teradata.

Module 2: Using R & Python with Vantage

Teradata Vantage Analytics Workshop

ADVANCED

Objectives

After completing this module, you will be able to:

- Describe different ways to run R and Python scripts
- Explain how to use R and Python on Vantage nodes
- Explain how to use R and Python on client systems
- Recognize interfaces that could be use with Vantage

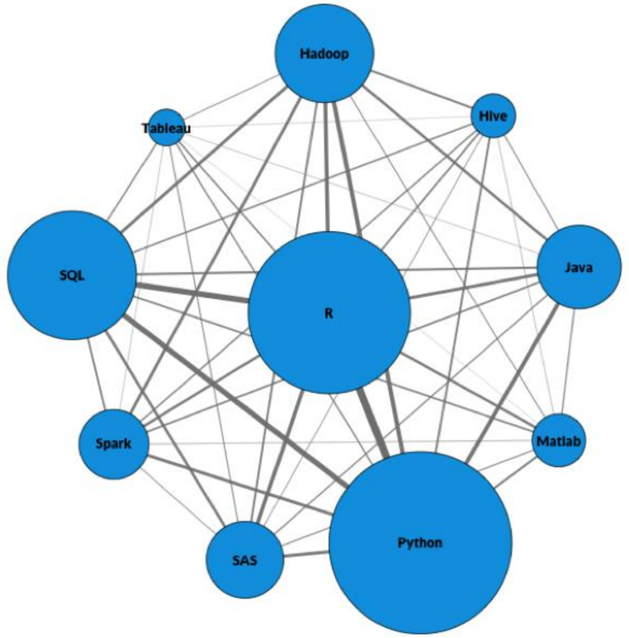
R and Python with Vantage

Data Scientist Languages

Python – An interpreted high-level programming language for general-purpose programming

R – R is a programming language and free software environment for statistical computing and graphics

SQL – A domain-specific language used in programming and designed for managing data held in a relational database management system



Before Starting, Compare Languages via 'Sessionize' (SQL vs. Python vs. R)

SQL syntax

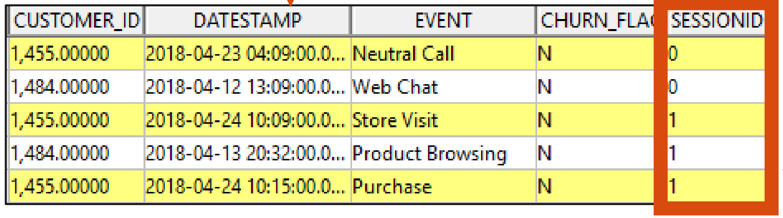
```
SELECT * FROM sessionize
 ( ON td_table
   PARTITION BY CUSTOMER_ID
   ORDER BY DATESTAMP
   USING  TIMECOLUMN('DATESTAMP')
    TIMEOUT(86400) );
```

Python syntax

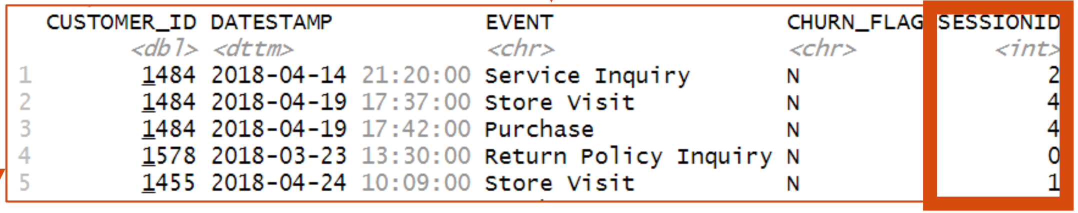
```
Psession_obj = sessionize
 ( data = td_df
 , data_partition_column = 'CUSTOMER_ID'
 , data_order_column = 'DATESTAMP'
 , time_column = 'DATESTAMP'
 , time_out = 86400.0 )

# Print result
Psession_obj
```

R syntax

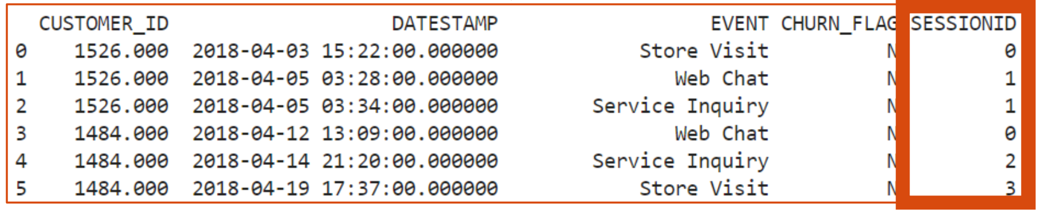
```
Rsession_obj <- td_sessionize
 ( data = td_df
 , data.partition.column = 'CUSTOMER_ID'
 , data.order.column = 'DATESTAMP'
 , time.column = 'DATESTAMP'
 , time.out = 86400 )

# Print result
Rsession_obj
```



CUSTOMER_ID	DATESTAMP	EVENT	CHURN_FLA	SESSIONID
1,455.00000	2018-04-23 04:09:00.0...	Neutral Call	N	0
1,484.00000	2018-04-12 13:09:00.0...	Web Chat	N	0
1,455.00000	2018-04-24 10:09:00.0...	Store Visit	N	1
1,484.00000	2018-04-13 20:32:00.0...	Product Browsing	N	1
1,455.00000	2018-04-24 10:15:00.0...	Purchase	N	1

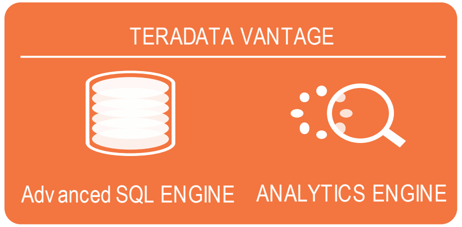


```
  CUSTOMER_ID DATESTAMP           EVENT                 CHURN_FLAG SESSIONID
        <dbl> <dttm>              <chr>                 <chr>          <int>
1        1484 2018-04-14 21:20:00 Service Inquiry       N                  2
2        1484 2018-04-19 17:37:00 Store Visit           N                  4
3        1484 2018-04-19 17:42:00 Purchase              N                  4
4        1578 2018-03-23 13:30:00 Return Policy Inquiry N                  0
5        1455 2018-04-24 10:09:00 Store Visit           N                  1
```



```
   CUSTOMER_ID                   DATESTAMP            EVENT CHURN_FLAG SESSIONID
0     1526.000  2018-04-03 15:22:00.000000      Store Visit          N         0
1     1526.000  2018-04-05 03:28:00.000000         Web Chat          N         1
2     1526.000  2018-04-05 03:34:00.000000  Service Inquiry          N         1
3     1484.000  2018-04-12 13:09:00.000000         Web Chat          N         0
4     1484.000  2018-04-14 21:20:00.000000  Service Inquiry          N         2
5     1484.000  2018-04-19 17:37:00.000000      Store Visit          N         3
```

Teradata's Journey with R and Python



Future — **Teradata Vantage**
External File System

2018 — **Teradata Vantage**
tdplyr (R), teradataml (Python), native R & Python connectivity drivers, Jupyter Hub App

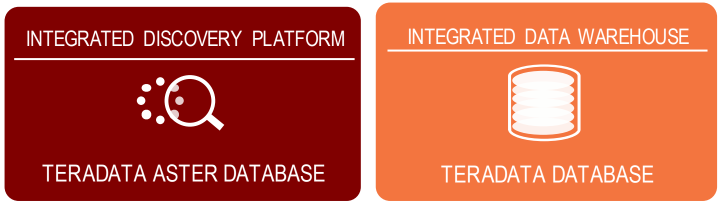
2015-2017 — **Teradata R Table Operator, Teradata Aster R**
Expanding the Open-Source R portfolio

2014 — **Teradata Script Table Operator**
Script (R/Python/Perl) execution in Teradata15.0

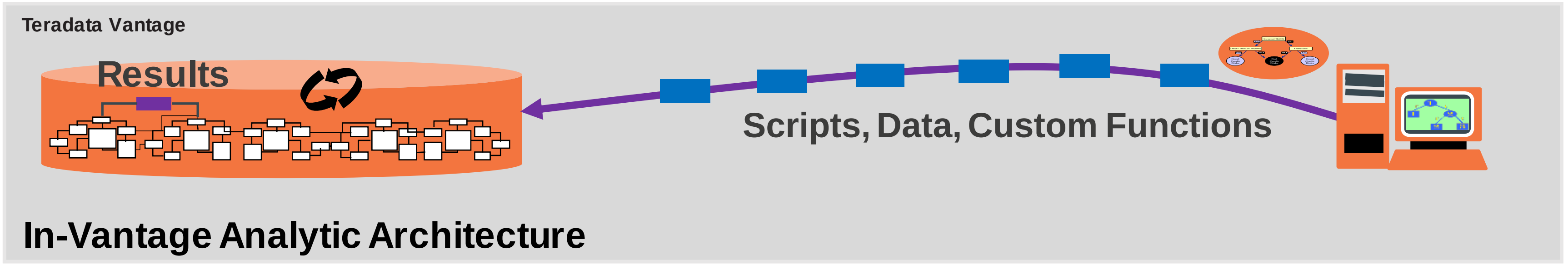
2013 — **Aster SQL-MR Stream**
Script (R/Python/Perl) execution in Aster 5.10

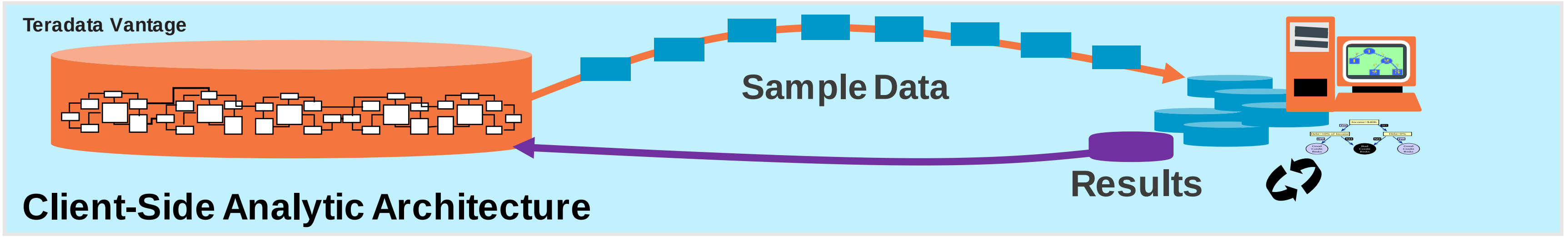
2012 — **Revolution Analytics Partnership**
Best-in-Class R + Best-in-Class EDW

2010 — **teradataR**
Teradata Warehouse Miner technology refactored in R



Analytics with External Languages


Teradata Vantage
Results
Scripts, Data, Custom Functions
In-Vantage Analytic Architecture



Approaches to Using R and Python

Powerful open-source languages used for data analysis

Most widely used in data science

Two approaches to using R and Python with Teradata Vantage:

1. **In-Vantage**
   - The language interpreters are installed on each Advanced SQL engine node (STO + ExecR)
   - Users execute scripts that stream input from and output to the database
2. **Client-side**
   - The language interpreters are installed on client machines (tdplyr + teradataml)
   - Users connect to Vantage and either:
     - Transfer data to the client to run analytics on the client
     - Use a suitable package to run analytics in-database via remote query submission

1. In-Vantage: Using SCRIPT and ExecR Table Operators

R/Python scripts are installed on Vantage

The language interpreters are installed on each Advanced SQL Engine node.

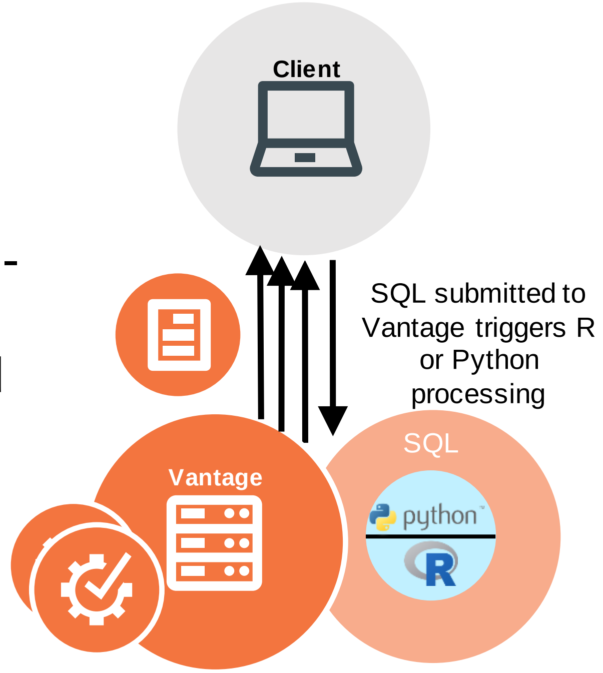
PUT-installable, RPM bundles that provide:

- Recent R and Python interpreter versions for SLES11-SP3 and SLES12-SP3
- Selections of R and Python add-on libraries for added functionality

Use SQL to call R scripts using SCRIPT and ExecR table operators.

Software runs independently on every unit of parallelism.

Ability to parallelize a given analytic task depends on location of needed input data for the task.



R and Python Bundles

Each language comes with two bundles

1. The Language Interpreter bundle
   - Contains a limited number of default add-on packages
   - Necessary pre-built dependency libraries for installation on a given SLES version
2. Add-ons bundle
   - Pre-selected set of add-on library packages for the corresponding language interpreter
   - For R, it includes 269 add-on R libraries
   - For Python, it includes 144 add-on Python libraries

Standard PUT installation can be used to install each R/Python In-Database bundle

All R/Python In-Database RPM bundles perform installations by copying files, folders, pre-built libraries and executables to target locations

Each language is independent from the other

Script Table Operator (STO)

Available since Teradata 15.00

Pipelines the database I/O to R or Python scripts executed via Linux command line through STO

Processing steps

- User “installs” their script file on Vantage
- User executes a SQL query with STO
- STO invokes R or Python, passing the installed script
- Script executes in a Linux “forked” child process
- Script reads from STDIN and writes to STDOUT
- Script runs on each unit of parallelism
- Each unit of parallelism returns its own results independently

Out-of-the-box parallelization for row or partition-based operations

- Map Reduce style programming for global operations (i.e., to combine the results from every unit of parallelism)

```python
import pandas as pd
import statsmodels.api as sm
import numpy as np
import sys
import pickle
import base64

if len(sys.argv) < 2:
    modelSaveName = 'ex5savedModel'
else:
    modelSaveName = str(sys.argv[1])

DELIMITER='\t'

# The Teradata database send in floats in scientific format
tbldata = []
while 1:
    try:
        line = raw_input()
        if line == '':   # Exit if user provides blank line
            break
        else:
            allnum = line.split(DELIMITER)
            allnum = [float(x.replace(" ","")) for x in allnum]
            tbldata.append(allnum)
    except (EOFError):   # Exit if reached EOF or CTRL-D
        break

ndl = len(tbldata)

# For AMPs that receive no data, simply exit the corresponding script instance.
if ndl==0:
    sys.exit()

df = pd.DataFrame(tbldata, columns = ['p_id','x1','x2','x3','x4','x5','y'])
del tbldata

# Create object with intercept and independent variables. The intercept column
```

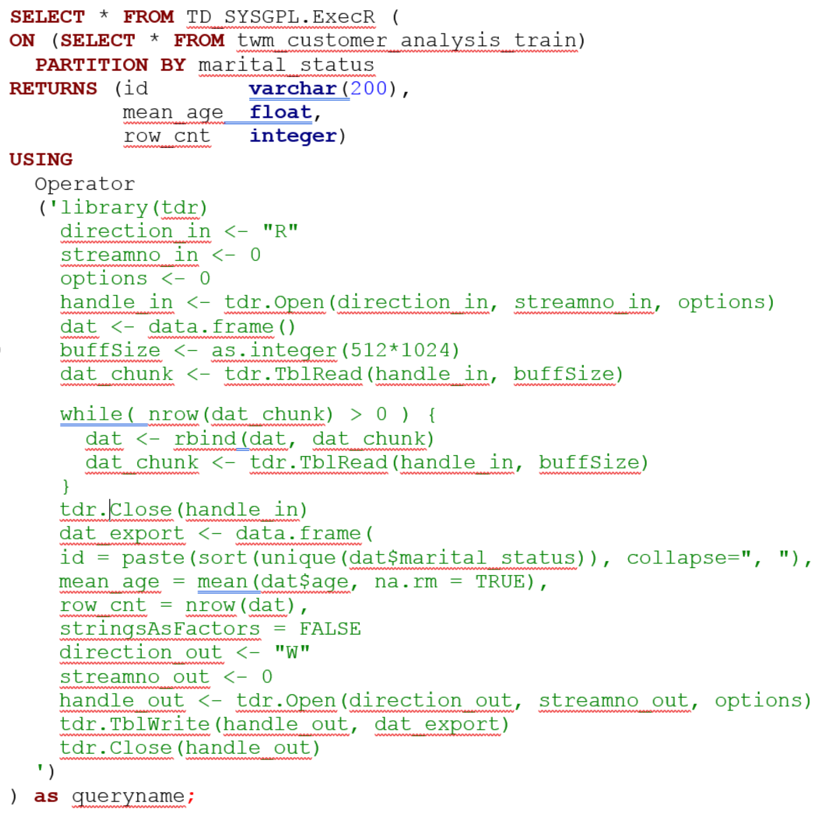
ExecR Table Operator (RTO)

Available since Teradata 15.10

Executes embedded R script with database I/O via API's

To execute an R script using RTO, a SQL Statement is constructed with:

- An ON clause specifying input(s)
- R code is passed to RTO in USING clauses
  - Contract – result schema returned (optional RETURNS clause)
  - Operator – R code to be run in parallel
- R API to access rows of data and metadata
- Same parallel processing considerations as SCRIPT



```
SELECT * FROM TD_SYSGPL.ExecR (
ON (SELECT * FROM twm_customer_analysis_train)
  PARTITION BY marital_status
RETURNS (id          varchar(200),
         mean_age    float,
         row_cnt     integer)
USING
  Operator
  ('library(tdr)
    direction_in <- "R"
    streamno_in <- 0
    options <- 0
    handle_in <- tdr.Open(direction_in, streamno_in, options)
    dat <- data.frame()
    buffSize <- as.integer(512*1024)
    dat_chunk <- tdr.TblRead(handle_in, buffSize)

    while( nrow(dat_chunk) > 0 ) {
      dat <- rbind(dat, dat_chunk)
      dat_chunk <- tdr.TblRead(handle_in, buffSize)
    }
    tdr.Close(handle_in)
    dat_export <- data.frame(
    id = paste(sort(unique(dat$marital_status)), collapse=", "),
    mean_age = mean(dat$age, na.rm = TRUE),
    row_cnt = nrow(dat),
    stringsAsFactors = FALSE
    direction_out <- "W"
    streamno_out <- 0
    handle_out <- tdr.Open(direction_out, streamno_out, options)
    tdr.TblWrite(handle_out, dat_export)
    tdr.Close(handle_out)
  ')
) as queryname;
```

SCRIPT and ExecR Table Operators: A Comparison

	SCRIPT	ExecR
Differences	Runs both R and Python script	Runs only R script
	Input from single table	Input from multiple tables
Similarities	Enable running external languages natively in Vantage	
	Out-of-the-box row and partition parallelization (single AMP)	
	Requires user programming of merging for all-AMP parallelization operations as multiple results are received	
	SQL-invoked	

2. Client-side: Using tdplyr and teradataml

R/Python forwards task to a parallel architecture

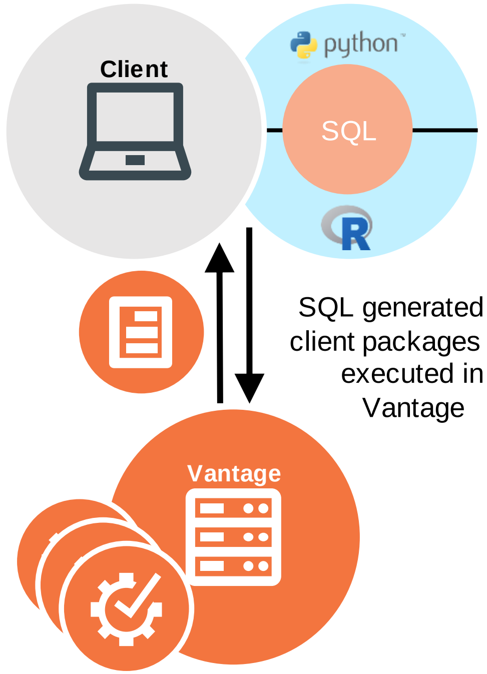
Single software instance runs on a **client** system

User invokes a Vantage analytic function from client:

- User invokes a function that is a wrapper call for task execution on Vantage
- Coding can be done in R/Python with no SQL required
- Scaling is achieved as the function is actually run on the Vantage parallel environment
- Results are stored in volatile tables on Vantage

User can then:

- Store results in persistent tables on Vantage
- Transfer/print results to client



Using R and Python with Vantage – FAQs

	Languages Running in Vantage	Languages Running on a Client to Vantage
Where is the interpreter?	Installed on each Vantage Advanced SQL Engine node	Installed on Vantage "client" machine
How do I connect to Vantage?	Directly: Interpreter runs inside the Vantage Advanced SQL Engine	Python – An IDE (Jupyter) + teradataml package R – An IDE (RStudio) + tdplyr package
Do I get In-Database Analytics?	Yes, via R/Python add-on package libraries	Yes, via R or Python function wrappers-interfaces to Vantage analytic functions
Do I have to use SQL?	Yes – to call your scripts via SCRIPT/ExecR	No
How does this help a seasoned R/Python programmer?	Existing scripts can be run in-database with minor I/O adjustments or even "as-is"	In Python: via SQLAlchemy/Pandas functionality In R: via dplyr/dbplyr functionality
Do I get scaling / parallelization?	Yes; level depends on data locale: ▪ Row-Based or Partition-Based Operations (single unit of parallelism) ✓ e.g. Model Scoring (row based) or Simultaneous Model Building (partition based) ▪ System-Wide Operations (all units of parallelism) ✓ Requires merge programming as multiple results are received from every unit of parallelism	Yes; the Vantage analytic functions do it for you

The Teradata R Package – tdplyr

The R Package – tdplyr

Combines the benefits of open-source R language environment with the massive parallel processing capabilities of Teradata Vantage

Allows users to develop and run R programs that take advantage of the Big Data and Machine Learning analytics capabilities of Vantage without writing SQL code

The Teradata R Package product is **tdplyr**

The **tdplyr** interface makes available to R users a collection of over 100 functions for analytics that reside on Teradata Vantage

- With Vantage 1.1 support of Teradata SQL driver for R as replacement for Teradata ODBC driver
- Complement to dplyr and dbplyr packages and follows the R data frame and dplyr conventions
  - Integrates dplyr verbs for data frame uses
  - Supports all dplyr base methods and scoped variants
  - R expressions mapped to SQL
- Teradata Analytic functions are prefaced with ‘TD_’ to prevent conflict with other R analytic functions with the same name

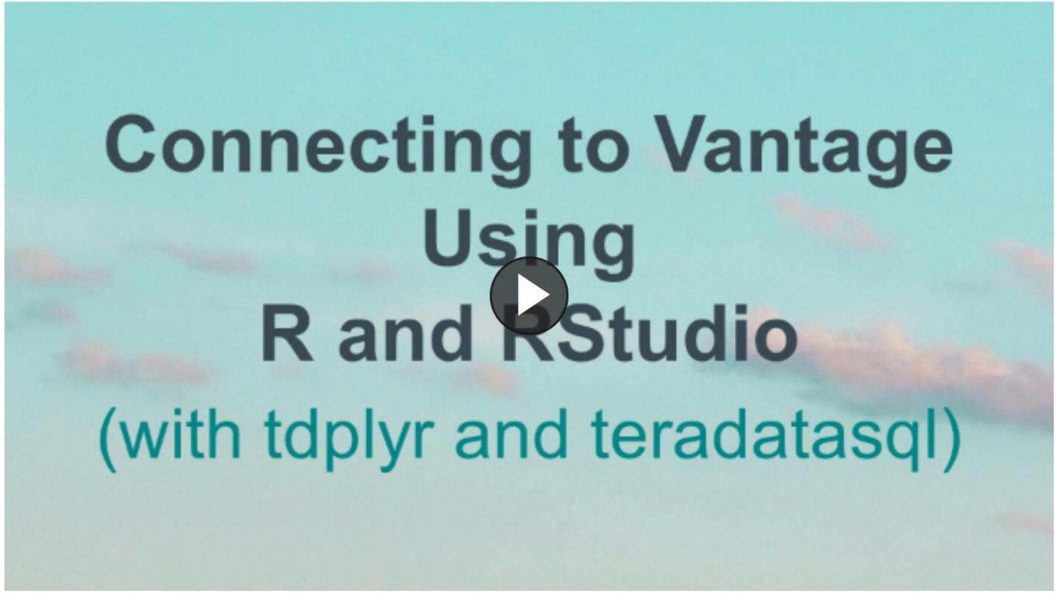
Package Requirements – tdplyr

OS – 64-bit	Development Environment	Driver Connectivity	Supported Version	Required Packages
Windows 7 and later OSX10.9 and later Linux • Ubuntu 16.04 and later • CentOS 7 and later • Red Hat 7 and later • SUSE 12 and later	Rstudio R Markdown Jupyter	Teradata ODBC 16.20 Teradata SQL Driver for R 16.20	R 3.5.1 and later versions	dplyr dbplyr DBI ODBC TeradataSQL

Software Package location – https://downloads.teradata.com/

Software Documentation – https://docs.teradata.com

Connecting to Vantage Using R and RStudio



https://web.microsoftstream.com/video/16a4c42c-f0a3-46bb-bf74-09c6179aad7d

The Teradata Python Package – teradataml

The Teradata Python Package – teradataml

The Teradata Python Package is **teradataml**, a Python library package like other open-source Python packages

Combines the benefits of the open-source Python language environment with the MPP capabilities of Teradata Vantage

Allows user to develop and run Python programs that take advantage of the Big Data and Machine Learning analytics capabilities of Vantage without writing SQL code

Follows pandas data fromat and SQLAlchemy conventions

- Teradata data frame “mimics” a pandas data frame
- Supports SQLAlchemy methods, aggregates and operators
- Supports table/data frame conversion

teradata.

Package Requirements – teradataml

OS – 64-bit	Development Environment	Driver Connectivity	Supported Version	Required Packages
Windows 7 and later OSX10.9 and later Linux • Ubuntu 16.04 and later • CentOS 7 and later • Red Hat 7 and later • SUSE 12 and later	Jupyter Notebooks JupyterLab JupyterHub	Teradata SQL Driver for Python 16.20 Teradata SQL Alchemy Dialect 16.20	Python 3.6.7 and later versions	SQLAlchemy Pandas TeradataSQLAlchemy TeradataSQL

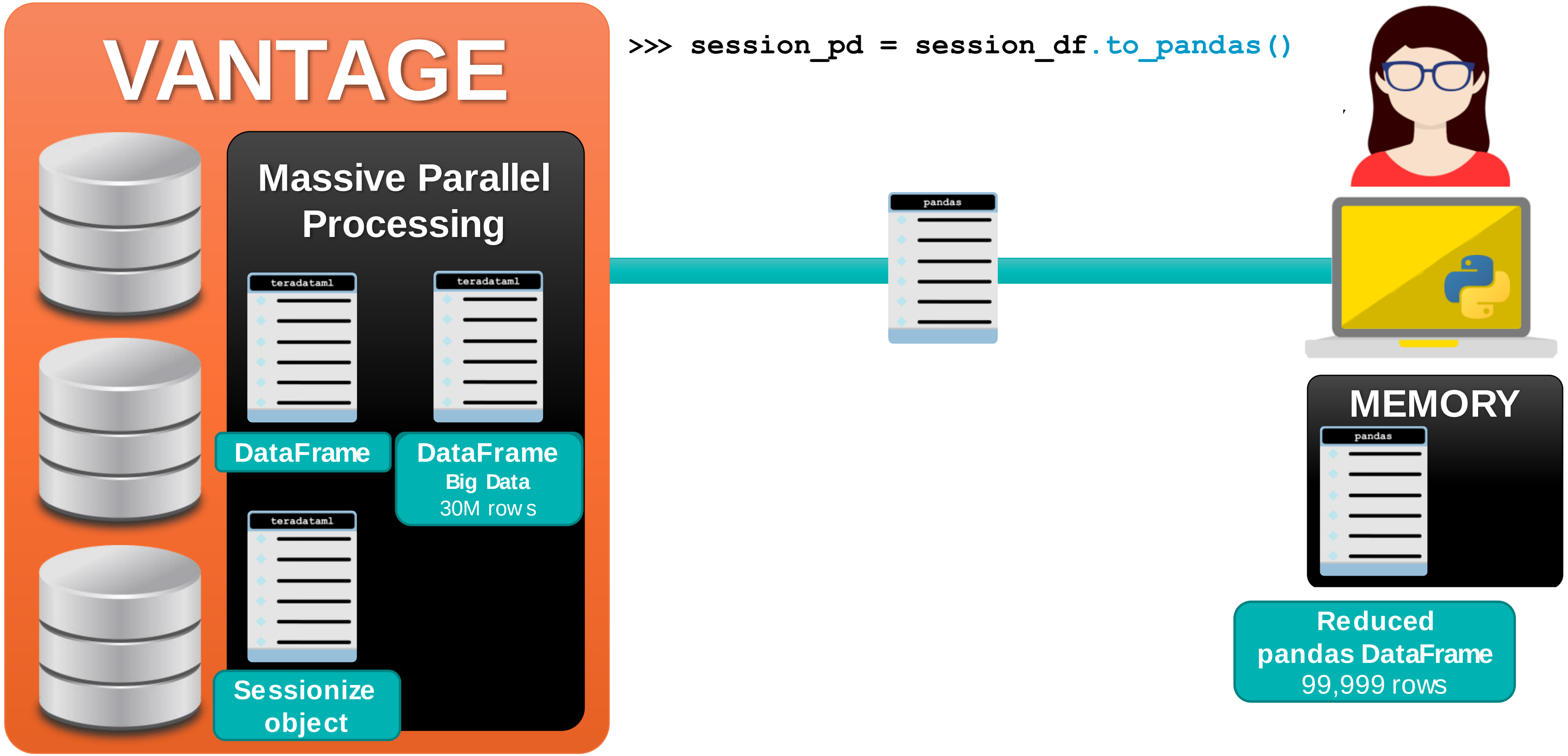
Software Package location – https://downloads.teradata.com/
https://pypi.org/project/teradataml/
https://pypi.org/project/teradatasql/
Software Documentation – https://docs.teradata.com

Teradataml DataFrame/Function Objects vs. Pandas and Numpy

teradata.

get_values()

```
>>> session_pd = session_df.to_pandas()
```

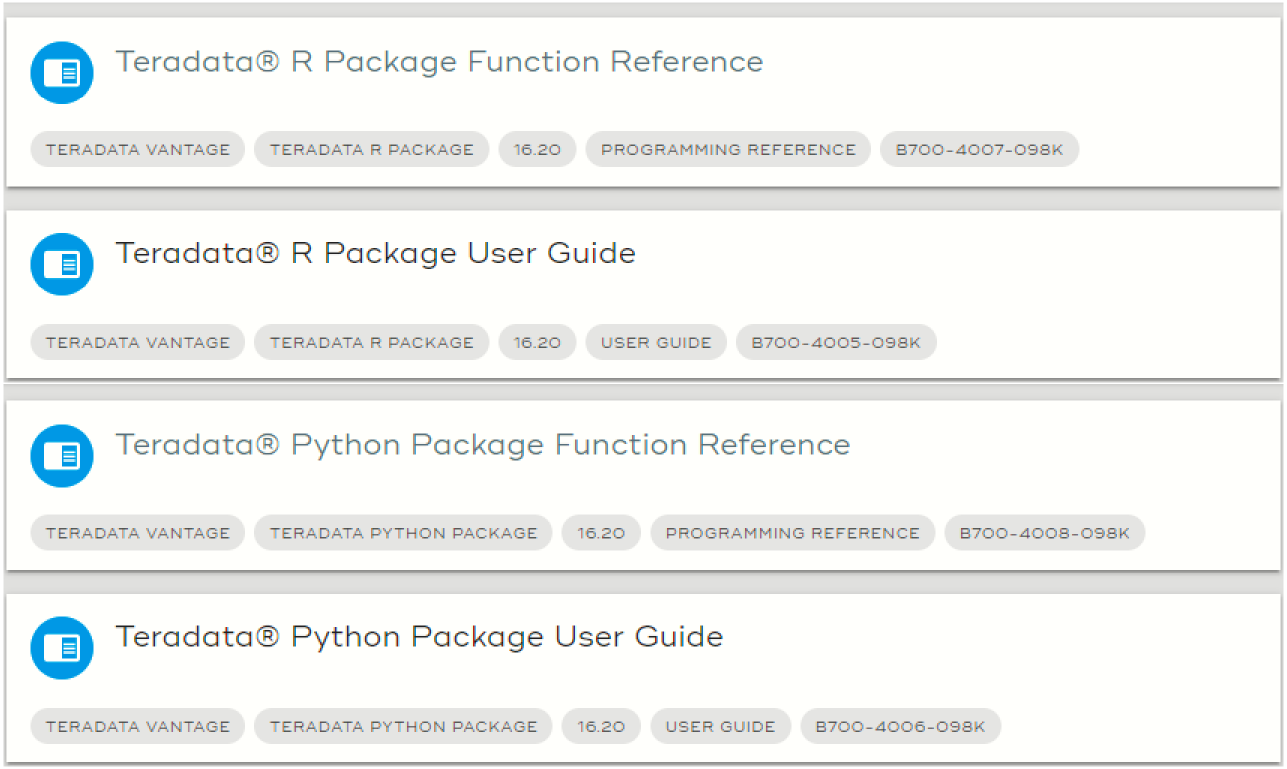


Using R and Python with Vantage

Using R and Python with Vantage

	In-Vantage Script Table Operator	Client-side Remote Processing (tdplyr/teradataml)	Client-side Local Processing (status quo)
Pros	Use Vantage MPP No restriction on Libraries/Functions Good for scoring and scalar transforms Good for simultaneous model building	Familiar environment for Users Natively uses the power of Vantage Users don't have to be SQL experts	No restriction on Libraries/Functions Familiar environment for users Simple to debug/code
Cons	Requires software installed on Advanced SQL Engine Requires libraries installed on Advanced SQL Engine Parallelization must be handled manually Complex to debug SQL expertise required	Not all Vantage functions have interfaces	Limited processing power No parallel (MPP) processing Data Transfer requirement Not suitable for big datasets

R and Python Documentation



Teradata® R Package Function Reference

TERADATA VANTAGE | TERADATA R PACKAGE | 16.20 | PROGRAMMING REFERENCE | B700-4007-098K

Teradata® R Package User Guide

TERADATA VANTAGE | TERADATA R PACKAGE | 16.20 | USER GUIDE | B700-4005-098K

Teradata® Python Package Function Reference

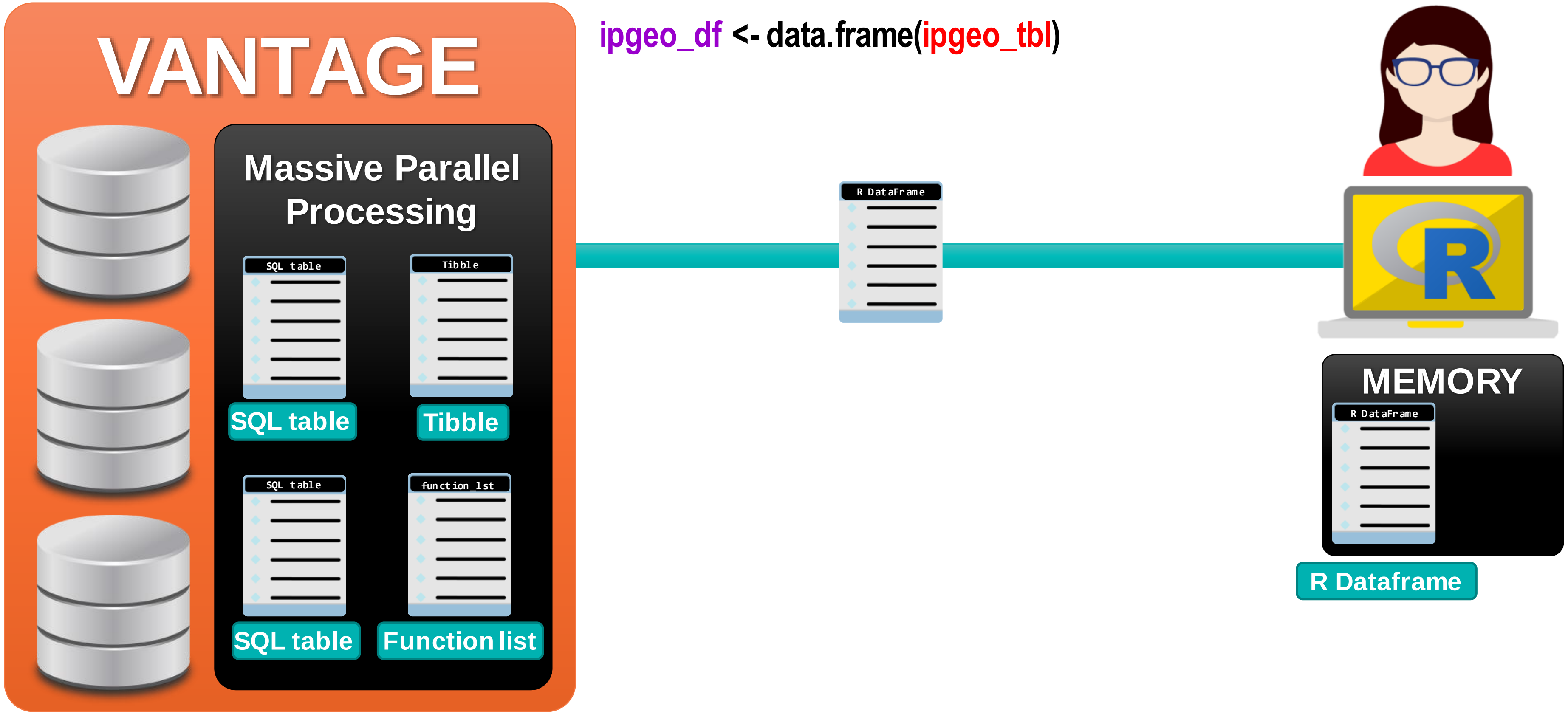
TERADATA VANTAGE | TERADATA PYTHON PACKAGE | 16.20 | PROGRAMMING REFERENCE | B700-4008-098K

Teradata® Python Package User Guide

TERADATA VANTAGE | TERADATA PYTHON PACKAGE | 16.20 | USER GUIDE | B700-4006-098K

Orange Book - R And_Python Analytics_with the SCRIPT Table_Operator Introductory_Guide

tdplyr – Tables, Tibbles, Lists and R DataFrames Objects



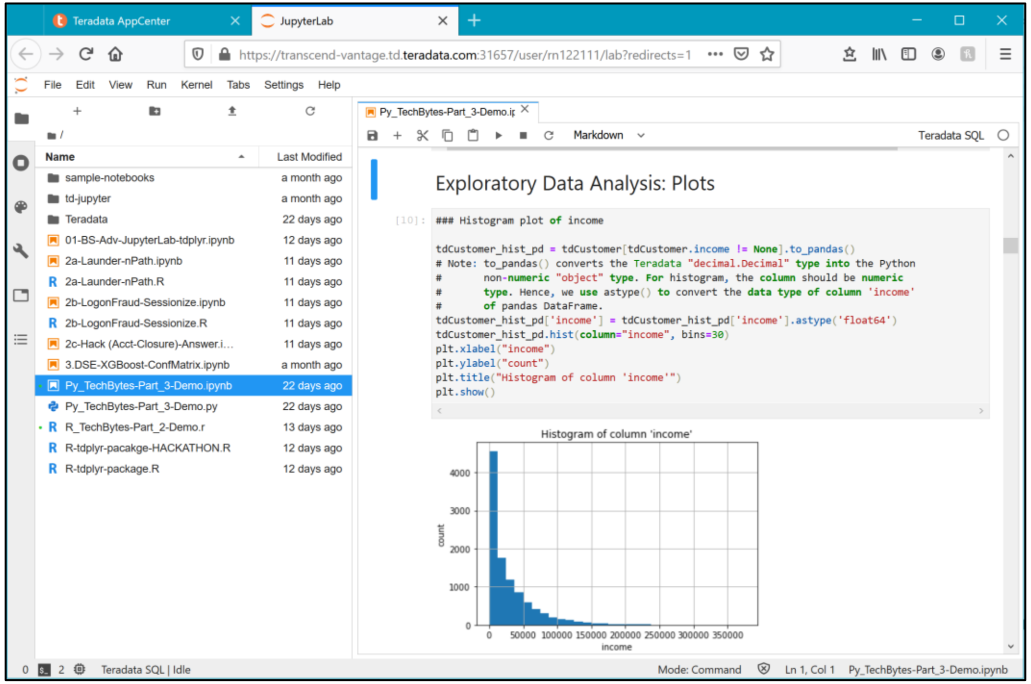
```
ipgeo_df <- data.frame(ipgeo_tbl)
```

R and Python Integrated Development Environment (IDE) Support in Vantage

JupyterLab Support in Vantage

JupyterHub from Transcend Home

- Containerized under Kubernetes
- Notebooks and Console
- SQL Kernel for Teradata
- SQL Magic Commands
- Complete R & Python support in Vantage
- Plotting
- Browse the Vantage Data Dictionary and get basic descriptive statistics
- Much more!


Teradata AppCenter
JupyterLab
https://transcend-vantage.td.teradata.com:31657/user/rn122111/lab?redirects=1
File Edit View Run Kernel Tabs Settings Help
Py_TechBytes-Part_3-Demo.ip
Markdown
Teradata SQL
Histogram of column 'income'
count
income
0 1000 2000 3000 4000
0 50000 100000 150000 200000 250000 300000 350000
0 2 Teradata SQL | Idle
Mode: Command
Ln 1, Col 1
Py_TechBytes-Part_3-Demo.ipynb

Name	Last Modified
sample-notebooks	a month ago
td-jupyter	a month ago
Teradata	22 days ago
01-BS-Adv-JupyterLab-tdplyr.ipynb	12 days ago
2a-Launder-nPath.ipynb	11 days ago
2a-Launder-nPath.R	11 days ago
2b-LogonFraud-Sessionize.ipynb	11 days ago
2b-LogonFraud-Sessionize.R	11 days ago
2c-Hack (Acct-Closure)-Answer.i...	11 days ago
3.DSE-XGBoost-ConfMatrix.ipynb	a month ago
Py_TechBytes-Part_3-Demo.ipynb	22 days ago
Py_TechBytes-Part_3-Demo.py	22 days ago
R_TechBytes-Part_2-Demo.r	13 days ago
R-tdplyr-pacakge-HACKATHON.R	12 days ago
R-tdplyr-package.R	12 days ago

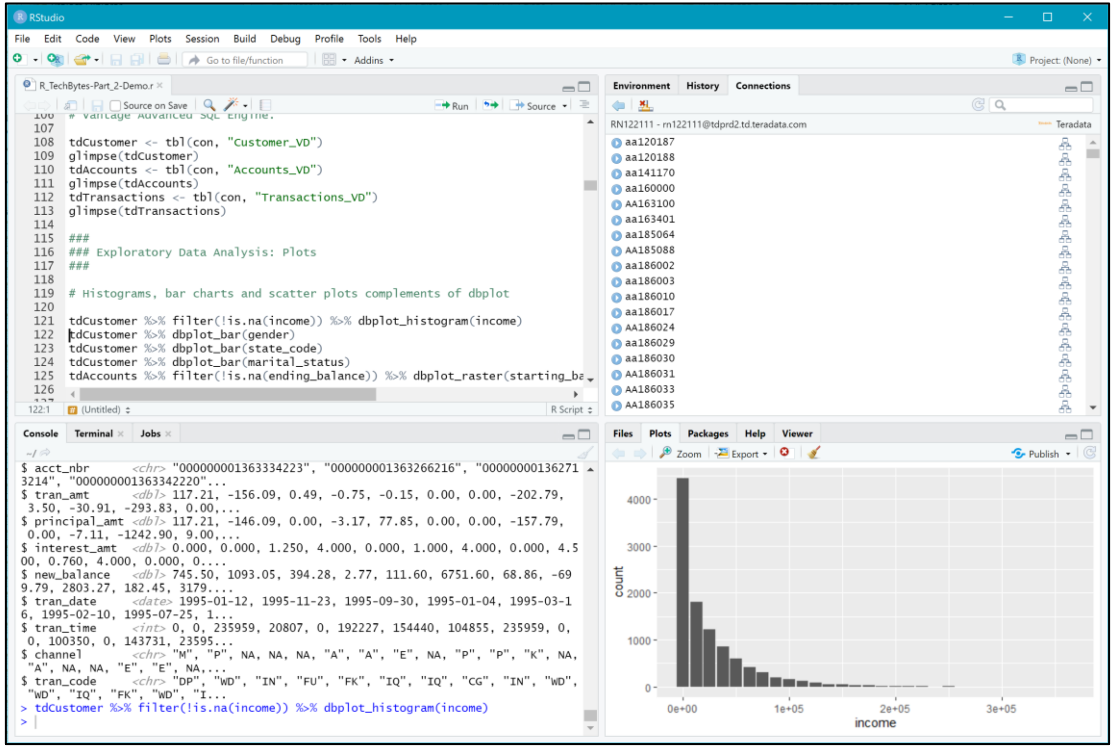
Exploratory Data Analysis: Plots

```
[10]: ### Histogram plot of income

tdCustomer_hist_pd = tdCustomer[tdCustomer.income != None].to_pandas()
# Note: to_pandas() converts the Teradata "decimal.Decimal" type into the Python
#       non-numeric "object" type. For histogram, the column should be numeric
#       type. Hence, we use astype() to convert the data type of column 'income'
#       of pandas DataFrame.
tdCustomer_hist_pd['income'] = tdCustomer_hist_pd['income'].astype('float64')
tdCustomer_hist_pd.hist(column="income", bins=30)
plt.xlabel("income")
plt.ylabel("count")
plt.title("Histogram of column 'income'")
plt.show()
```

RStudio Support in Vantage

RStudio Desktop

- Teradata SLES support
- Vantage Data Dictionary Browser
- DBI/dplyr/dbplyr Integration
- R Interfaces to Vantage Functions through tdplyr
- Plotting
- Much more!

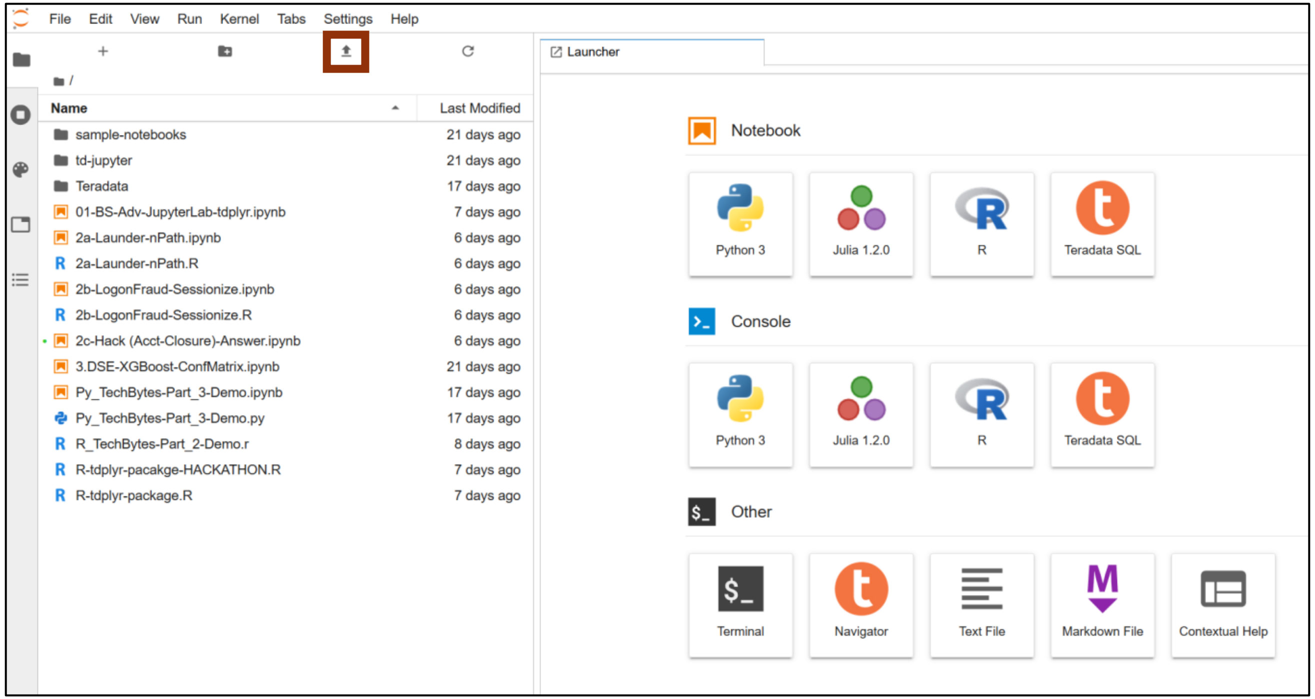


```
# Vantage Advanced SQL Engine.

tdCustomer <- tbl(con, "Customer_VD")
glimpse(tdCustomer)
tdAccounts <- tbl(con, "Accounts_VD")
glimpse(tdAccounts)
tdTransactions <- tbl(con, "Transactions_VD")
glimpse(tdTransactions)

###
### Exploratory Data Analysis: Plots
###

# Histograms, bar charts and scatter plots complements of dbplot

tdCustomer %>% filter(!is.na(income)) %>% dbplot_histogram(income)
tdCustomer %>% dbplot_bar(gender)
tdCustomer %>% dbplot_bar(state_code)
tdCustomer %>% dbplot_bar(marital_status)
tdAccounts %>% filter(!is.na(ending_balance)) %>% dbplot_raster(starting_ba
```

```
RN122111 - rn122111@tdprd2.td.teradata.com
aa120187
aa120188
aa141170
aa160000
AA163100
aa163401
aa185064
AA185088
aa186002
aa186003
aa186010
aa186017
AA186024
aa186029
aa186030
AA186031
AA186033
AA186035
```

```
$ acct_nbr      <chr> "0000000001363334223", "0000000001363266216", "00000000136271
3214", "0000000001363342220"...
$ tran_amt      <dbl> 117.21, -156.09, 0.49, -0.75, -0.15, 0.00, 0.00, -202.79,
 3.50, -30.91, -293.83, 0.00,...
$ principal_amt <dbl> 117.21, -146.09, 0.00, -3.17, 77.85, 0.00, 0.00, -157.79,
 0.00, -7.11, -1242.90, 9.00,...
$ interest_amt  <dbl> 0.000, 0.000, 1.250, 4.000, 0.000, 1.000, 4.000, 0.000, 4.5
00, 0.760, 4.000, 0.000, 0....
$ new_balance   <dbl> 745.50, 1093.05, 394.28, 2.77, 111.60, 6751.60, 68.86, -69
9.79, 2803.27, 182.45, 3179....
$ tran_date     <date> 1995-01-12, 1995-11-23, 1995-09-30, 1995-01-04, 1995-03-1
6, 1995-02-10, 1995-07-25, 1...
$ tran_time     <int> 0, 0, 235959, 20807, 0, 192227, 154440, 104855, 235959, 0,
 0, 100350, 0, 143731, 23595...
$ channel       <chr> "M", "P", NA, NA, NA, "A", "A", "E", NA, "P", "P", "K", NA,
 "A", NA, NA, "E", "E", NA,...
$ tran_code     <chr> "DP", "WD", "IN", "FU", "FK", "IQ", "IQ", "CG", "IN", "WD",
 "WD", "IQ", "FK", "WD", "I...
> tdCustomer %>% filter(!is.na(income)) %>% dbplot_histogram(income)
> 
```

Using JupyterLab

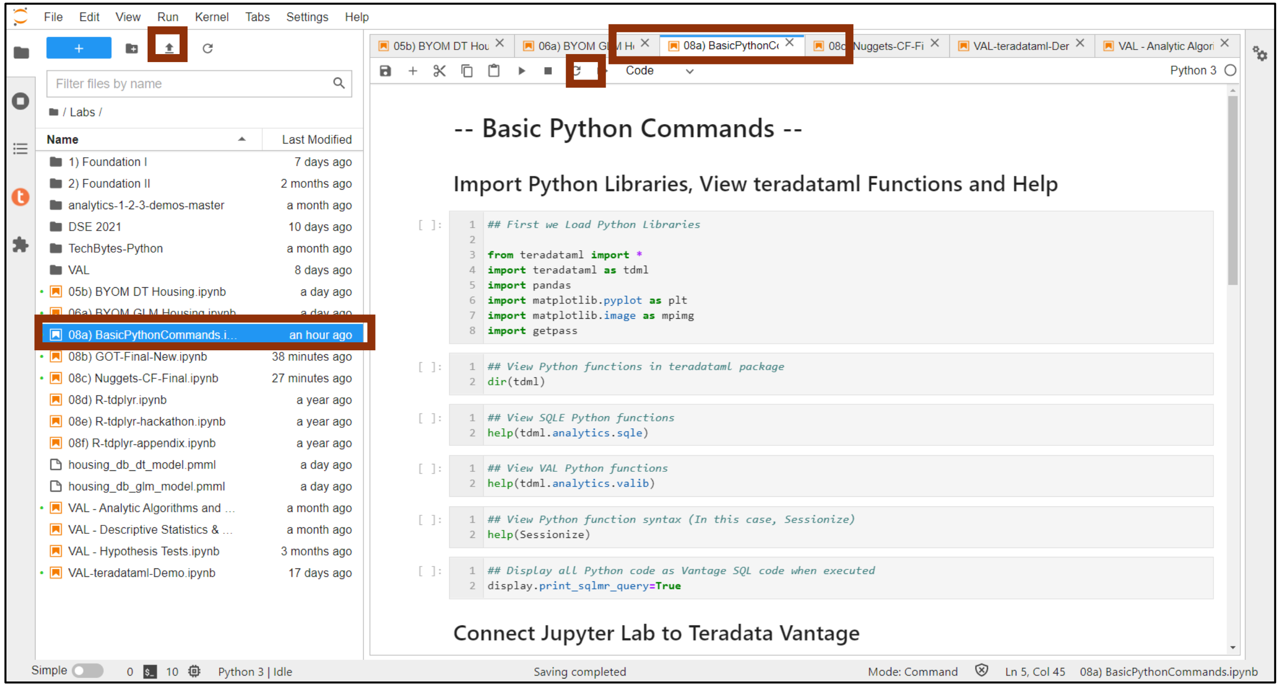


Name	Last Modified
sample-notebooks	21 days ago
td-jupyter	21 days ago
Teradata	17 days ago
01-BS-Adv-JupyterLab-tdplyr.ipynb	7 days ago
2a-Launder-nPath.ipynb	6 days ago
2a-Launder-nPath.R	6 days ago
2b-LogonFraud-Sessionize.ipynb	6 days ago
2b-LogonFraud-Sessionize.R	6 days ago
2c-Hack (Acct-Closure)-Answer.ipynb	6 days ago
3.DSE-XGBoost-ConfMatrix.ipynb	21 days ago
Py_TechBytes-Part_3-Demo.ipynb	17 days ago
Py_TechBytes-Part_3-Demo.py	17 days ago
R_TechBytes-Part_2-Demo.r	8 days ago
R-tdplyr-pacakge-HACKATHON.R	7 days ago
R-tdplyr-package.R	7 days ago

From the JupyterLab desktop, you can upload your R and Python scripts

Lab 1: Python, Open Jupyter Lab

Click the Upload button to upload the Basic Python Commands Lab script



Name	Last Modified
1) Foundation I	7 days ago
2) Foundation II	2 months ago
analytics-1-2-3-demos-master	a month ago
DSE 2021	10 days ago
TechBytes-Python	a month ago
VAL	8 days ago
05b) BYOM DT Housing.ipynb	a day ago
06a) BYOM GLM Housing.ipynb	a day ago
08a) BasicPythonCommands.i...	an hour ago
08b) GOT-Final-New.ipynb	38 minutes ago
08c) Nuggets-CF-Final.ipynb	27 minutes ago
08d) R-tdplyr.ipynb	a year ago
08e) R-tdplyr-hackathon.ipynb	a year ago
08f) R-tdplyr-appendix.ipynb	a year ago
housing_db_dt_model.pmml	a day ago
housing_db_glm_model.pmml	a day ago
VAL - Analytic Algorithms and ...	a month ago
VAL - Descriptive Statistics & ...	a month ago
VAL - Hypothesis Tests.ipynb	3 months ago
VAL-teradataml-Demo.ipynb	17 days ago

-- Basic Python Commands --

Import Python Libraries, View teradataml Functions and Help

```
## First we Load Python Libraries

from teradataml import *
import teradataml as tdml
import pandas
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import getpass
```

```
## View Python functions in teradataml package
dir(tdml)
```

```
## View SQLE Python functions
help(tdml.analytics.sqle)
```

```
## View VAL Python functions
help(tdml.analytics.valib)
```

```
## View Python function syntax (In this case, Sessionize)
help(Sessionize)
```

```
## Display all Python code as Vantage SQL code when executed
display.print_sqlmr_query=True
```

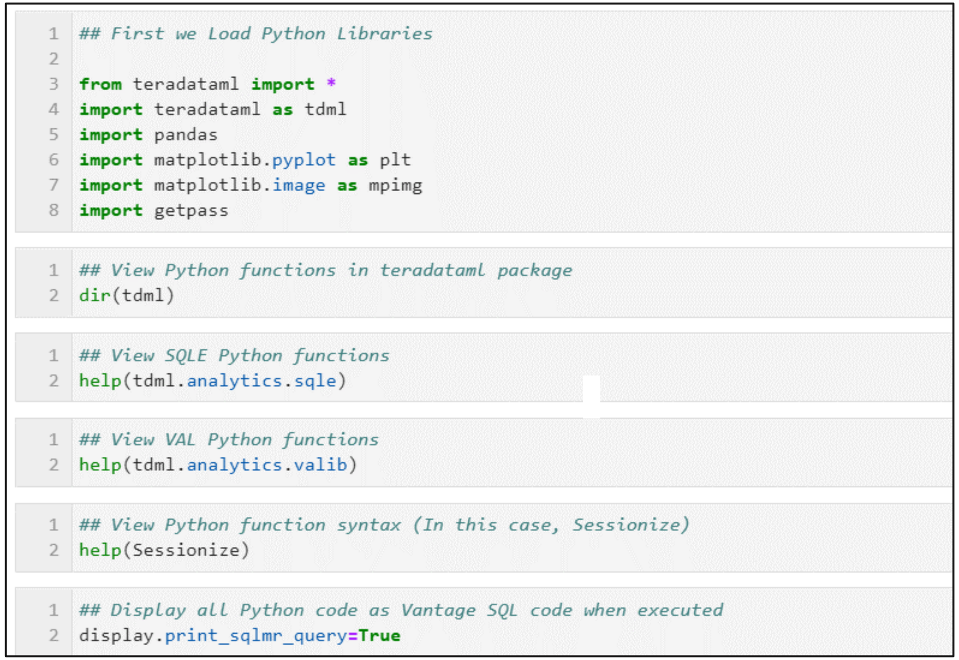
Connect Jupyter Lab to Teradata Vantage

Double clicking the Python script will open it as a tab in the main work area

Use the RUN button to execute the script statements

Lab 2: Import Python Libraries

1. Highlight Cell 1 (you'll get a blue vertical bar for that Cell)
2. Click Run button ▶. Kernel indicator circle will fill in. When finished it will be White again (sometimes it happens so fast won't see circle fill in)



```
## First we Load Python Libraries

from teradataml import *
import teradataml as tdml
import pandas
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import getpass
```

```
## View Python functions in teradataml package
dir(tdml)
```

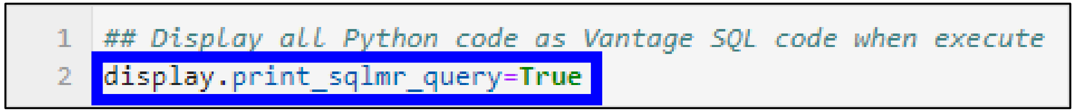
```
## View SQLE Python functions
help(tdml.analytics.sqle)
```

```
## View VAL Python functions
help(tdml.analytics.valib)
```

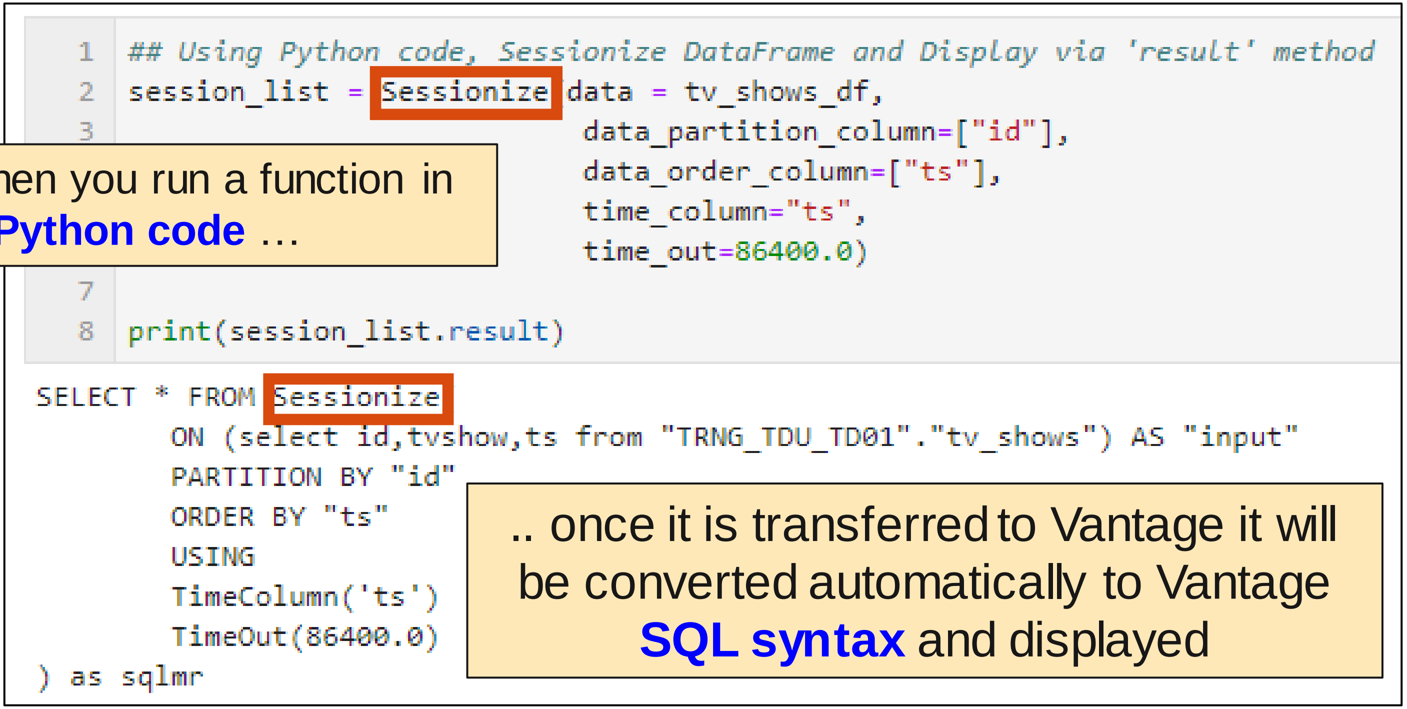
```
## View Python function syntax (In this case, Sessionize)
help(Sessionize)
```

```
## Display all Python code as Vantage SQL code when executed
display.print_sqlmr_query=True
```

View Underlying Vantage SQL Query



```
## Display all Python code as Vantage SQL code when execute
display.print_sqlmr_query=True
```



```
## Using Python code, Sessionize DataFrame and Display via 'result' method
session_list = Sessionize(data = tv_shows_df,
                          data_partition_column=["id"],
                          data_order_column=["ts"],
                          time_column="ts",
                          time_out=86400.0)

print(session_list.result)
```

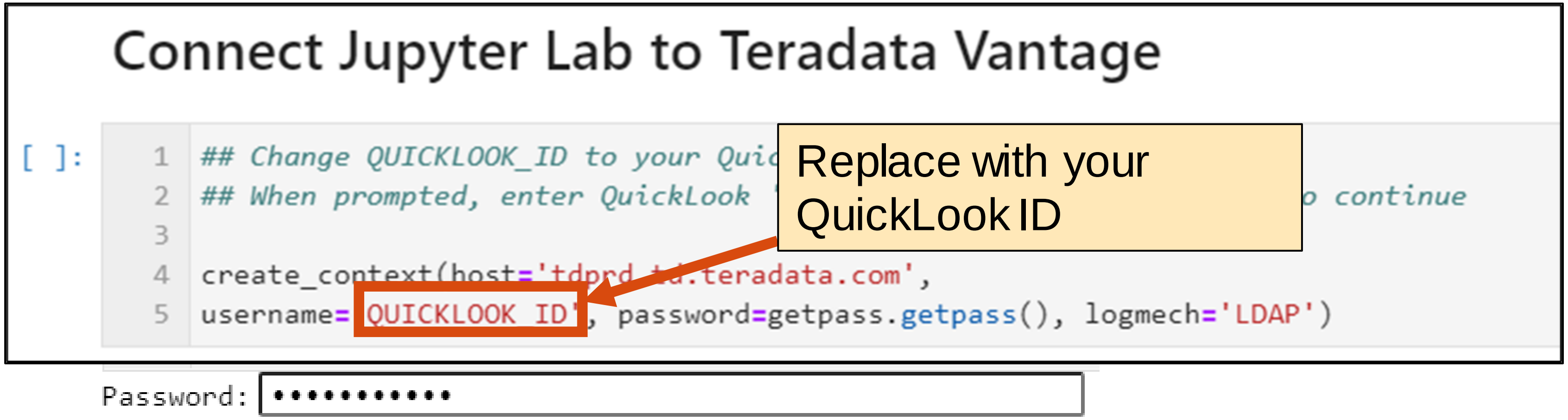
```
SELECT * FROM Sessionize(
        ON (select id,tvshow,ts from "TRNG_TDU_TD01"."tv_shows") AS "input"
        PARTITION BY "id"
        ORDER BY "ts"
        USING
        TimeColumn('ts')
        TimeOut(86400.0)
) as sqlmr
```

Later, when you run a function in **Python code** …

.. once it is transferred to Vantage it will be converted automatically to Vantage **SQL syntax** and displayed

Lab 3: Connect JupyterLab to Vantage

Run 'create_context' method to connect Python client to Vantage Cluster via JDBC



Connect Jupyter Lab to Teradata Vantage

```
[ ]:
## Change QUICKLOOK_ID to your Quic
## When prompted, enter QuickLook '                                       o continue

create_context(host='tdprd    .teradata.com',
username='QUICKLOOK_ID', password=getpass.getpass(), logmech='LDAP')
```

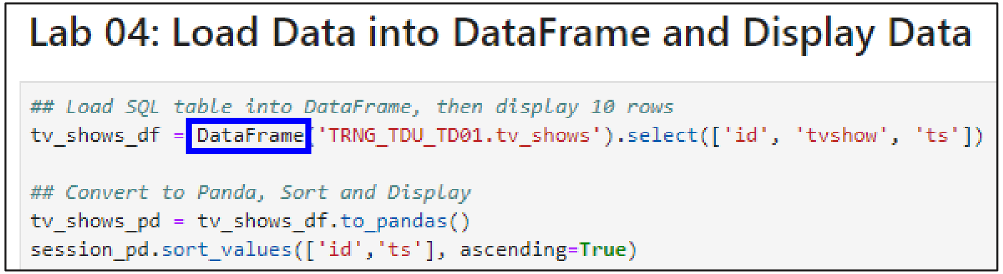
```
Password: ••••••••••••
```

Replace with your QuickLook ID

Must enter your LDAP 'password' followed by the Enter key to proceed

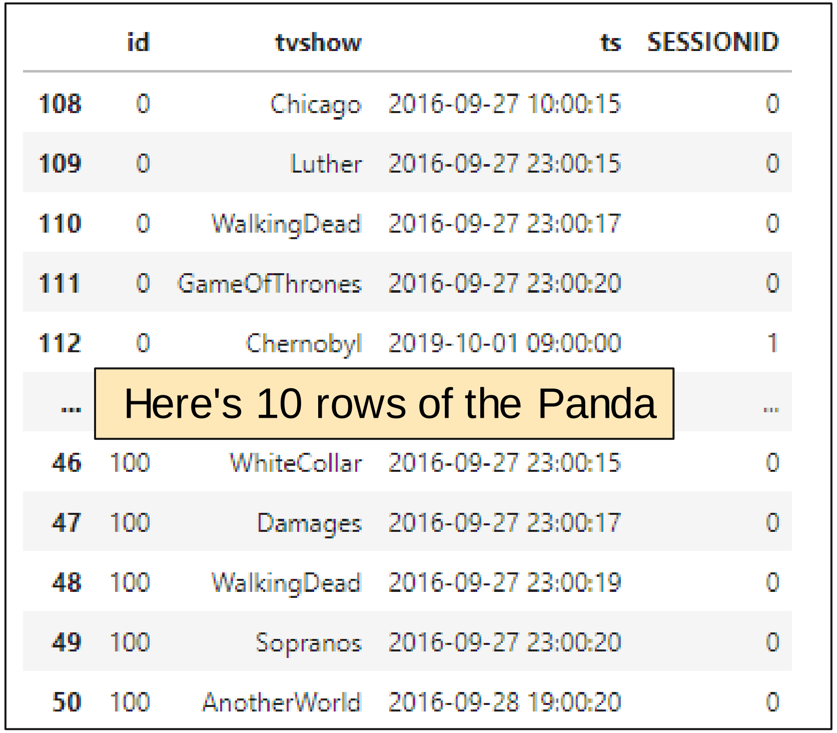
Lab 4: Load Table into Dataframe and Display

Use 'DataFrame' command to load SQL Table into a DataFrame.
Convert to Panda, Sort and Display



Lab 04: Load Data into DataFrame and Display Data

```
## Load SQL table into DataFrame, then display 10 rows
tv_shows_df = DataFrame('TRNG_TDU_TD01.tv_shows').select(['id', 'tvshow', 'ts'])

## Convert to Panda, Sort and Display
tv_shows_pd = tv_shows_df.to_pandas()
session_pd.sort_values(['id','ts'], ascending=True)
```



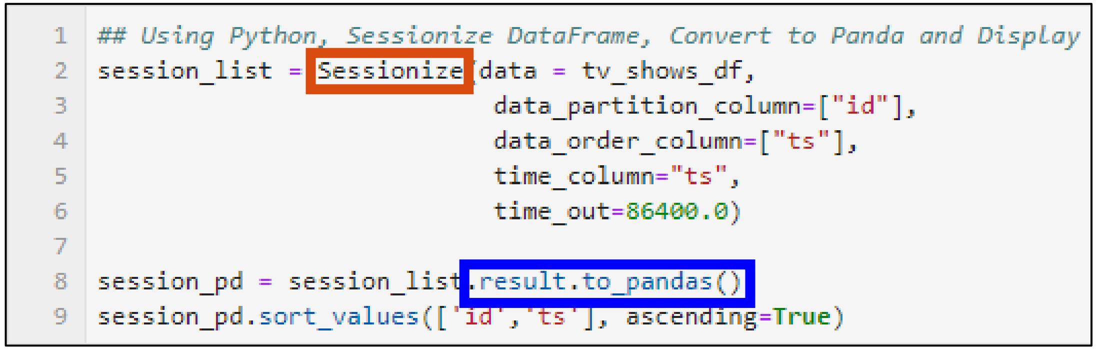
	id	tvshow	ts	SESSIONID
108	0	Chicago	2016-09-27 10:00:15	0
109	0	Luther	2016-09-27 23:00:15	0
110	0	WalkingDead	2016-09-27 23:00:17	0
111	0	GameOfThrones	2016-09-27 23:00:20	0
112	0	Chernobyl	2019-10-01 09:00:00	1
...				...
46	100	WhiteCollar	2016-09-27 23:00:15	0
47	100	Damages	2016-09-27 23:00:17	0
48	100	WalkingDead	2016-09-27 23:00:19	0
49	100	Sopranos	2016-09-27 23:00:20	0
50	100	AnotherWorld	2016-09-28 19:00:20	0

Here's 10 rows of the Panda

teradata.

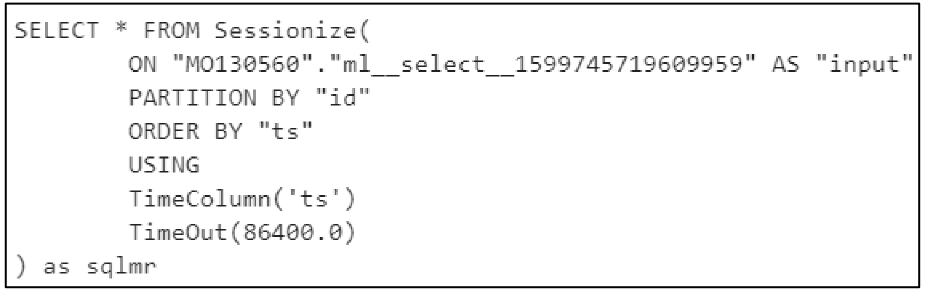
Lab 5a: Sessionize the DataFrame

Note use of the 'result' and 'to_pandas'. This converts DataFrame into a Panda

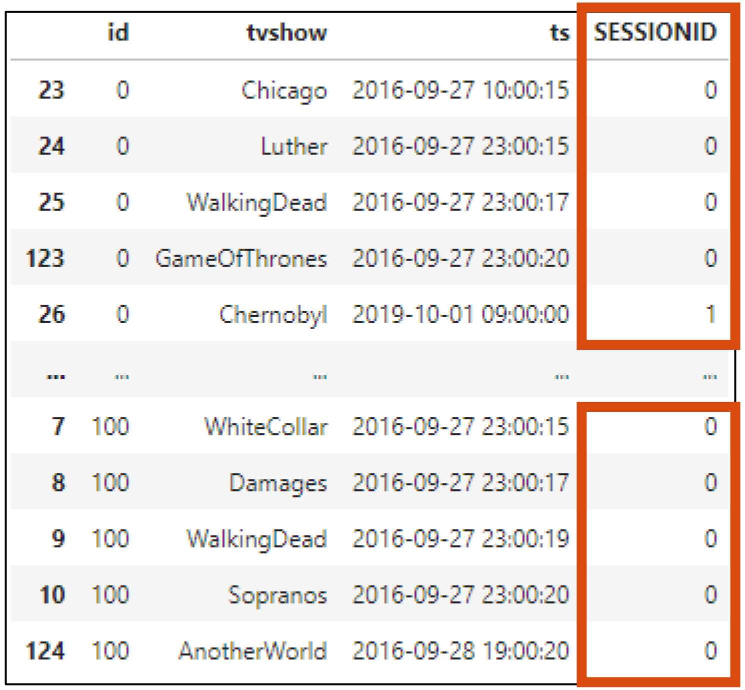


```
## Using Python, Sessionize DataFrame, Convert to Panda and Display
session_list = Sessionize(data = tv_shows_df,
                          data_partition_column=["id"],
                          data_order_column=["ts"],
                          time_column="ts",
                          time_out=86400.0)

session_pd = session_list.result.to_pandas()
session_pd.sort_values(['id','ts'], ascending=True)
```

SQL conversion



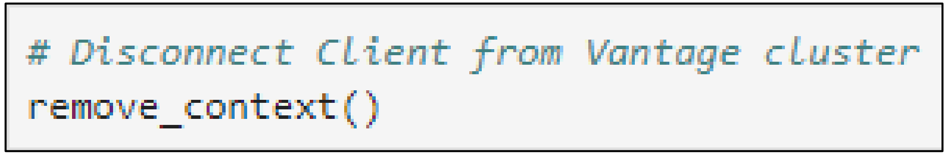
```
SELECT * FROM Sessionize(
        ON "MO130560"."ml__select__1599745719609959" AS "input"
        PARTITION BY "id"
        ORDER BY "ts"
        USING
        TimeColumn('ts')
        TimeOut(86400.0)
) as sqlmr
```



	id	tvshow	ts	SESSIONID
23	0	Chicago	2016-09-27 10:00:15	0
24	0	Luther	2016-09-27 23:00:15	0
25	0	WalkingDead	2016-09-27 23:00:17	0
123	0	GameOfThrones	2016-09-27 23:00:20	0
26	0	Chernobyl	2019-10-01 09:00:00	1
...	...	...	...	...
7	100	WhiteCollar	2016-09-27 23:00:15	0
8	100	Damages	2016-09-27 23:00:17	0
9	100	WalkingDead	2016-09-27 23:00:19	0
10	100	Sopranos	2016-09-27 23:00:20	0
124	100	AnotherWorld	2016-09-28 19:00:20	0

Lab 5b: Remove Context

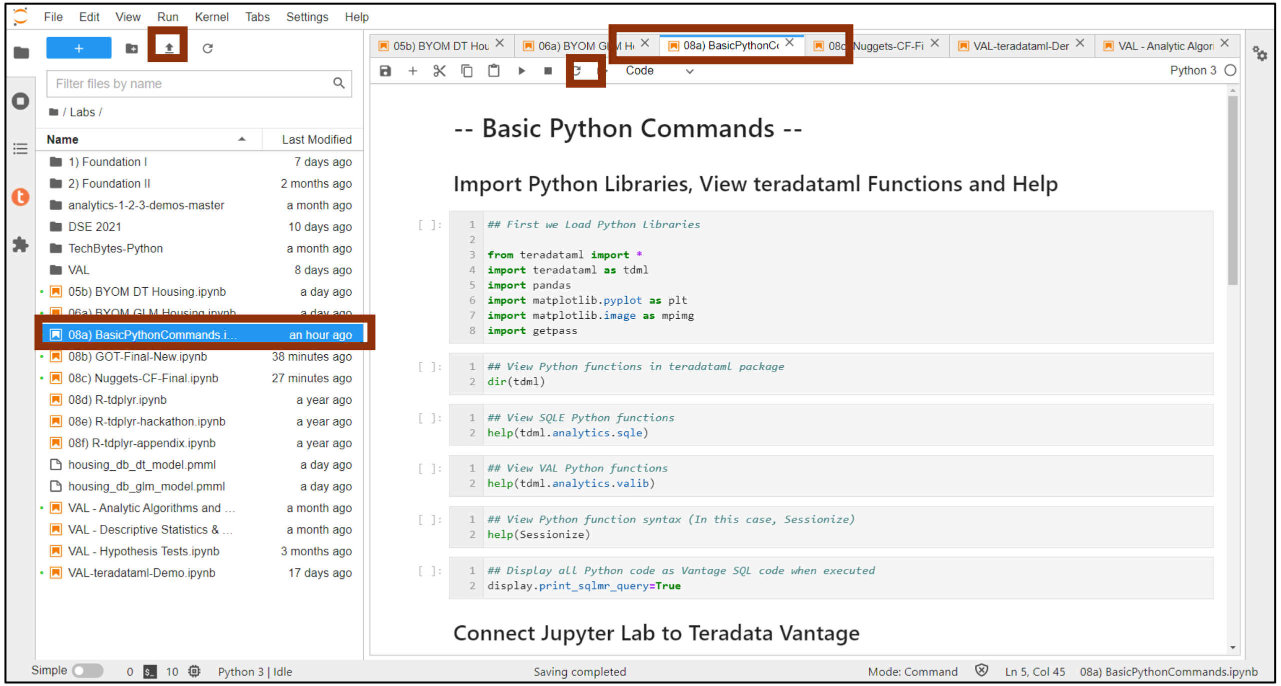
This command also does any Garbage Collection that is needed (removes Temporary tables)



```
# Disconnect Client from Vantage cluster
remove_context()
```

Lab 6: R, Open Jupyter Lab

Click the Upload button to upload the Basic R Commands Lab script



/ Labs /

Name	Last Modified
1) Foundation I	7 days ago
2) Foundation II	2 months ago
analytics-1-2-3-demos-master	a month ago
DSE 2021	10 days ago
TechBytes-Python	a month ago
VAL	8 days ago
05b) BYOM DT Housing.ipynb	a day ago
06a) BYOM GLM Housing.ipynb	a day ago
08a) BasicPythonCommands.i...	an hour ago
08b) GOT-Final-New.ipynb	38 minutes ago
08c) Nuggets-CF-Final.ipynb	27 minutes ago
08d) R-tdplyr.ipynb	a year ago
08e) R-tdplyr-hackathon.ipynb	a year ago
08f) R-tdplyr-appendix.ipynb	a year ago
housing_db_dt_model.pmml	a day ago
housing_db_glm_model.pmml	a day ago
VAL - Analytic Algorithms and ...	a month ago
VAL - Descriptive Statistics & ...	a month ago
VAL - Hypothesis Tests.ipynb	3 months ago
VAL-teradataml-Demo.ipynb	17 days ago

-- Basic Python Commands --

Import Python Libraries, View teradataml Functions and Help

```
[ ]: ## First we Load Python Libraries

     from teradataml import *
     import teradataml as tdml
     import pandas
     import matplotlib.pyplot as plt
     import matplotlib.image as mpimg
     import getpass
```

```
[ ]: ## View Python functions in teradataml package
     dir(tdml)
```

```
[ ]: ## View SQLE Python functions
     help(tdml.analytics.sqle)
```

```
[ ]: ## View VAL Python functions
     help(tdml.analytics.valib)
```

```
[ ]: ## View Python function syntax (In this case, Sessionize)
     help(Sessionize)
```

```
[ ]: ## Display all Python code as Vantage SQL code when executed
     display.print_sqlmr_query=True
```

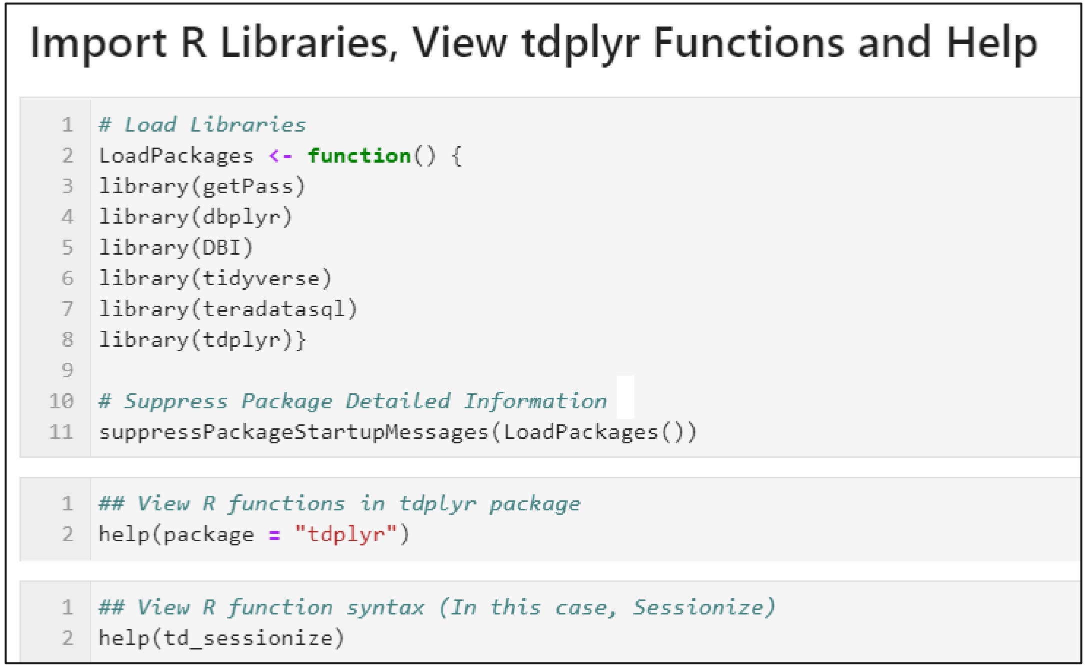
Connect Jupyter Lab to Teradata Vantage

Double clicking the Python script will open it as a tab in the main work area

Use the RUN button to execute the script statements

Lab 7a: Import R Libraries

1. Highlight Cell 1 (you'll get a blue vertical bar for that Cell)
2. Click Run button ▶. Kernel indicator circle will fill in. When finished it will be White again (sometimes it happens so fast won't see circle fill in)



Import R Libraries, View tdplyr Functions and Help

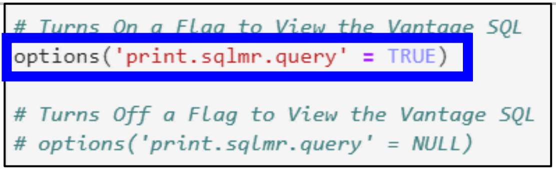
```
# Load Libraries
LoadPackages <- function() {
library(getPass)
library(dbplyr)
library(DBI)
library(tidyverse)
library(teradatasql)
library(tdplyr)}

# Suppress Package Detailed Information
suppressPackageStartupMessages(LoadPackages())
```

```
## View R functions in tdplyr package
help(package = "tdplyr")
```

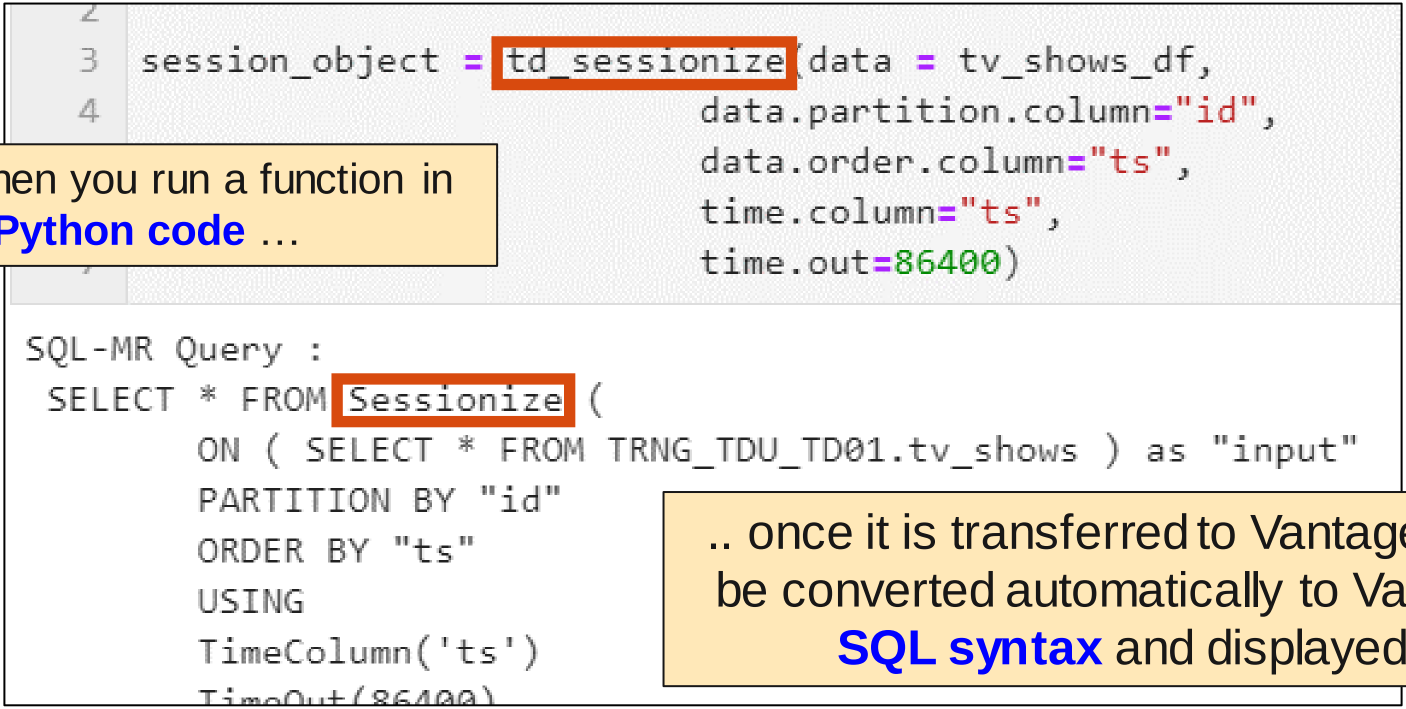
```
## View R function syntax (In this case, Sessionize)
help(td_sessionize)
```

teradata.

Lab 7b: View Underlying Vantage SQL Query



```
# Turns On a Flag to View the Vantage SQL
options('print.sqlmr.query' = TRUE)

# Turns Off a Flag to View the Vantage SQL
# options('print.sqlmr.query' = NULL)
```



```
session_object = td_sessionize(data = tv_shows_df,
                                data.partition.column="id",
                                data.order.column="ts",
                                time.column="ts",
                                time.out=86400)
```

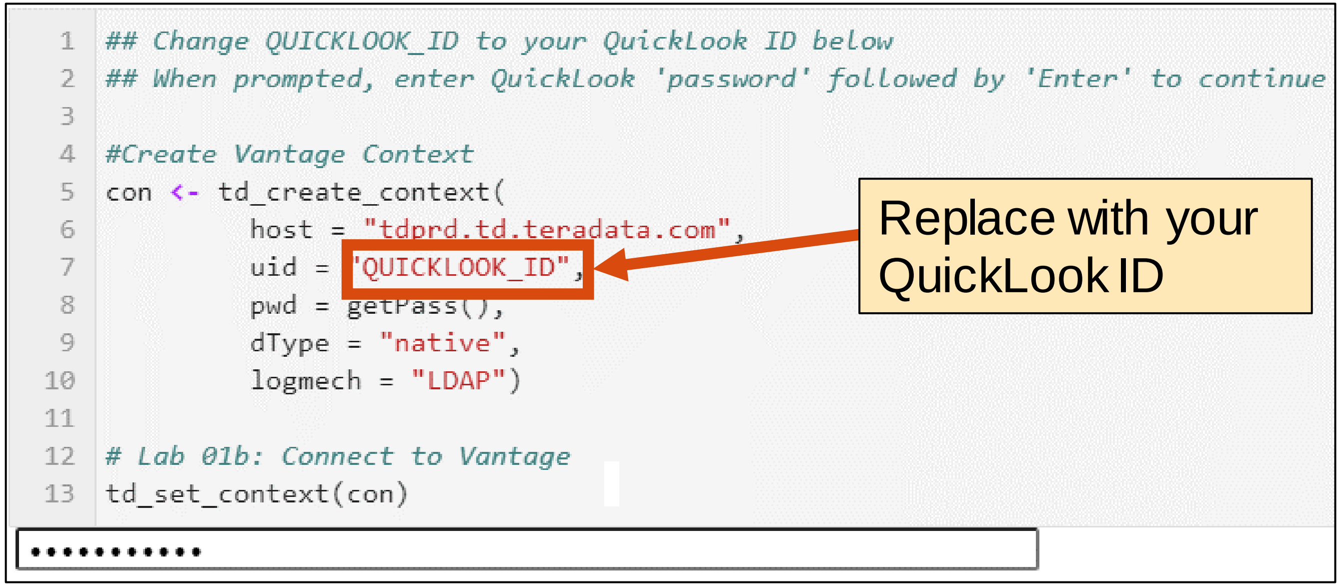
```
SQL-MR Query :
 SELECT * FROM Sessionize (
       ON ( SELECT * FROM TRNG_TDU_TD01.tv_shows ) as "input"
       PARTITION BY "id"
       ORDER BY "ts"
       USING
       TimeColumn('ts')
       TimeOut(86400)
```

Later, when you run a function in **Python code** …

.. once it is transferred to Vantage it will be converted automatically to Vantage **SQL syntax** and displayed

Lab 8: Connect JupyterLab to Vantage

Run 'td_create_context' method to connect R client to Vantage Cluster



```
## Change QUICKLOOK_ID to your QuickLook ID below
## When prompted, enter QuickLook 'password' followed by 'Enter' to continue

#Create Vantage Context
con <- td_create_context(
         host = "tdprd.td.teradata.com",
         uid = "QUICKLOOK_ID",
         pwd = getPass(),
         dType = "native",
         logmech = "LDAP")

# Lab 01b: Connect to Vantage
td_set_context(con)
```

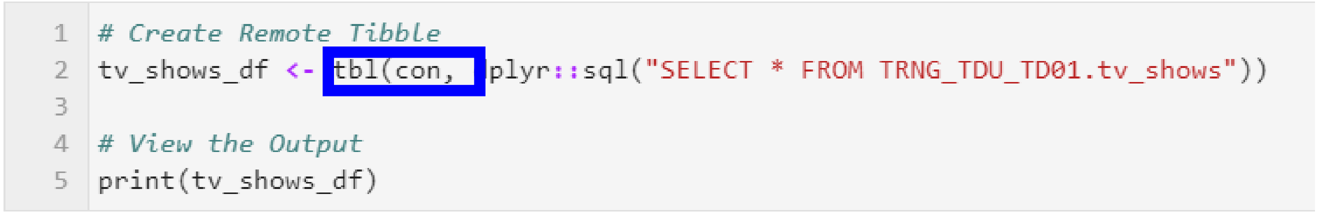
```
••••••••••••
```

Replace with your QuickLook ID

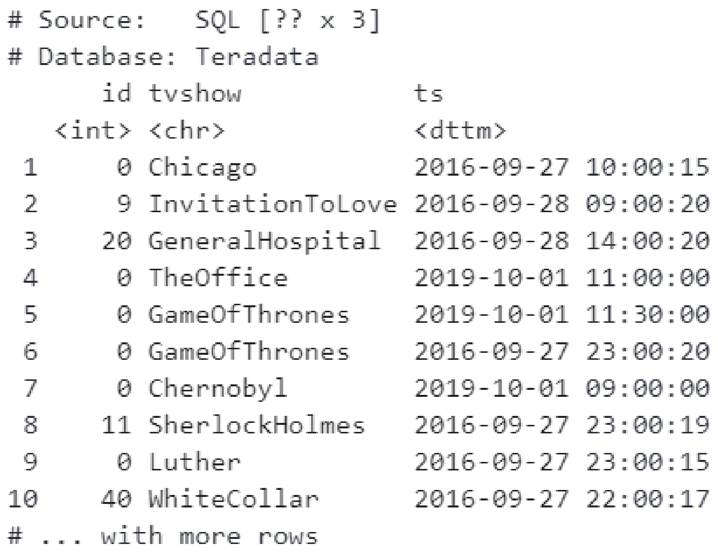
Must enter your LDAP 'password' followed by the Enter key to proceed

Lab 9: Load Table into Dataframe and Display

Use 'tbl' command to load SQL Table into a Tibble and display



```r
# Create Remote Tibble
tv_shows_df <- tbl(con, dplyr::sql("SELECT * FROM TRNG_TDU_TD01.tv_shows"))

# View the Output
print(tv_shows_df)
```

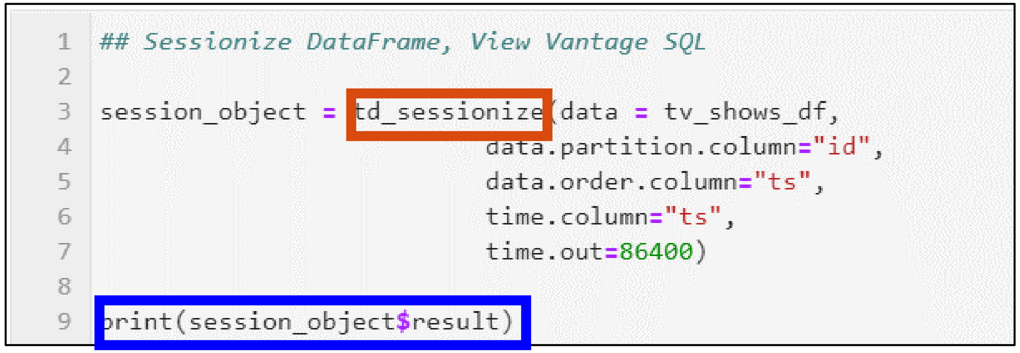


```
# Source:   SQL [?? x 3]
# Database: Teradata
      id tvshow            ts                 
   <int> <chr>             <dttm>             
 1     0 Chicago           2016-09-27 10:00:15
 2     9 InvitationToLove  2016-09-28 09:00:20
 3    20 GeneralHospital   2016-09-28 14:00:20
 4     0 TheOffice         2019-10-01 11:00:00
 5     0 GameOfThrones     2019-10-01 11:30:00
 6     0 GameOfThrones     2016-09-27 23:00:20
 7     0 Chernobyl         2019-10-01 09:00:00
 8    11 SherlockHolmes    2016-09-27 23:00:19
 9     0 Luther            2016-09-27 23:00:15
10    40 WhiteCollar       2016-09-27 22:00:17
# ... with more rows
```

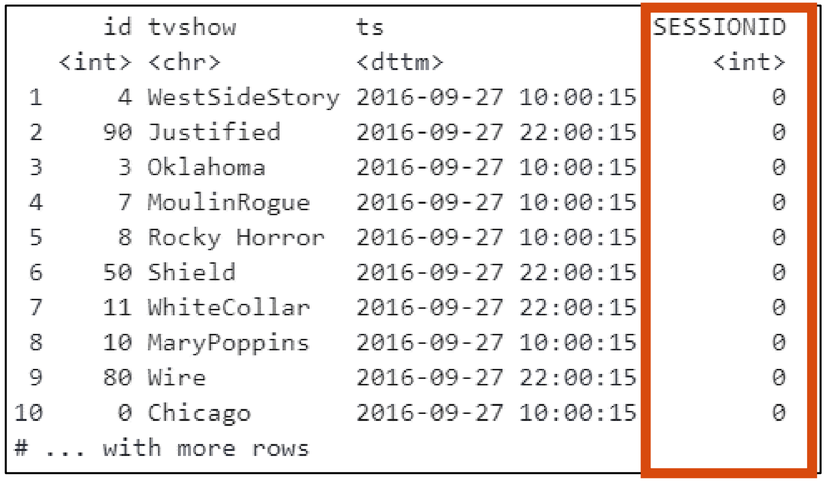
Here's 10 rows of the Tibble

Lab 10a: Sessionize the DataFrame

Note use of the '$result'. This converts the object into a tibble.

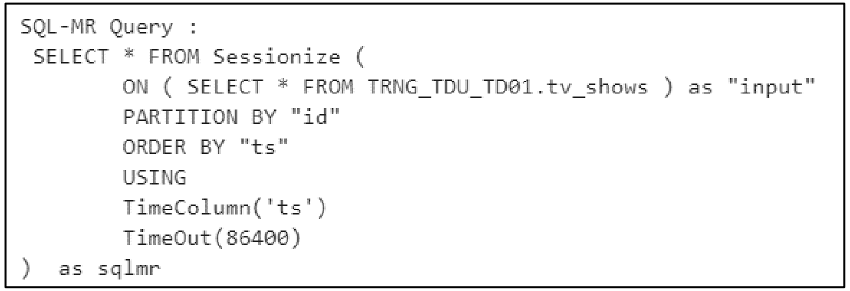


```
## Sessionize DataFrame, View Vantage SQL

session_object = td_sessionize(data = tv_shows_df,
                               data.partition.column="id",
                               data.order.column="ts",
                               time.column="ts",
                               time.out=86400)

print(session_object$result)
```



```
     id tvshow        ts                  SESSIONID
  <int> <chr>         <dttm>                  <int>
 1    4 WestSideStory 2016-09-27 10:00:15         0
 2   90 Justified     2016-09-27 22:00:15         0
 3    3 Oklahoma      2016-09-27 10:00:15         0
 4    7 MoulinRogue   2016-09-27 10:00:15         0
 5    8 Rocky Horror  2016-09-27 10:00:15         0
 6   50 Shield        2016-09-27 22:00:15         0
 7   11 WhiteCollar   2016-09-27 22:00:15         0
 8   10 MaryPoppins   2016-09-27 10:00:15         0
 9   80 Wire          2016-09-27 22:00:15         0
10    0 Chicago       2016-09-27 10:00:15         0
# ... with more rows
```

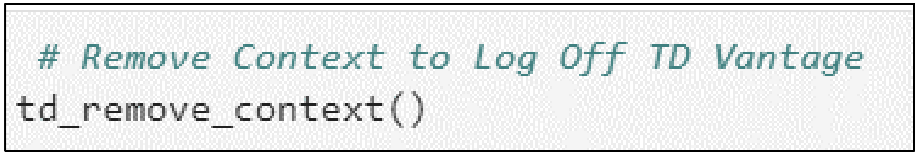
SQL conversion



```
SQL-MR Query :
 SELECT * FROM Sessionize (
        ON ( SELECT * FROM TRNG_TDU_TD01.tv_shows ) as "input"
        PARTITION BY "id"
        ORDER BY "ts"
        USING
        TimeColumn('ts')
        TimeOut(86400)
)  as sqlmr
```

Lab 10b: Remove Context

This command also does any Garbage Collection that is needed (removes Temporary tables)



```
 # Remove Context to Log Off TD Vantage
td_remove_context()
```

Summary

In this module, you learned how to:

- Describe different ways to run R and Python scripts
- Explain how to use R and Python on Vantage nodes
- Explain how to use R and Python on client systems
- Recognize interfaces that could be use with Vantage

Thank you.

teradata.