype(str)
df['vehicle_id'] = df['vehicle_id'].astype(str)
df['snapshot_file_path'] = df['snapshot_file_path'].astype(str)
df['last_completed_event'] = df['last_completed_event'].astype(str)
df['carrier_type'] = df['carrier_type'].astype(str)
df['next_event'] = df['next_event'].astype(str)
df['home_bl_no'] = df['home_bl_no'].astype(str)
df['home_awb_no'] = df['home_awb_no'].astype(str)
df['snapshot_bucket'] = df['snapshot_bucket'].astype(str)
df['is_vendor_accepted'] = df['is_vendor_accepted'].astype(str)
df['type'] = df['type'].astype(str)
df['vehicle_type'] = df['vehicle_type'].astype(str)
df['carrier_name'] = df['carrier_name'].astype(str)
df['created_user_email'] = df['created_user_email'].astype(str)
df['created_user_name'] = df['created_user_name'].astype(str)
df['remarks'] = df['remarks'].astype(str)
#this will modify all datatypes of the columns.
```



To fill null values in the Excel/CSV file, use the following code snippet. This code is designed to handle and replace any null values present in the dataset, ensuring a complete and more consistent dataset for further analysis or processing.

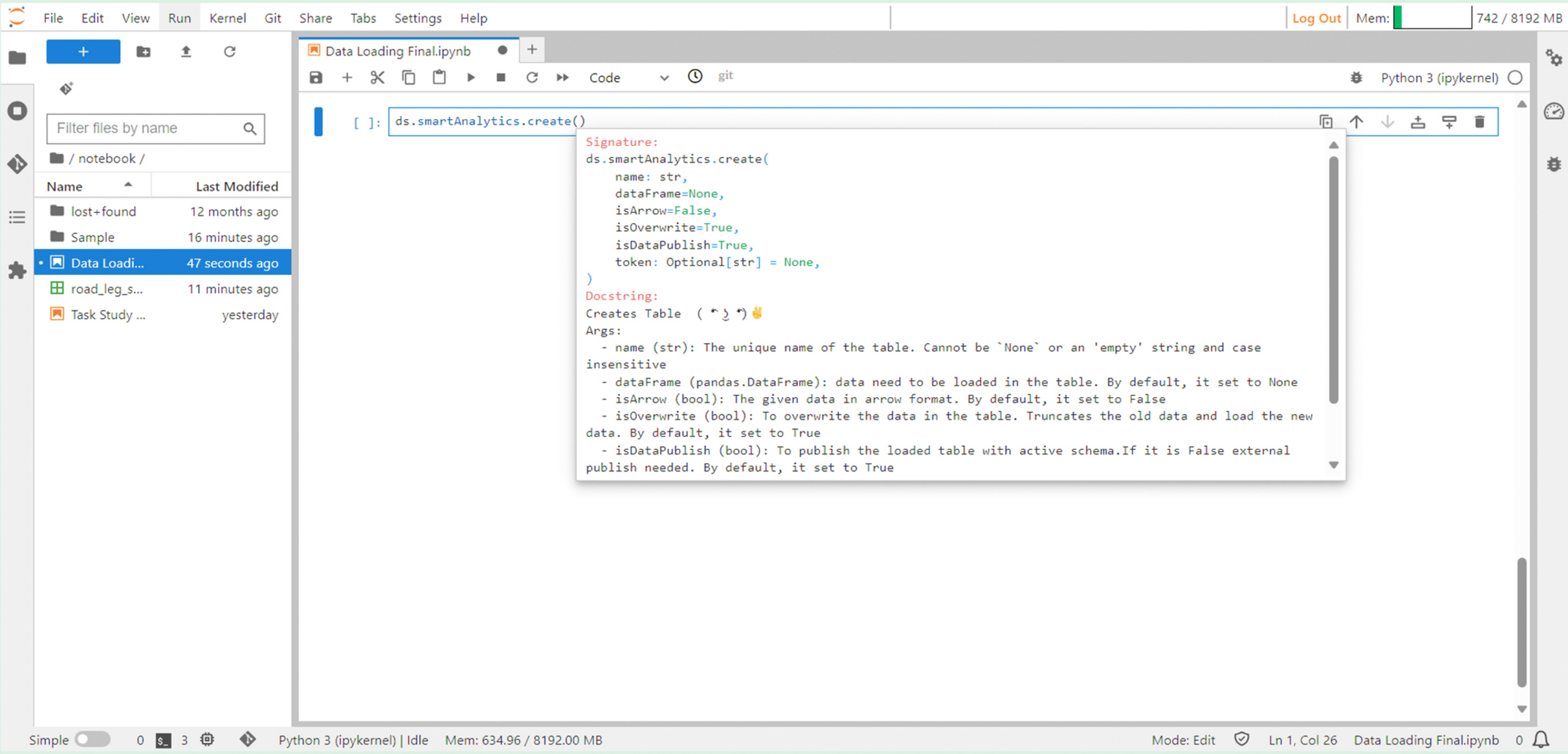
Sample Code

```
import numpy as np
#code to fill null values
for col in df.columns:
    if df[col].dtype == 'object':  # Check if the column is of string type
        df[col].fillna("", inplace=True)
        df[col]=df[col].astype(str).str.replace('nan', '')
    elif df[col].dtype == 'float64':  # Check if the column is of float type
        df[col].fillna(0.0, inplace=True)
        df[col].replace(np.NaN, 0.0, inplace=True)
    elif df[col].dtype == 'int64':  # Check if the column is of integer type
        df[col].fillna(0, inplace=True)
        df[col].replace(np.NaN, 0, inplace=True)
```


```
import numpy as np
#code to fill null values
for col in df.columns:
    if df[col].dtype == 'object':  # Check if the column is of string type
        df[col].fillna("", inplace=True)
        df[col]=df[col].astype(str).str.replace('nan', '')
    elif df[col].dtype == 'float64':  # Check if the column is of float type
        df[col].fillna(0.0, inplace=True)
        df[col].replace(np.NaN, 0.0, inplace=True)
    elif df[col].dtype == 'int64':  # Check if the column is of integer type
        df[col].fillna(0, inplace=True)
        df[col].replace(np.NaN, 0, inplace=True)
```

5. Smart Analytics Integration

Smart Analytics stands as a cornerstone in the ConverSight platform, offering a robust framework for advanced data analysis. This feature enables the performance of intricate analytics, identification of patterns and extraction of actionable insights from diverse datasets. Smart Analytics can be used to upload the data into ConverSight. By pressing the **'Shift + Tab'** button, all the available functions will be displayed, providing a convenient way to access and utilize various functions within Smart Analytics for efficient data handling and analysis within ConverSight.



The fundamental methods of Smart Analytics management empower you to initiate Smart Analytics creation, modify existing datasets, publish valuable insights and efficiently manage data deletion, ensuring optimal utilization of ConverSight's analytical capabilities.

5.1 create Method

To commence the creation of Smart Analytics, utilize the **'create'** method. This method can generate a new Smart Analytics table or update an existing one. By setting the **'isOverwrite'** parameter to True, you initiate overwriting an existing table, which involves truncating the current data. Conversely, setting it to False enables the appending of new data to the existing dataset.

Sample Code

```
ds.smartAnalytics.create(name= "road_leg_shipments", dataFrame = df, isArrow = False, isOverwrite = True,isDataPublish = True)
#this will create Smart Analytics.
```

Parameter	Type	Default Value	Description
name	str	N/A	The name of the smart analytics object being created.
dataFrame	Optional	None	The data frame containing the data to be used. If not provided, defaults to None.
isArrow	bool	False	Indicates if the data frame is in Apache Arrow format. Defaults to False.
isOverwrite	bool	True	Specifies whether to overwrite existing data with the same name. Defaults to True.
isDataPublish	bool	True	Determines if the data should be published immediately. Defaults to True.
token	Optional[str]	None	An optional authentication token for accessing secure resources. If not provided, defaults to None.

```
ds.smartAnalytics.create(name= "road_leg_shipments", dataFrame = df, isArrow = False, isOverwrite = True,isDataPublish = True)
```

```
[2024-01-11 11:59:31,594] [INFO] Setting connection details
[2023-10-27 17:09:01,756] [WARNING] Dataset [652fa7ae-oADElZnSm] is not active, current status is: Active and refreshStatus: A
ctive]
[2023-10-27 17:09:04,782] [INFO] SME data published successfully !!
{'status': 'success',
 'message': 'Smart analytics created and sme published successfully !!'}
```

5.2 update Method

To modify an existing Smart Analytics setup, you can employ the **'update'** method. If the **'isOverwrite'** parameter is set to True, the method will overwrite the current data in the table. Conversely, if set to False, it will append new data to the existing table without overwriting the current data. Below are the required arguments for updating a Smart Analytics:

Sample Code

```
ds.smartAnalytics.update(name= "road_leg_shipments", dataFrame = df, isArrow = False, isOverwrite = True,isDataPublish = True)
#this will update Smart Analytics.
```

Command	Type	Default Value	Description
name	str	N/A	Unique name of the table. Cannot be None or empty.
dataFrame	Optional	None	The data frame containing the data to be used. If not provided, defaults to None.
isArrow	bool	False	Indicates if the data frame is in Apache Arrow format. Defaults to False.

isOverwrite	bool	True	Overwrites existing data. Defaults to True.
isDataPublish	bool	True	Publishes the updated table. Defaults to True.
token	Optional[str]	None	Optional authentication token. Defaults to None.

```
ds.smartAnalytics.update(name= "road_leg_shipments", dataFrame = df, isArrow = False, isOverwrite = True,isDataPublish = True)
```

```
[2024-01-11 12:39:21,348] [INFO] Data inserted successfully !!
[2024-01-11 12:39:21,355] [INFO] Query executed successfully !!
[2024-01-11 12:39:21,716] [INFO] Request to hit createDict ====> [{dataset: 659f9fcc-NNP80qFIi, filter: Smart Analytics, table
s: [road_leg_shipments]}]
{'status': 'success', 'message': 'Smart analytics updated successfully !!'}
```

5.3 publish Method

The `**publishDataset**` method plays a crucial role in publishing Smart Analytics. Its argument determines the action, defaulting to **'update'** for updating the dictionary of specified tables. To publish newly created tables, set the argument to **'create'**. The object arguments are used to publish specific objects or update the dictionary for the given objects.

Sample Code

```
ds.smartAnalytics.publishDataset( action='update', objectList=['Data_file'] )
#this will publish your Smart Analytics in SME coaching page.
```

Parameter	Type	Default Value	Description
action	str	Update	Specifies the action: 'update' to update existing tables or 'create' to publish new tables.
objectList	List	None	List of objects (tables) to be published or updated. Defaults to None.
token	Optional[str]	None	Optional authentication token for secure access. Defaults to None if not provided.

When creating or updating Smart Analytics using the `**create**` and `**update**` methods, if the `**isDataPublish**` argument is set to True, changes are automatically reflected in SME Coaching (Subject Matter Expert). However, if set to False, the created or updated Smart Analytics is stored temporarily and requires explicit publishing using the `**publishDataset**` method to be reflected in SME.

```
ds.smartAnalytics.publishDataset('update', objectList = ['Data_file'])
```

```
[2024-05-23 06:17:53,319] [INFO] Request to publish dataset ====> {datasetId: 65e95410-dRzDo20Si, objectList: [Data
_file]}
[2024-05-23 06:17:53,723] [INFO] Dataset published successfully !!
{'status': 'success', 'message': 'success'}
```

5.4 delete Method

To remove Smart Analytics, employ the `**delete**` method and specify the Smart Analytics name as the required argument. The `**isDataPublish**` parameter determines the deletion type. If set to False, the raw table is deleted, allowing for the reuse of tables with the existing schema. On the other hand, if set to True (default), the deletion removes both the schema table and its data, effectively erasing the entire table. Choose the appropriate setting based on whether you want to retain the table's schema for potential reuse or completely remove both schema and data.

Sample Code

```
ds.smartAnalytics.delete("road_leg_shipments")
#this will delete the Smart Analytics.
```

Parameter	Type	Default Value	Description
name	str	N/A	The unique name of the table. This cannot be None or an empty string and is case-insensitive.
isDataPublish	bool	True	Indicates whether to delete the schema table (True) or just the raw table (False). Defaults to True.
token	Optional[str]	None	An optional authentication token for secure access. If not provided, defaults to None.

```
ds.smartAnalytics.delete("road_leg_shipments")
```

```
[2024-01-11 12:04:13,338] [INFO] Request to delete metadata=====> [{data_set: 659f9fcc-NNP80qFIi, table: , id: road_leg_shipme
nts, type: smartAnalytics}]
{'status': 'success',
 'message': 'table [road_leg_shipments] dropped and metadata deleted successfully !!'}
```

Join our customers who have accelerated growth with ConverSight

DAIMLER | Koken USA | CEAT
Dr. Agarwals EYE HOSPITAL | gimme SEAWEED | EXTRACT
THERMO BOND BUILDINGS | ATD AMERICAN TIRE DISTRIBUTORS | ASHOK LEYLAND
Mahindra | [MAVPAK] | salesbricks
CHAMPIONX | motherson | ISTHARA
ZOOX | meditrend | LEADER PROMOS

About ConverSight

ConverSight's Adaptive Analytics platform uses conversational AI, Natural Language Processing and machine learning to converge the distance between humans and data through data stories, presenting the meaning of data in the most effective, personalized and efficient form possible. ConverSight's patented AI business assistant, Athena, connects distributed databases to answer questions and Augment the consumers through 4 key functions: Information on demand, Automated Story Telling, Proactive Insights, and Recommended Actions.

For more information, visit www.conversight.ai

Gartner peerinsights™ | Users Love Us