Using Notebooks for Data Science with Denodo

Revision 20240327

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 INTRODUCTION

Notebooks can be saved as files, checked into revision control just like code, and freely shared. They run anywhere, thanks to their browser-based user interface. In this document we will go over the integration of two different notebooks with Denodo:

- Though the most influential notebook, Jupyter, has its origins in the Python programming language, it now supports many other programming languages, including R, Scala, and Julia.
- The popular Apache Spark analytics platform has its own notebook project, Apache Zeppelin, which includes Scala, Python, and SparkSQL capabilities, as well as providing visualization tools.

2 QUERYING DENODO WITH ZEPPELIN NOTEBOOK

Zeppelin is an open source notebook that is primarily used with Spark. Its backend is written in Java and front end in Angular. In this document we will focus on the use of Apache Zeppelin for Denodo, a version of the standard Apache Zeppelin distribution that introduces new features that are specifically customized to integrate smoothly with Denodo, resulting in a more integrated and enhanced experience for users.

2.1 INSTALLING APACHE ZEPPELIN FOR DENODO

Apache Zeppelin for Denodo is a web-based notebook that aims to improve the user experience while working with Denodo.

This application allows users to verify their identity and grant access to a Virtual DataPort (VDP) server by using the authentication system provided by Denodo. It also simplifies the process of running VQL statements directly within Zeppelin paragraphs, which helps to improve workflow efficiency and productivity.

The application can be downloaded through the Denodo Support Site. Note that there are two distinct versions of this application available. The **Shared Server version** is intended for deployment on servers accessed by multiple users within your network, whether concurrently or not. This version is designed with security considerations in mind and includes only the Denodo and Markdown interpreters. If additional interpreters are required, the **Standalone version** should be installed, as it is intended for use on a local machine by a single user.

The Standalone version includes the interpreters for angular, Denodo, file, JDBC, Markdown, Python, Shell, and Spark. For further interpreter installations, reference can be made to Appendix III of the Apache Zeppelin for Denodo user manual.

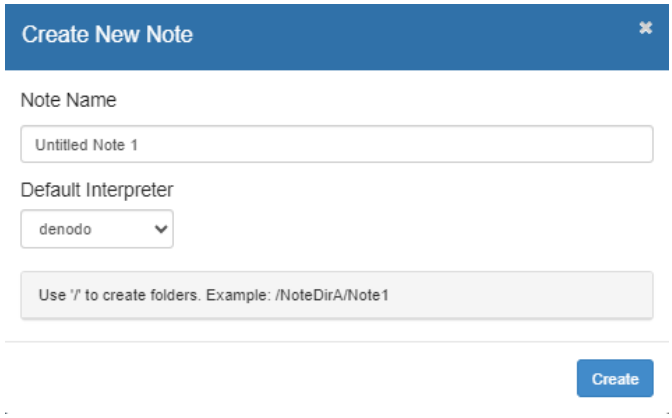
Upon downloading and extracting the zip file, initiate the application by executing **`bin/zeppelin.cmd`** on Windows or **`bin/zeppelin-daemon.sh start`** on Linux. Please be advised that the startup process may take up to a minute to complete.

To stop the zeppelin process in Linux, execute **`bin/zeppelin-daemon.sh stop`**. In Windows, terminate the cmd process.

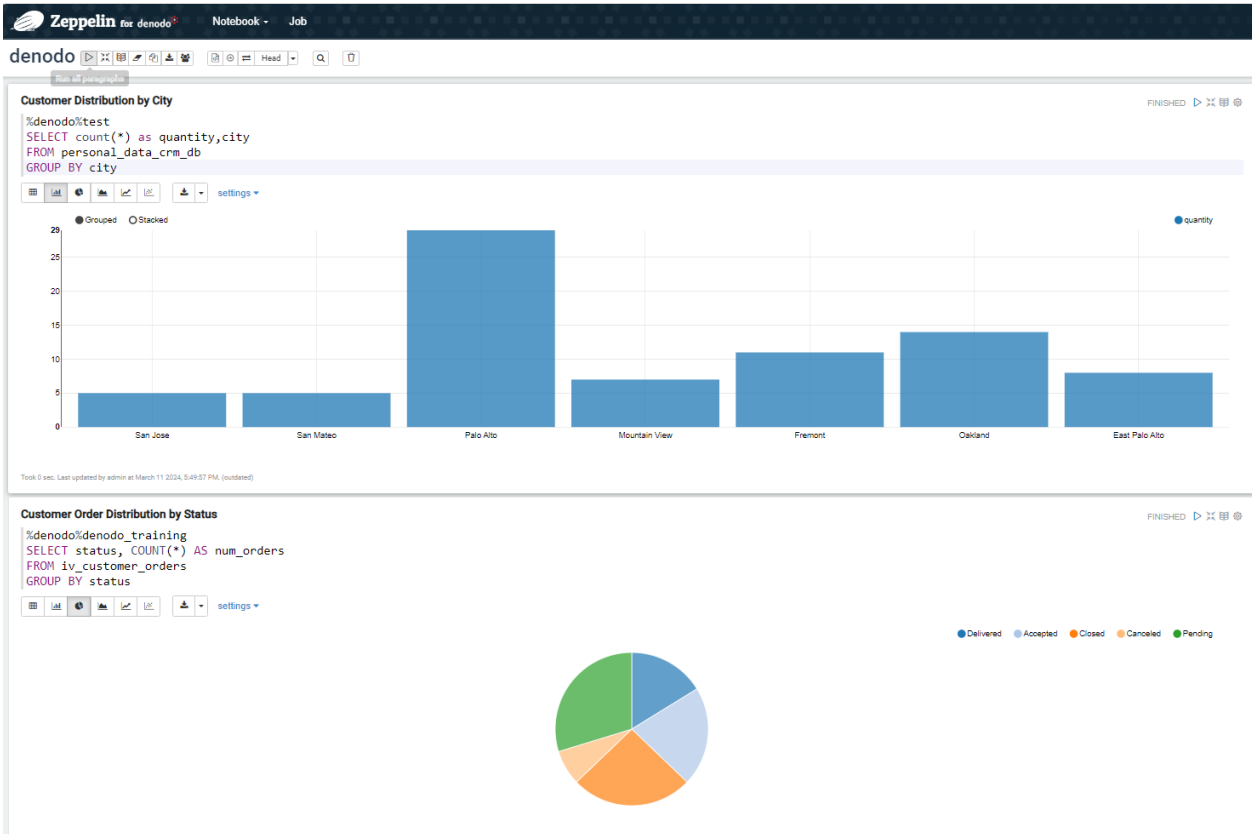
For Linux users, ensure execution permissions are set by running **`chmod u+x bin/*.sh`**.

For further information, please refer to the Apache Zeppelin for Denodo user manual.

In the notebook make sure **denodo** is selected as an interpreter and then you can use **%denodo** to run queries against the **denodo** interpreter.



Now, you can execute queries against the Denodo interpreter using the following syntax: **%denodo%<database name>**:



2.2 DENODO AS A SOURCE IN SPARK USING DATAFRAMES

A common way to use Denodo data from Spark is by moving the results of a Denodo query to a remote table in Spark.

However, in many cases it will be also interesting to ingest data directly from Denodo, without the need for replication. You can do so with Spark's DataFrames API via JDBC

To connect to Denodo from a Spark interpreter, you have to first place Denodo's JDBC driver jar in `$SPARK_HOME/jars` folder and restart Zeppelin. Once it is done, you can query a view in denodo with a code like below:

```
%spark.pyspark
url = "jdbc:vdb://localhost:9999/machine_learning"
user = "admin"
password = "your_password"
dbtable = "text_prediction"

df = sqlContext.read.format("jdbc").option("url",
url).option("user",user).option("password",password).option("dbtable",
dbtable).load()
```

3 QUERYING DENODO WITH JUPYTER NOTEBOOK

3.1 INSTALLATION

1. The easiest way to start working with Jupyter is to Install Anaconda. Anaconda is a widely used python distribution with many libraries that are used by Data Analysts.
2. Start Jupyter: `jupyter notebook`

3.2 CREATING A NOTEBOOK

A browser window will be automatically opened which will list the current working directory contents. Select **`New > Python`**

3.3 QUERYING DENODO

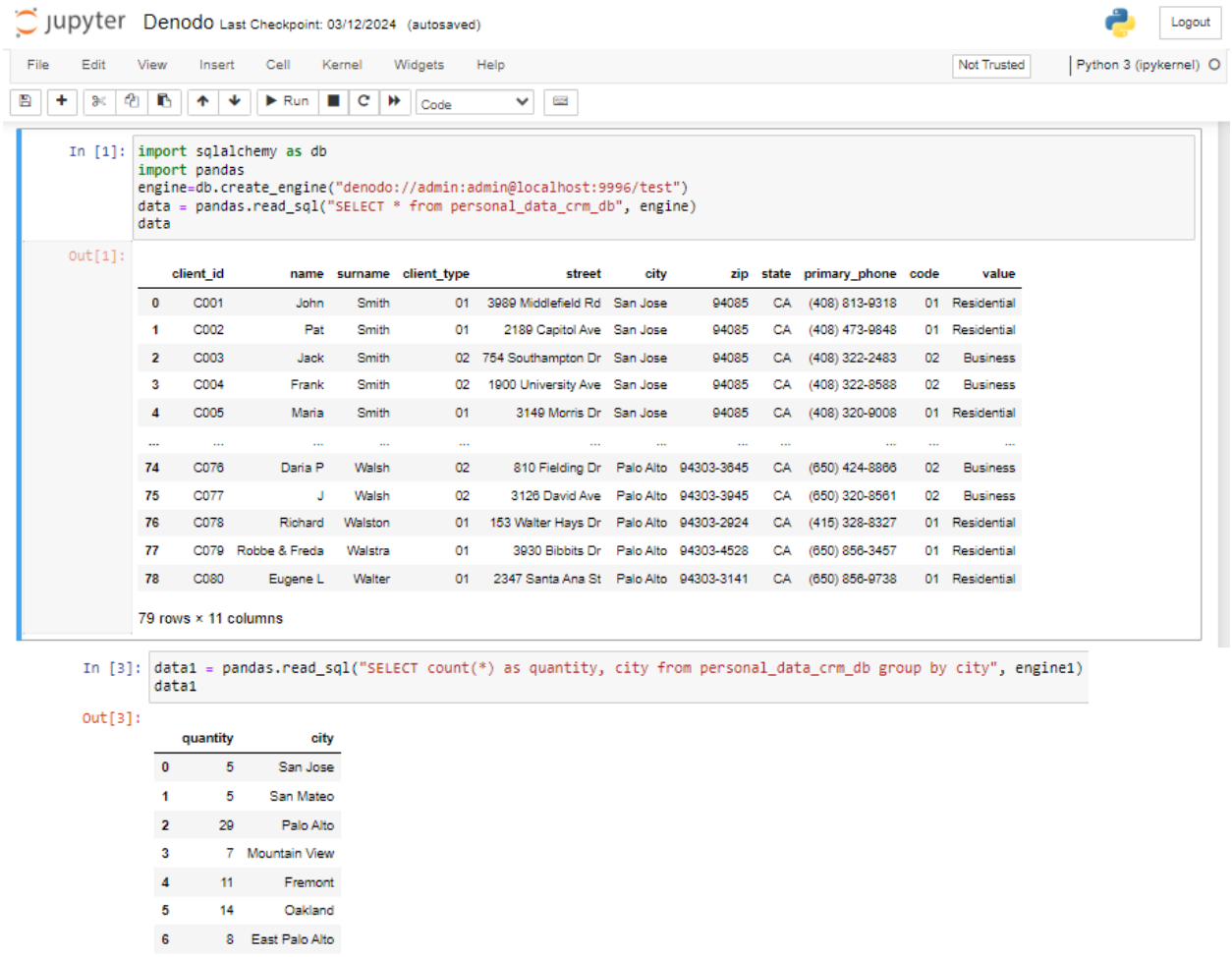
The recommended way to connect to a database using Jupyter is through the SQLAlchemy library. Once you have imported the library, we can use a standard SQLAlchemy program to query Denodo. For example:

```
# SQLAlchemy library
import sqlalchemy as db
# pandas for table display
import pandas

# create a connection using the SQLAlchemy URL pattern for Denodo dialect
engine=db.create_engine("denodo://admin:admin@localhost:9996/admin")

# Execute the query and display using Pandas
df = pandas.read_sql("SELECT * from personal_data_crm_db", engine)
df
```

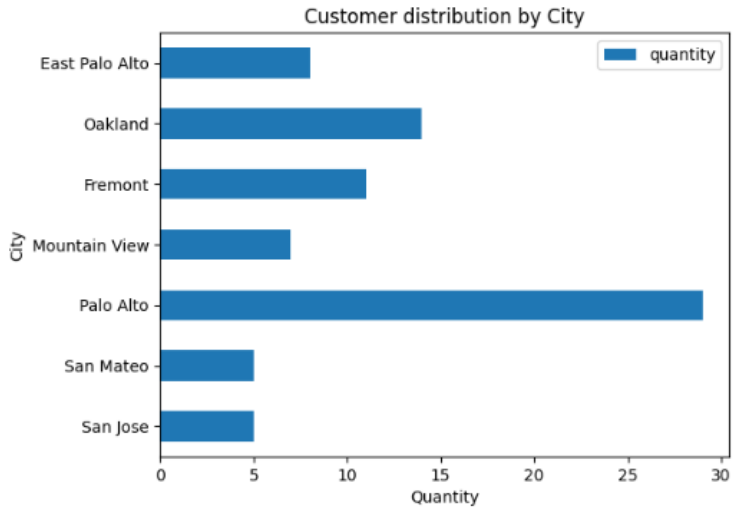
Output



We would like to refer you to this Denodo Dialect for SQLAlchemy user manual where you will find further information on the procedure for installing modules and how you can easily execute queries with SQLAlchemy.

You can also use pandas easily to draw charts:

```
ax= data1.plot.barh(x='city', y='quantity', title="Customer distribution by City", xlabel= "Quantity", ylabel="City")
```



4 REFERENCES

Apache Zeppelin for Denodo
Denodo Dialect for SQLAlchemy