Overview of Jupyter Notebook

Table of Contents

Overview of Jupyter Notebook

This guide contains all the necessary information needed to access Jupyter Notebook and to use the available data marts.

Jupyter Notebook is an open-source development environment that uses Python by default. It serves as the primary interface for you to access the Abacus Insights Platform. Abacus Insights has extended Jupyter Notebook so that you can query data marts using Amazon Redshift:

More documentation about these technologies is available at the following links:

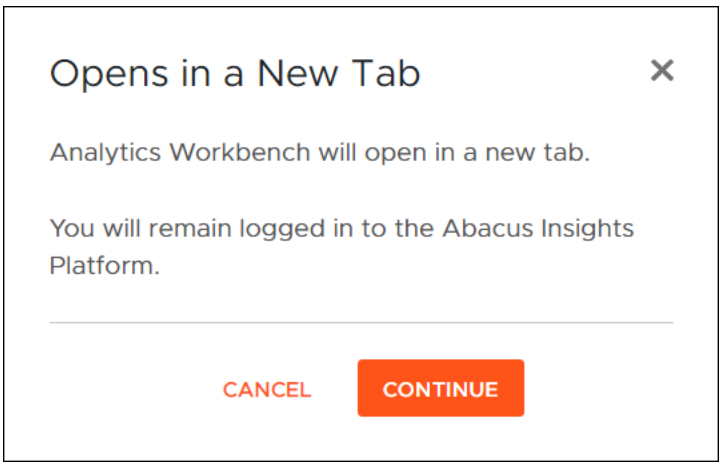
- Jupyter Notebook
- Amazon Redshift:
  - SQL Commands
  - Querying Nested Data
  - SELECT Syntax

Accessing Jupyter Notebook

To access Jupyter Notebook from the Abacus Insights Platform:

1. Log in to the Abacus Insights Platform.
2. Click **Analytics Workbench**.

   The platform alerts you that Analytics Workbench will open in a new tab.

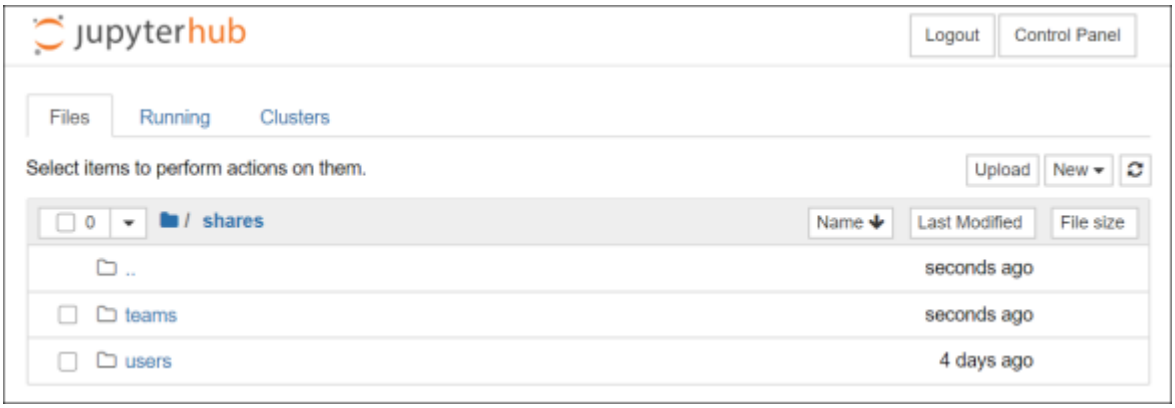


3. Click **CONTINUE**.

   Jupyter Notebook appears.

To Access Data Marts in Jupyter Notebook

If you are logging in to Jupyter Notebook for the first time, you will see the `shares` folder. Click it to see the following file tree:



- The `teams` folder contains shared folders for any teams you are a member of.

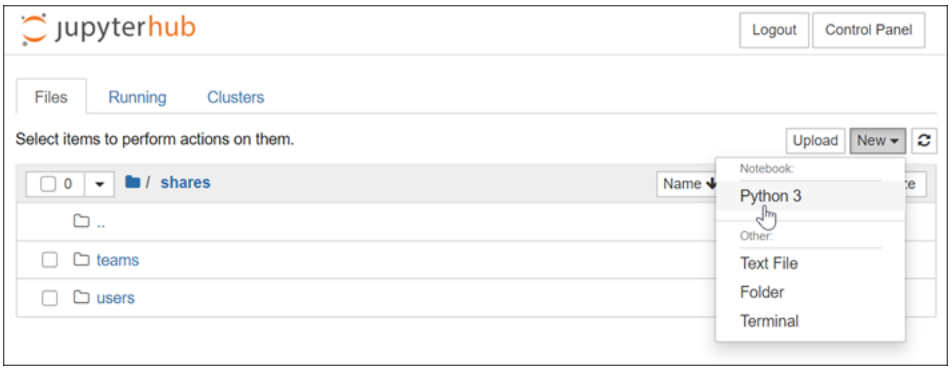
  Note: Any member of this team can access files stored in this folder.

- The `users` folder contains a personal working folder for you that is not accessible to other team members.

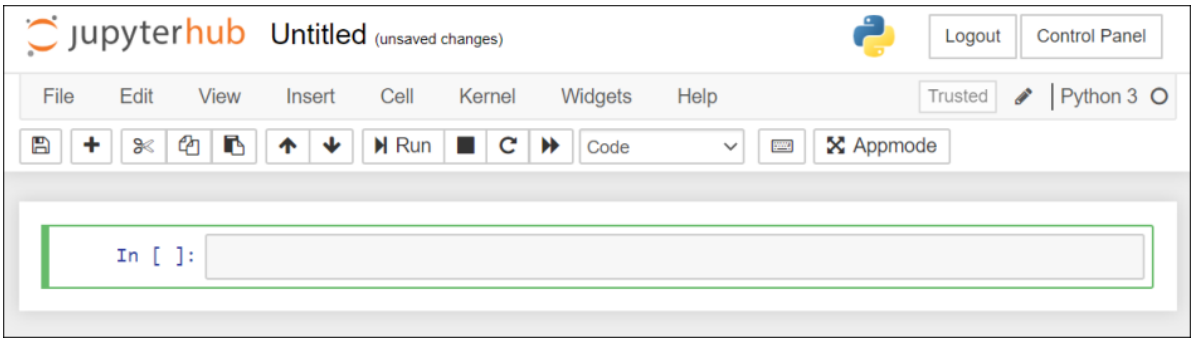
The `teams` and `users` locations are backed by S3. Documents placed in these folders in S3 are available to you in Jupyter.

To Create a New Jupyter Notebook

1. Click **New** located on the top right corner and select **Python3**.



   A new cell in Jupyter Notebook opens. In Jupyter Notebook, a cell is an area where code is written.



   Optional: By default, cells in Jupyter Notebook execute Python 3. You can also use the Redshift extension `%reload_ext redshift` to connect to and query Abacus data marts. Enter `%reload_ext redshift` and click **Run**.

```
In [ ]: #Connecting to Abacus data marts and tables
        %reload_ext redshift
```

2. The output data is saved in a variable.

   In the interest of proper coding style, install the following Python libraries to manipulate your data:

```
In [2]: # importing Python libraries

        import pandas as pd
        import nump as np
        import matplotlib as plt
```

3. Click **Run**.

   For more information on our pre-installed libraries, see "Pre-Installed Jupyter Notebook Libraries" on page 10.

4. To view the available data marts based on your team's access level, execute `%describe`.
5. Click **Run**.

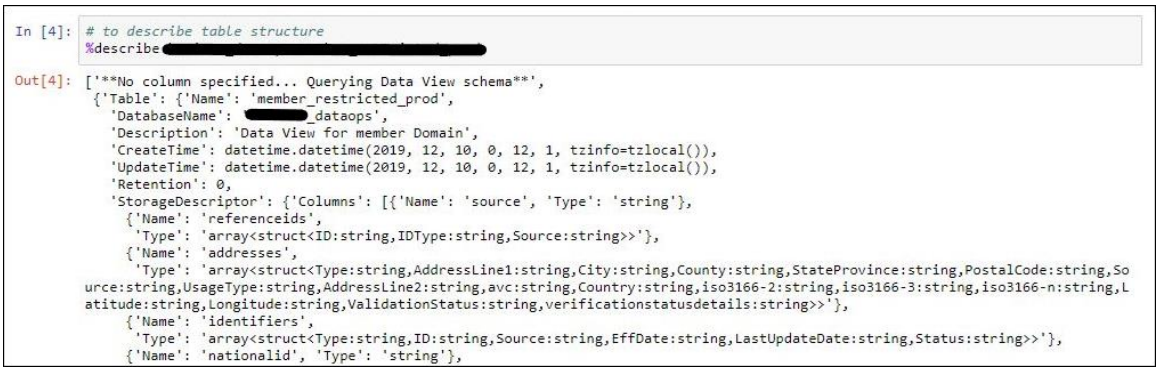
   The schema name and table names appear.

```
In [1]: #Connecting to Abacus data views/ tables
        %reload_ext redshift

In [3]: # describing available views based on team level access
        %describe

Out[3]: ['You must specify a database and a Data View to query. Type "%describe?" at the Jupyter prompt for usage details.',
         'These are your databases (schemaname) and Data Views (tablename).....',
            schemaname                          tablename
         0     dataops                      allergy_prod
         1     dataops          allergy_restricted_prod
         2     dataops                      allergy_test
```

   Note: Each team has its own schema name based on the team's access level. Table access is granted based on your team. You are not granted individual access to specific tables.

6. Replace the example schema name with the schema name assigned to your team.
7. To view the data schema of the table, execute `%describe [organization]_schemaname tablename`.
8. Click **Run**.

The attribute names in the table appear as well as the data type, such as struct, array, string, or integer.



9. Load the Redshift extension with the following script: `%reload_ext redshift`.
10. Click **Run**.

Note: The Redshift extension in Jupyter Notebook automatically trims white spaces in data fields. Therefore, you do not have to apply "trim" functions as part of your data preparation.

Redshift allows you to connect to and query Abacus data marts. You can query top-level attributes in flat marts or nested marts. For more information on flat marts and nested marts, see "Types of Abacus Data Marts" on page 12.

11. To query Abacus data marts, execute `%%redshift`, then a select statement. Provide an alias next to `%%redshift` to store the output data in a data frame variable. For example, `df`.

```
In [3]: %%redshift df
        SELECT
            <vn>.domainupdates.source,
            i.<attribute>,
            i.type,
            <vn>.basedemographics.gender,
            <vn>.personname.name,
            a.city

        from <team_name>.<view_name> <vn>, <vn>.identifiers i, <vn>.addresses a
        WHERE <vn>.domainupdates.source = 'ACX'

        Querying Database...

Out[3]: 'Result present in pandas dataframe - df'
```

Note: The scripts in this step are written in SQL, and a limit is required for each query.

For more information on `select` statements, see "Writing Queries in Amazon Redshift" on page 14.

12. Ensure that you have installed the packages mentioned previously. Apply Python functions, such as `df.head()`, to your data sets and click **Run** after each one

```
In [5]: df.head() #allows us to preview the first few values
```

Out[5]:

	source	id	type	gender	name	city
0	ACX	PAH263587412-00	CCID	F	Skylar Puckett	NORTHAMPTON
1	ACX	BIS583263080-01	CCID	F	Jessica Hanson	LYNN
2	ACX	GAR826046234-01	CCID	F	Demi Allen	LYNN
3	ACX	HOL618023971-01	CCID	F	Audrey Clarke	LYNN
4	ACX	SHA304736395-01	CCID	M	Lavelle Ayres-Perry	LYNN

Additionally, the `df.describe` function provides a profile of the data:

```
In [4]: df.describe() #gives us a profile of our data
```

Out[4]:

	source	id	type	gender	name	city
count	10	10	10	10	10	10
unique	1	10	1	2	10	3
top	ACX	GAR826046234-01	CCID	F	Sara Strother	LYNN
freq	10	1	10	9	1	8

13. In a single cell, execute the following script to display columns and rows:

```
pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)
```

14. Click **Run**.
15. To save the data on your local machine in a CSV format, execute the following script:

```
tablealias.to_csv('./test_view.csv')
```

16. Click **Run**.
17. To merge two tables, execute the following script:

```
pd.merge(table1, table2 , how='inner', on='unique attribute')
```

18. Click **Run**.

Pre-Installed Jupyter Notebook Libraries

The Abacus Insights Platform includes pre-installed, open-source libraries. You can access documentation on select libraries from the **Help** menu. For the other libraries, links to online documentation are also available.

For Basic Computation and Statistics

```
import numpy
```

- The numpy library lets you work with multidimensional arrays and high-level mathematical functions.
- For more information, see numpy_reference_scipy.org.

```
import scipy
```

- The scipy library includes functions related to calculus, linear algebra, and probability.
- For more information, see scipy.org.

For Visualization

```
import seaborn
```

- The seaborn library leverages the matplotlib package to create statistical graphics.
- For more information, see seaborn.pydata.org.

```
import plotly
```

- The plotly library allows you to create interactive, publication-quality charts.
- For more information, see plot.ly.com.

For Machine Learning

```
import sklearn
```

- Use the sklearn library for standard machine learning and data mining.
- For more information, see scikit-learn.org.

Types of Abacus Data Marts

The Abacus Insights Platform supports two types of data marts: nested marts in the Abacus Domain Model schema, and flat marts that provide joins and aggregations.

You can query these marts using dot notation. Assign table and record names to shorter variables, such as `a`.

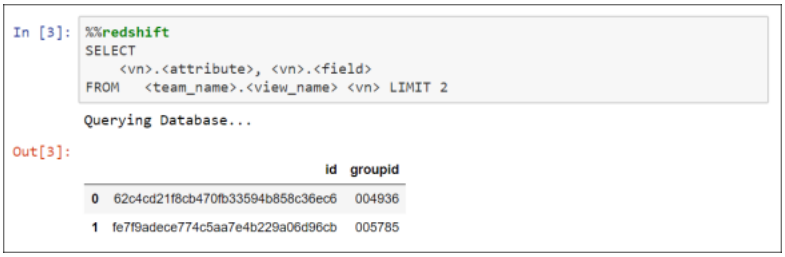
With a nested mart, you might need to interact with the following:

- Individual top-level fields, such as ID.
- Fields that contain additional nested fields within them, such as personname, which contains firstname and lastname.
- Arrays that contain multiple iterations of a nested field. For example, a member record might have multiple addresses, each of which has an addressline1.

With a flat mart, you will only interact with individual top-level fields.

The following provides different ways to query Abacus data marts.

To access top-level fields in both a flat mart and a nested mart, use the following query:



Nested fields can be added under the first string as follows:

```
In [4]: %%redshift
SELECT
    <vn>.<attribute>, <vn>.<field>,
    <vn>.personname.firstname, <vn>.personname.lastname
FROM   <team_name>.<view_name> <vn> LIMIT 2

Querying Database...

Out[4]:
```

	id	groupid	firstname	lastname
0	62c4cd21f8cb470fb33594b858c36ec6	004936	Skylar	Puckett
1	fe7f9adece774c5aa7e4b229a06d96cb	005785	Jessica	Hanson

To unpack values in an array that contains multiple iterations of a nested field, you can assign record names to short values, such as `a`.

```
In [5]: %%redshift
SELECT
    <vn>.<attribute>, <vn>.<field>,
    <vn>.personname.firstname, <vn>.personname.lastname,
    a.addressline1, a.city, a.stateprovince
FROM   <team_name>.<view_name> <vn>, <vn>.addresses a LIMIT 2

Querying Database...

Out[5]:
```

	id	groupid	firstname	lastname	addressline1	city
0	62c4cd21f8cb470fb33594b858c36ec6	004936	Skylar	Puckett	2305 Zurich Way	NORTHAMPTON
1	fe7f9adece774c5aa7e4b229a06d96cb	005785	Jessica	Hanson	7668 Branford Way	LYNN

You may print the output of your query in Jupyter Notebook, write it to a dataframe, `df`, or continue working with it in Python.

Writing Queries in Amazon Redshift

Redshift is an Amazon data warehousing service. Its distinct database design and syntax will be familiar to PostgreSQL practitioners. Common `SELECT` syntax is outlined below. More information is available in the Database Developer Guide: Select of Redshift.

```
[ WITH with_subquery [, ...] ]

SELECT

[ TOP number | [ ALL | DISTINCT ]

* | expression [ AS output_name ] [, ...] ]

[ FROM table_reference [, ...] ]

[ WHERE condition ]

[ GROUP BY expression [, ...] ]

[ HAVING condition ]

[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]

[ ORDER BY expression

[ ASC | DESC ]

[ LIMIT { number | ALL } ]

[ OFFSET start ]
```

As you interact with large volumes of data in the Abacus data lake, follow these best practices in your Redshift queries:

- Avoid using `SELECT *`.

- Set limits on your queries when you test them and until you're ready to write directly to a dataframe.
- If one table in your query is used for only predicate conditions, use subqueries as long as each subquery returns a small number of rows (less than about 200).
- Avoid using cross-joins.

You can find more guidelines and best practices for query design at Amazon Redshift best practices for designing queries.