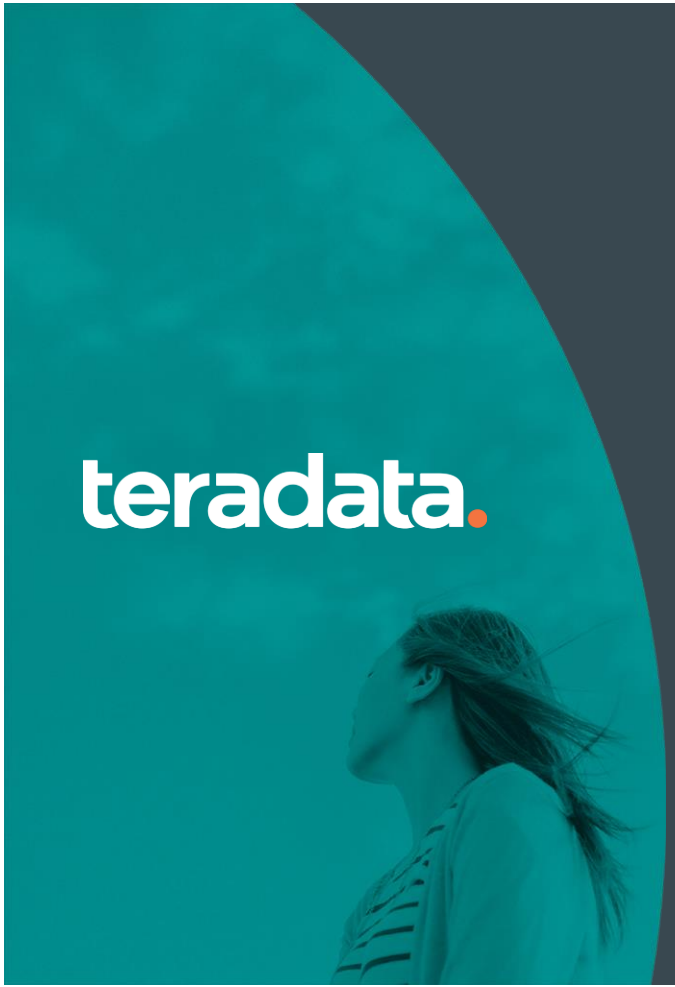


Module 4: R & tdplyr Labs

Teradata Vantage Analytics Workshop
ADVANCED

Objectives

After completing this module, you will be able to:

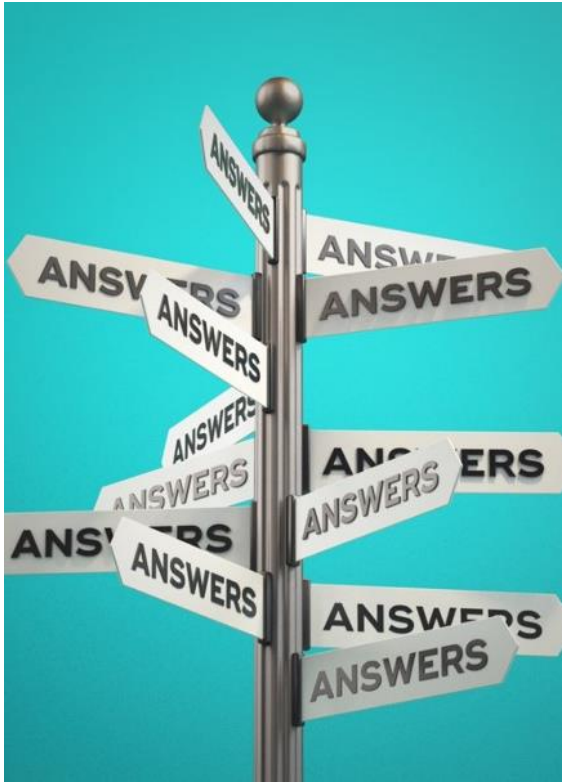
- Articulate how R and Jupyter Lab fit within the Teradata Vantage architecture
- Connect to a Teradata Vantage context from Jupyter Lab
- Create remote tibbles from Teradata Vantage tables
- Transform a data set with NGramSplitter
- Find patterns with nPath
- Find Associations with VAL

Topics

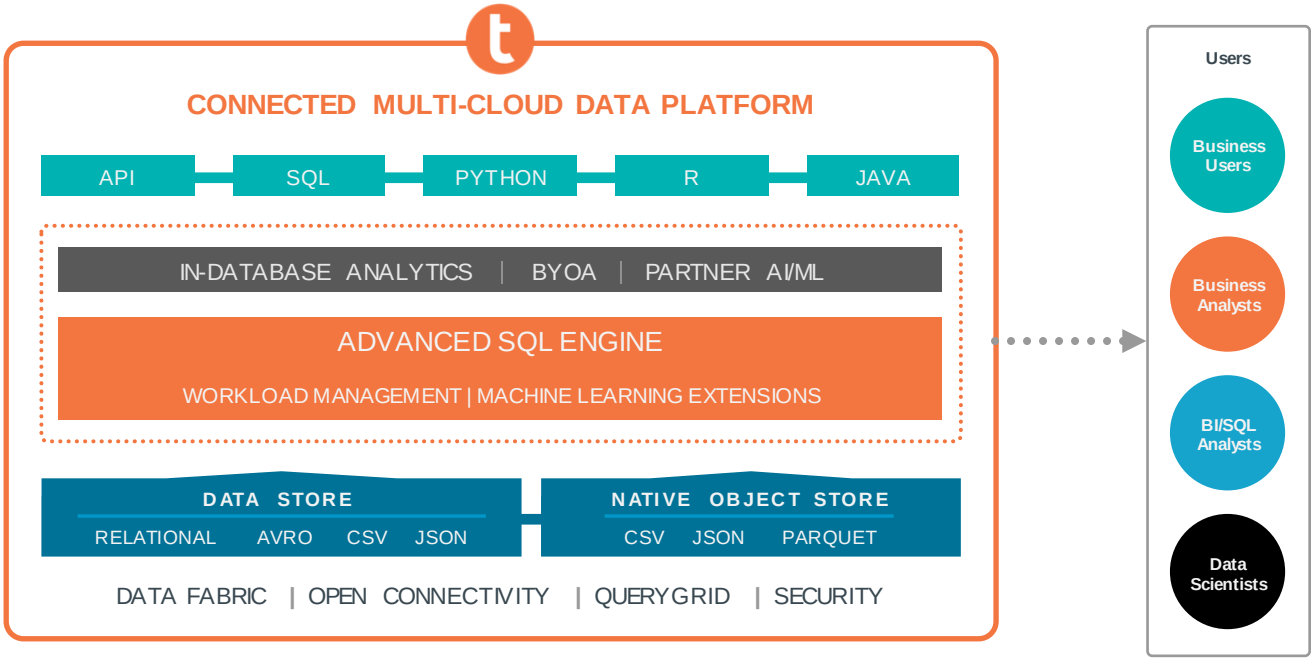
- Using R with JupyterLab via Teradata Vantage
  - Teradata Vantage Architecture
  - Connect to Teradata Vantage from JupyterLab
  - What is a Tibble?
- Text Functions
  - NGramSplitter
- Path & Pattern Analysis
  - nPath
- Association with VAL
- Review & Summary

Current Topic – Using R with JupyterLab via Teradata Vantage

- **Using R with JupyterLab via Teradata Vantage**
  - **Teradata Vantage Architecture**
  - **Connect to Teradata Vantage from JupyterLab**
  - **What is a Tibble?**
- Text Functions
  - NGramSplitter
- Path & Pattern Analysis
  - nPath
- Association with VAL
- Review & Summary



Teradata Vantage



Lab 1: Load Libraries

Load Dependent R Libraries followed by 'tdplyr'

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

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

See next pages for details on these two Libraries

Using 'dplyr' with Vantage

The Grammar of Data Manipulation

- One of the packages within the **tidyverse**
  - What is the sum of the values, grouped by product ID?
  - What are the most common car mechanical problems?
  - Which are the products with more than 10,000 reviews?
  - How do I see my data in descending order?
- Like base **SQL** and **Pandas** in **Python**

List of Helpful 'dplyr' Verbs

top_n()

Select the top *n* number of rows

mutate()

Adds new variable that are functions of existing variable

filter()

Picks cases based on their values

select()

Picks variables based on their names

summarize()

Reduces multiple values down to a single summary

arrange()

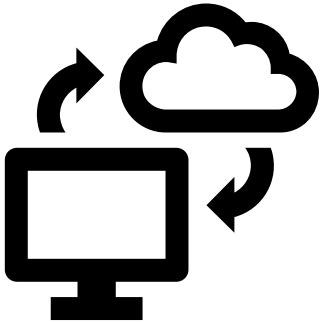
Changes the ordering of the rows

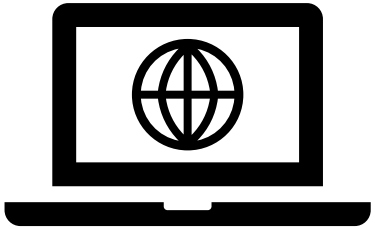
'tdplyr' Package Compared to 'dplyr' Package

- td_cfilter_mle()
- td_glm_mle()
- td_ngramsplitter()

- arrange()
- filter()
- top_n()
- summarize()
- select()
- mutate()

Create and Set Teradata Vantage Context





td_create_context

Create a Context to perform analytic functions on Teradata Vantage

td_set_context

Initialize a Context to perform analytic functions on Teradata Vantage

Lab 2: Create and Set Teradata Vantage Context


```
# Create Vantage Context
con <- td_create_context (
          host = "host_name",
          uid = "user_id",
          pwd = getpass(),
          dType = "native",
          logmech = "LDAP")

# Connect to Vantage
td_set_context(con)
```

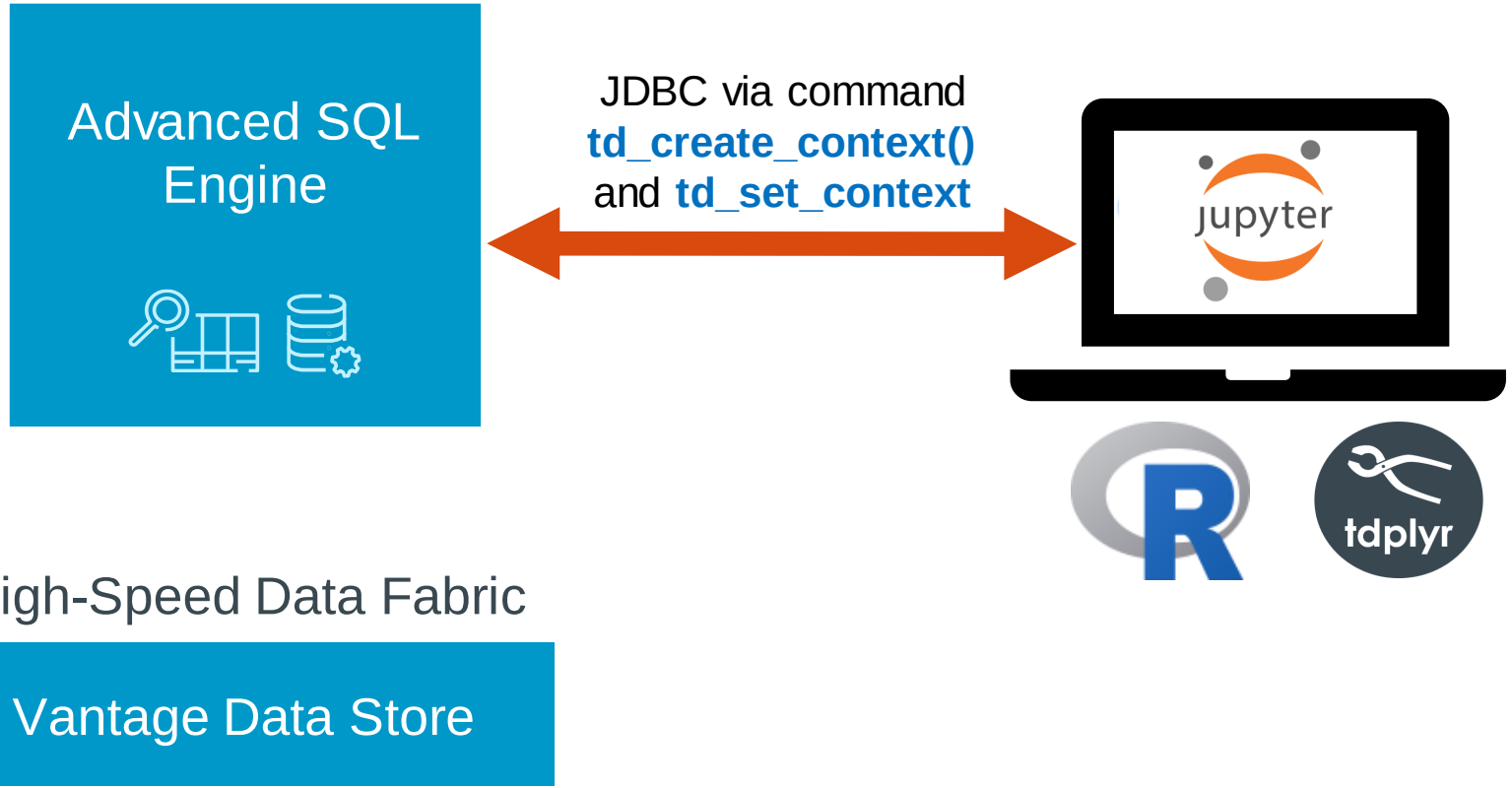
Your code may vary slightly from this Generic example

Create a variable name **con**

1. Use the **td_create_context** function
2. Input the appropriate information for the remaining arguments.

3. Input the **con** variable as the parameter using the **td_set_context** function

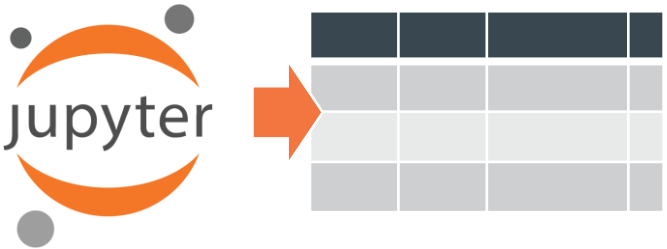
Access to Analytic Functions



Dataframes and Tables

Jupyter Lab accesses **dataframes**

- Row names should be unique
  - e.g., row numbers
- No NULLs in column names
- Data types: NUM, CHAR, factor
- Each column has one data type



Vantage stores data in **tables**

- Relational tables
- Rows are managed by AMPs, evenly distributed
- Single table can be millions of rows

EMPLOYEE

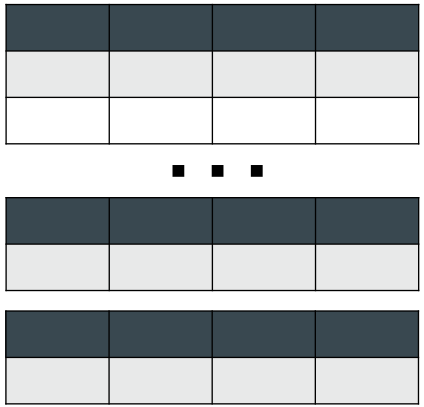
EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT NUMBER	JOB CODE	LAST NAME	FIRST NAME	HIRE DATE
1006	1019	301	312101	Stein	John	861015
1008	1019	301	312102	Kanieski	Carol	870201
1005	0801	403	431100	Ryan	Loretta	861015
1004	1003	401	412101	Johnson	Darlene	861015
1007				Villegas	Arnando	870102
1003	0801	401	411100	Trader	James	860731

Teradata Vantage

For our purposes, a Dataframe is the same as a Table

What is a Tibble in R?

Very similar to a data frame and data table, other than:



1

Provides a better method for printing large data sets

2

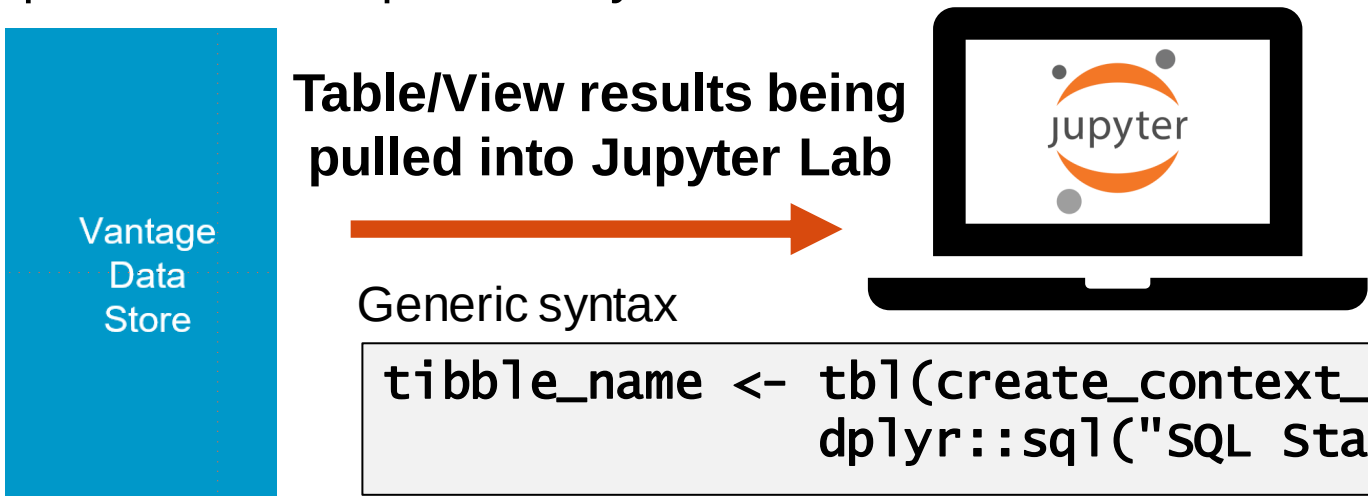
Subsetting a Tibble will always return a Tibble

3

Allows you to interact with data from remote data sources

Creating a Remote Tibble

Tibbles are our way to pull in any Teradata Vantage table into JupyterLab and perform subsequent analysis



Generic syntax

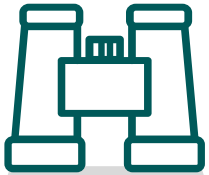
```
tibble_name <- tbl(create_context_variable,
                   dplyr::sql("SQL Statement"))
```

Example:

```
scale_housing <- tbl(con, dplyr::sql("SELECT * FROM TRNG_TDU_TD01.scale_housing"))
```

Lab 3: Converting R code to Vantage SQL


```
options('print.sqlmr.query' = TRUE)
```



SQL

Jupyter Lab

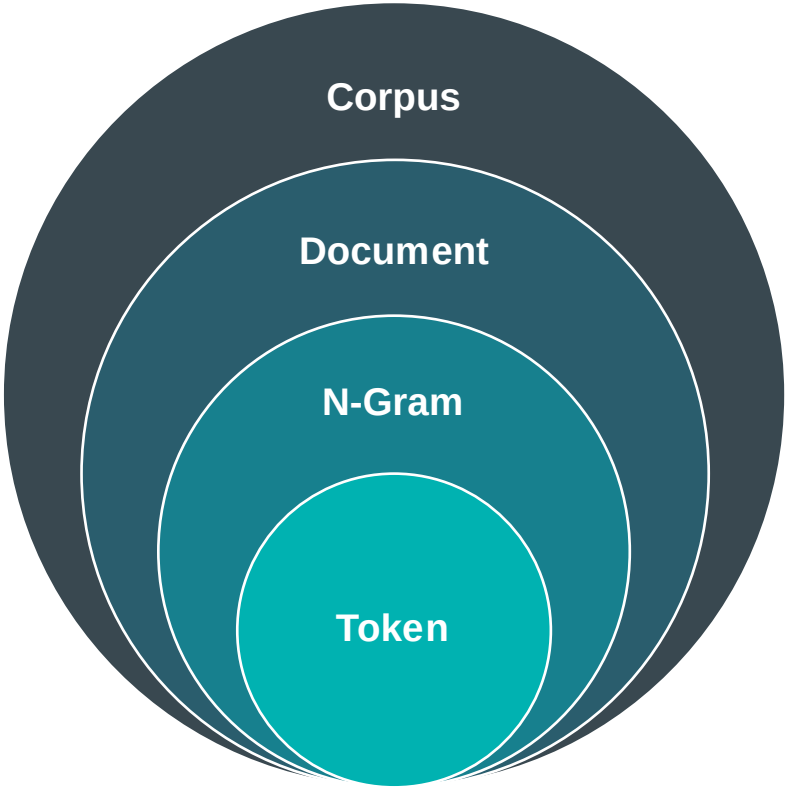
```
SQL-MR Query :
 SELECT * FROM NGramSplitter (
        ON ( SELECT * FROM TRNG_TDU_TD01.diary ) as "input"
        PARTITION BY ANY
        USING
        TextColumn('text_col')
        Grams('1')
        OverLapping('TRUE')
        Delimiter(' ')
        Punctuation('[.,?!]')
        Reset('[.,?!]')
)  as sqlmr
```

Vantage

Current Topic – Text Functions

- Using R with JupyterLab via Teradata Vantage
  - Teradata Vantage Architecture
  - Connect to Teradata Vantage from JupyterLab
  - What is a Tibble?
- **Text Functions**
  - **NGramSplitter**
- Path & Pattern Analysis
  - nPath
- Review & Summary

Some Text Analytics Terminology: A Hierarchy



Each outer layer is essentially a superset of the others:

- **Corpus:** Data source
- **Document:** Row of text data
- **N-Gram:** Group of words
- **Token:** Single word

Examples of Corpus and Document

Corpus: The column with all the reviews

MOVIE_REVIEWS

AccountID	Movie	Date	Stars	Comments
31964358	Guns and Bla	041218	4	This director is so talented, this is one of the best…
22727277	Romance in P	120617	5	My favorite part was when Oscar and Lily decided…
85643108	Italian Connec	010119	4	The character development is exquisite and the vi…
55467372	Robots Amok	071619	3	The special effects were lacking but I like the plot i…
91150045	All My Cats	012318	1	This was such a waste of time we walked out of th…

Document: One row in a column

Examples of N-Grams and Tokens

MOVIE_REVIEWS

AccountID	Movie	Date	Stars	Comments
31964358	Guns and Bla	041218	4	This director is so talented, this is one of the best
22727277	Romance in P	120617	5	My favorite part was when Oscar and Lily decided
85643108	Italian Connec	010119	4	The character development is exquisite and the vi
55467372	Robots Amok	071619	3	The special effects were lacking but I like the plot i
91150045	All My Cats	012318	1	This was such a waste of time we walked out of th

N-Grams:
Bigram
Trigram

Token: Individual word

This director is so talented this is one of the best action movies I have seen in years. I normally go for these kinds of films anyway, but it is such an unusual and unique take on this genre. The span of time from old-time Wild West to the futuristic laser gun battle is epic and especially like the way Conrad weaves in the ragtag team of misfits and how they come together to vanquish the Marquohar. The visual aspect of the effects made me feel like I was part of the action and I especially recommend seeing this in a theater with surround sound, Dolby, and the like to get the full effect and experience. Highly recommend!

Why Tokenize Text? "Noisy Data"

Misspellings

`The Iccredribles movie`

Irrelevant to the context

`Eating popcorn`

`Have a great day!`

`Throwback Thursday`

Extra Punctuation

`It was super!!!!!`

`Best. Day. Ever.`

Emojis

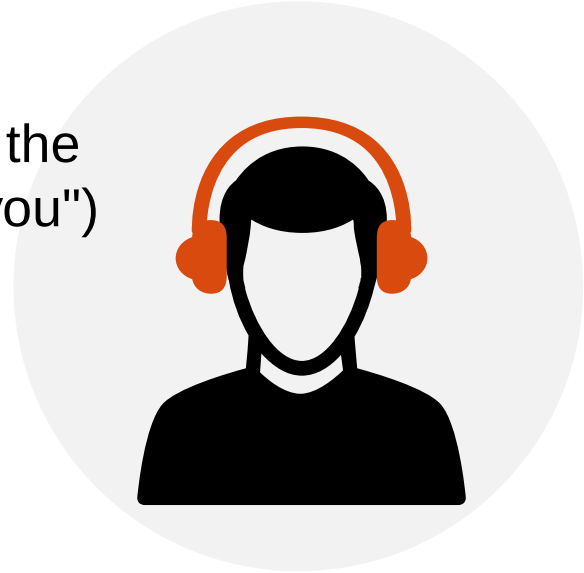
☺ `:-)`

Abbreviations

`ICYMI` `LOL` `FOMO`

Techniques for Reducing Noise

- Remove all punctuation and numbers
- Lowercase all words
- Reduce words to root words (stemming or lemmatization)
- Ignore certain words (stop words)
  - Frequent words ("the", "an", "a")
  - Irrelevant words and phrases ("Welcome to the Call Center" "How may I help you" "Thank you")
    - Custom dictionaries



Bag of Words Model

- The model characterizes data in terms of individual words
- No implied order or structure
- No context
- Produces word counts, frequencies
- Adds numerical value or weight to text
- Can analyze, rank, or calculate

Text Data Tokenizing Functions

Pre-processing functions for tokenizing text

Single-Word Tokenization

td_text_parser

More pre-processing parameters:

- Lowercase
- Remove punctuation
- **Stemming**
- **Remove stop words**

Multiple-Word Tokenization

td_ngramsplitter_sqle

Fewer pre-processing parameters:

- Lowercase
- Remove punctuation
- **More context with grams**
- **More accurate ML models**

Syntax: nGram Splitter


```
td_ngramsplitter_sqle (
        data = NULL,
        text.column = NULL,
        delimiter = " ",
        grams = NULL,
        overlapping = TRUE,
        to.lower.case = TRUE,
        punctuation = "`~#^&*()-",
        reset = ".,?!",
        total.gram.count = FALSE,
        total.count.column = "totalcnt",
        accumulate = NULL,
        n.gram.column = "ngram",
        num.grams.column = "n",
        frequency.column = "frequency",
        data.order.column = NULL
```

Overlapping:
Recommend changing to **FALSE**

Frequency Column:
Can be used to determine word count

Lab 4a: Diary Remote Tibble


```
diary <- tbl(con,dplyr::sql
          ("SELECT * FROM diary"))
```

Create a remote tibble named **diary**

1. Use the **tbl** function
2. Reference our **con** Vantage context variable
3. Use the **dplyr::sql** function to query the Vantage table

Input

	text_col
	<chr>
1	"1\tJune 12th 1941: Saw Klaus watching me again in class. I think he likes me.

Lab 4b: Tokenize with nGrams Splitter


```
tokenized_diary <- td_ngramsplitter_sqle (
      data = diary,
      text.column = "text_col",
      delimiter = " ",
      grams = "1",
      overlapping = TRUE,
      punctuation = "[.,?!]",
      reset = "[.,?!]")
```

Create an object named **text_parser_diary**

1. Use the **td_text_parser_mle** function
2. Input appropriate values for the pre-processing **parameters**
3. Input **TRUE** to remove stop words and list the position of the word within the document

Arrange with dplyr

View the output by executing the following command:

```
arrange(tokenized_diary$result, desc(frequency))
```

Arrange by the frequency column in descending order

Output

	id	text_col	ngram	n	frequency
	<int>	<chr>	<chr>	<int>	<int>
1	1	"June 12th 1941: Saw Klaus watching me again in …	i	1	10
2	1	"June 12th 1941: Saw Klaus watching me again in …	he	1	8
3	1	"June 12th 1941: Saw Klaus watching me again in …	we	1	8
4	1	"June 12th 1941: Saw Klaus watching me again in …	to	1	6
5	1	"June 12th 1941: Saw Klaus watching me again in …	that	1	5
6	1	"June 12th 1941: Saw Klaus watching me again in …	the	1	5
7	1	"June 12th 1941: Saw Klaus watching me again in …	a	1	4
8	1	"June 12th 1941: Saw Klaus watching me again in …	have	1	4
9	1	"June 12th 1941: Saw Klaus watching me again in …	says	1	4
10	1	"June 12th 1941: Saw Klaus watching me again in …	in	1	4

Current Topic – Path & Pattern Analysis

- Using R with JupyterLab via Teradata Vantage
  - Teradata Vantage Architecture
  - Connect to Teradata Vantage from JupyterLab
  - What is a Tibble?
- Text Functions
  - NGramSplitter
- **Path & Pattern Analysis**
  - **nPath**
- Review & Summary

nPath Description

- The **nPath** function scans a set of rows, looking for patterns that you specify
- For each set of input rows that matches the pattern, **nPath** produces a single output row
- The function provides a flexible pattern-matching capability that lets you specify complex patterns in the input data and define the values that are output for each matched input set
- **nPath** is useful when your goal is to identify the paths that lead to an outcome
- The output from the **nPath** function can be input into other Machine Learning Engine functions or into a visualization tool such as Teradata AppCenter

nPath Description (cont.)

What is nPath?

- Function designed for time-series sequence analysis of data
- Links an outcome with a preceding path

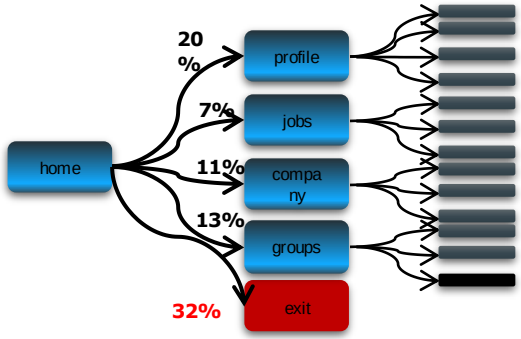
Benefits

- *Pattern detection can be completed in a single pass* over the data
- Allows you to understand relationships across rows of data
- Transcend's SQL ordered-data limitations that require either complex, multi-pass SQL or custom UDFs for each analysis

Example use cases:

- Web analytics (clickstream, Golden Path)
- Complex Marketing revenue paths
- Granular product & process analysis (A/B)
- Granular pattern detection (fraud, QA,..)

nPath Website Path Analysis



Complete Application:

- ***n*Path** identifies path patterns and exit points
- The **Sessionize** function is used prepare the input data for **nPath** analysis.
- In the example above, we are viewing which paths users take once they've landed on our Home page. Note that 32% of Home page visits end up in the users' exiting the website altogether.

nPath Use Cases

Some examples of how **nPath** can be used follow:

- A retailer wishes to analyze Web site click data, to identify paths that lead to sales over a specified amount
- A manufacturer analyzes sensor data from industrial processes, to identify paths to poor product quality
- A healthcare provider analyzes healthcare records of individual patients, to identify paths that indicate that patients are at risk of developing conditions such as heart disease or diabetes
- A financial institution reviews financial data for individuals, to identify paths that provide information about credit or fraud risks

nPath Workflow

- **Input Tibble:** Data is read from specified input tables
- **nPath:** The following arguments are specified when the function is invoked
  - **Mode (overlapping or nonoverlapping)**
  - **Pattern to match**
  - **Symbols to use**
  - **[Optional] Filters to apply**
  - **Results to output**
- **Output object:** Data is written to an output object

Lab 5: Create borre_z Remote Tibble


```
borre_z <- tbl(con,dplyr::sql
               ("SELECT * FROM borre_z"))
```

Create a remote tibble named **borre_z**

1. Use the **tbl** function
2. Reference our **con** Vantage context variable
3. Use the **dplyr::sql** function to query the Vantage table

Input

```
  user_id event ts
    <int> <chr> <dttm>
1       1 a     2017-01-01 13:21:03
2       1 a     2017-01-01 13:21:04
3       1 a     2017-01-01 13:21:02
4       1 a     2017-01-01 13:21:01
```

Syntax: nPath

- The **td_npath_sqle** function scans a set of rows, looking for patterns that you specify
- For each set of input rows that matches the pattern, **nPath** produces a single output row
- The function provides a flexible pattern-matching capability that lets you specify complex patterns in the input data and define the values that are output for each matched input set

```
td_npath_sqle (
      data1 = NULL,
       mode = NULL,
       pattern = NULL,
       symbols = NULL,
       result = NULL,
       filter = NULL,
       data2 = NULL,
       data3 = NULL,
       data1.partition.column = NULL,
       data2.partition.column = NULL,
       data3.partition.column = NULL,
       data1.order.column = NULL,
       data2.order.column = NULL,
       data3.order.column = NULL)
```

Pattern Operators

- Use with pattern symbols to customize pattern-matching rules
  - '**.**' : followed by (Use to separate a series of pattern symbols)
  - '**|**' : alternative (The equivalent of an OR)
  - '**?**' : occurs at most once (0-1)
  - '*****' : occurs zero or more times (0-n)
  - '**+**' : occurs at least once (1-n)
  - '**^**' : pattern must begin with value specified. Also, value specified must be the first row within the partition.
  - '**$**' : pattern must end with

- Customizing pattern matching rules:

(X){a}: exactly A number of occurrences of X	**pattern('A.B{3}')**
(X){a,}: at least A number of occurrences of X	**pattern('A.B{1,}')**
(X){a,b}: A to B occurrences of X	**pattern('A.B{1,3}')**

Pattern Operators (cont.)

Operator	Description	Precedence
A	Matches one row that meets the definition of A	1 (highest)
A.	Matches one row that meets the definition of A	1
A?	Matches 0 or 1 rows that satisfy the definition of A	1
*A**	Matches 0 or more rows that satisfy the definition of A (greedy operator)	1
A+	Matches 1 of more rows that satisfy the definition of A (greedy operator)	1
A.B	Matches two rows, where the first row meets the definition of A and the second row meets the definition of B	2
A|B	Matches one row that meets the definition of either A or B	3

The **nPath** function uses greedy pattern matching. That is, it finds the longest available match when matching patterns specified by nongreedy operators

Simple nPath Example


```
npath_out <- td_npath_sqle (
     data1 = borre_z,
     data1.partition.column = c("user_id"),
     data1.order.column = "ts",
     mode = "nonoverlapping",
     pattern = "X.X",
     symbols = "event = 'a' as X",
     result = "ACCUMULATE (event of X) AS x_pattern")
```

Input

	user_id	event	ts
	<int>	<chr>	<dttm>
1	1	a	2017-01-01
2	1	a	2017-01-01
3	1	a	2017-01-01
4	1	a	2017-01-01

Create an object named **borre_z**

1. Use the **tbl_npath_sqle** function
2. Reference our **borre_z** remote tibble
3. Select **user_id** as the partition column
4. Order by the **ts** column
5. Input the remaining required arguments

Output

	x_pattern
	<chr>
1	[a, a]
2	[a, a]

Lab 6a: nPath Between Remote Tibble


```
npath_between <- tbl(con,dplyr::sql
                     ("SELECT * FROM npathBetween2"))
```

Create a remote tibble named **npath_between**

1. Use the **tbl** function
2. Reference our **con** Vantage context variable
3. Use the **dplyr::sql** function to query the Vantage table

Input

	c1	c2	c3
	<int>	<int>	<chr>
1	1	1	B
2	3	1	C
3	2	1	B
4	5	1	D
5	4	1	A

Lab 6b: nPath 'Or' Pattern


```
npath_or_out <- td_npath_sqle (
     data1 = npath_between,
     data1.partition.column = "c1",
     data1.order-column = "c1",
     mode = "nonoverlapping",
     pattern = "B|C|A",
     symbols = c("c3 = 'A' as A", "c3 = 'B' as B",
                 "c3 = 'C' as C")
     results = "ACCUMULATE(c3 of ANY(B,C,A))
               AS matches")
```

Lab 6c: View the Output

View the output by executing the following command:

```
print(npath_or_out)
```

Input

	c1	c2	c3
	<int>	<int>	<chr>
1	1	1	B
2	3	1	C
3	2	1	B
4	5	1	D
5	4	1	A

Output

	matches
	<chr>
1	[B]
2	[C]
3	[B]
4	[A]

Persist Dataframe as a Vantage Table

To more easily create a visualization in **Teradata AppCenter** we can **copy** our Dataframe into a **Vantage table**

```
copy_to (con,npath_or_out$result,name = "npath_or_out")
```

Current Topic – Association with VAL

- Using R with JupyterLab via Teradata Vantage
  - Teradata Vantage Architecture
  - Connect to Teradata Vantage from JupyterLab
  - What is a Tibble?
- Text Functions
  - NGramSplitter
- Path & Pattern Analysis
  - nPath
- **Association with VAL**
- Review & Summary

Lab: Load Table into Dataframe and Display

Use 'DataFrame' command to load SQL Table into a DataFrame, then display

```
1  # Create Remote Tibble
2  sales_detail1_df <- tbl(con, dplyr::sql("SELECT * FROM TRNG_TDU_TD01.sales_detail1"))
3
4  # View the Output
5  print(sales_detail1_df)
```

```
# Source:   SQL [?? x 11]
# Database: Teradata
   product_name     product_category_name store_name region_name city_name
   <chr>            <chr>                <chr>      <chr>       <chr>
 1 Bluberries       Fruits               Denver     Western     Denver
 2 Chicken Nuggets  Other Snacks         Denver     Western     Denver
 3 Slurpee          Drinks               Denver     Western     Denver
 4 Sprite           Drinks               Denver     Western     Denver
 5 Toaster pastries Ethnic Snacks        Denver     Western     Denver
 6 Tuna Snacks      Other Snacks         Denver     Western     Denver
 7 Chocolate Bars   Candy                San Diego  Western     San Diego
 8 Taffy            Candy                Denver     Western     Denver
 9 Chicken Nuggets  Other Snacks         Denver     Western     Denver
10 Red Bull         Drinks               Chicago    Eastern     Chicago
# ... with more rows, and 6 more variables: sales_date <dttm>,
#   customer_id <int>, basket_id <int>, store_id <int>, sales_quantity <int>,
#   discount_amount <dbl>
```

Lab 15: Association on the Data

First, we run Association against all the Data

```
## Using R code, Association all Products and Display
obj <- td_association_valib(data=sales_detail1_df, group.column="basket_id",
                            item.column="product_name")
```

ITEM1OF2	ITEM2OF2	LSUPPORT	RSUPPORT	SUPPORT	CONFIDENCE	LIFT	ZSCORE
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Cookies	Corn chips	0.0163	0.0118	0.00148	0.0909	7.68	2.41
Nerds	French Fries	0.0163	0.0178	0.00148	0.0909	5.12	1.82
French Fries	Nerds	0.0178	0.0163	0.00148	0.0833	5.12	1.82
Mike and Ikes	Smores	0.0222	0.0118	0.00148	0.0667	5.63	1.95
Muddy buddies	Bagel chips	0.0222	0.0222	0.00148	0.0667	3.00	1.16
Bagel chips	Muddy buddies	0.0222	0.0222	0.00148	0.0667	3.00	1.16
Soused herring	Cola	0.0133	0.0178	0.00148	0.111	6.26	2.10
Smores	Mike and Ikes	0.0118	0.0222	0.00148	0.125	5.63	1.95
Corn chips	Cookies	0.0118	0.0163	0.00148	0.125	7.68	2.41
Fairy bread	Pretzels	0.0251	0.0207	0.00296	0.118	5.68	2.78

Lab 15: Association on the Data (cont.)

Sort by LIFT

```
1  # Sort by LIFT decending
2  assoc <- arrange((obj$result.11), desc(LIFT))
```

```
1  print(assoc)
```

```
# Source:
#   table<"TRNG_TDU_TD01"."r__t__valib_td_association_valib1632772754800163_11">
#   [?? x 8]
# Database:   Teradata
# Ordered by: desc(LIFT)
   ITEM1OF2      ITEM2OF2      LSUPPORT RSUPPORT SUPPORT CONFIDENCE  LIFT ZSCORE
   <chr>         <chr>            <dbl>    <dbl>   <dbl>      <dbl> <dbl>  <dbl>
 1 Cheese curls  Gatorade       0.00592  0.00740 0.00148      0.25   33.8   5.64
 2 Gatorade      Cheese curls   0.00740  0.00592 0.00148      0.2    33.8   5.64
 3 Cheese curls  Chocolate Tr~  0.00592  0.00888 0.00148      0.25   28.2   5.12
 4 Chocolate Tr~ Cheese curls   0.00888  0.00592 0.00148      0.167  28.2   5.12
 5 Deep-fried T~ Cheese puffs   0.00592  0.0104  0.00148      0.25   24.1   4.71
 6 Cheese puffs  Deep-fried T~  0.0104   0.00592 0.00148      0.143  24.1   4.71
 7 Chocolate Tr~ Gatorade       0.00888  0.00740 0.00148      0.167  22.5   4.54
 8 Gatorade      Chocolate Tr~  0.00740  0.00888 0.00148      0.2    22.5   4.54
 9 Meze          Cheese curls   0.0118   0.00592 0.00148      0.125  21.1   4.38
10 Cheese curls  Meze           0.00592  0.0118  0.00148      0.25   21.1   4.38
```

Lab 15: Association on the Data (cont.)

Select only Chicken Nuggets in first column

```
1 arrange((obj$result.11), desc(LIFT))%>%filter(ITEM1OF2 == "Chicken Nuggets")
```

```
# Source:     lazy query [?? x 8]
# Database:   Teradata
# Ordered by: desc(LIFT)
   ITEM1OF2        ITEM2OF2    LSUPPORT RSUPPORT SUPPORT CONFIDENCE  LIFT ZSCORE
   <chr>           <chr>          <dbl>    <dbl>   <dbl>      <dbl> <dbl>  <dbl>
 1 Chicken Nuggets Almonds       0.0163  0.00888 0.00148     0.0909 10.2    2.89
 2 Chicken Nuggets Energy bars   0.0163  0.0104  0.00148     0.0909  8.78   2.63
 3 Chicken Nuggets Taffy         0.0163  0.0133  0.00148     0.0909  6.83   2.23
 4 Chicken Nuggets Cup noodles   0.0163  0.0133  0.00148     0.0909  6.83   2.23
 5 Chicken Nuggets Sprite        0.0163  0.0148  0.00148     0.0909  6.15   2.08
 6 Chicken Nuggets Jaffa cake    0.0163  0.0163  0.00148     0.0909  5.59   1.94
 7 Chicken Nuggets Peanuts       0.0163  0.0163  0.00148     0.0909  5.59   1.94
 8 Chicken Nuggets Toaster pa~   0.0163  0.0178  0.00148     0.0909  5.12   1.82
 9 Chicken Nuggets Cola          0.0163  0.0178  0.00148     0.0909  5.12   1.82
10 Chicken Nuggets Gummy Bears   0.0163  0.0207  0.00148     0.0909  4.39   1.62
```

Lab 16: Remove Context

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

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

Current Topic – Review & Summary

- Using R with JupyterLab via Teradata Vantage
  - Teradata Vantage Architecture
  - Connect to Teradata Vantage from JupyterLab
  - What is a Tibble?
- Text Functions
  - NGramSplitter
- Path & Pattern Analysis
  - nPath
- Association with VAL
- **Review & Summary**

Review & Summary

In this module, you learned how to:

- Articulate how R and Jupyter Lab fit within the Teradata Vantage architecture
- Connect to a Teradata Vantage context from Jupyter Lab
- Create remote tibbles from Teradata Vantage tables
- Transform a data set with NGramSplitter
- Find patterns with nPath
- Find Associations with VAL

Thank you.

teradata.