DATA VIRTUALITY MASTERCLASS

Topic: Sending formatted reports from Data Virtuality

Welcome to the DV Masterclass! Agenda

Day 1

2:00pm - 3:00pm: Analytical storage considerations

3:00pm - 3:15pm: Break

3:15pm - 4:00pm: Support stories

4:00pm - 4:15pm: Break

4:15pm - 5:00pm: Logging with log4j

Day 2

2:00pm - 3:00pm: Sending formatted reports from Data Virtuality

3:00pm - 3:15pm: Break

3:15pm - 4:00pm: Updating Data Virtuality Server

4:00pm - 4:15pm: Break

4:15pm - 4:45pm: Receiving webhooks

What to expect from this session?

In this track, we will discuss the options of analytical storages for Data Virtuality.

- Emailing data as a CSV attachment (not covered in the video but code is included)
- Emailing data as an inline HTML table
- Generating HTML reports using Python

Demo: Emailing data as a CSV attachment

Emailing data as a CSV attachment

Emailing files as attachments is very simple. The things to remember are to create the ARRAYs and cast the body to a CLOB data type.

- Below is an example

```
call "UTILS.sendMail"(
    "Recipients" => 'datavirtuality@yopmail.com',
    "Subject" => subject,
    "Body" => cast(email_body as clob),
         "Attachments" => array_files,
          "AttachmentNames" => array_file_names,
         "AttachmentMimeTypes" => array_mime_types
)
```

Demo: Emailing data as an inline HTML table

Emailing data as an inline HTML table

At this time, Data Virtuality doesn't include a sproc to convert a table. For this session, we created a sproc that takes a table or view name and creates an HTML table using the data. The sproc relies heavily on dynamic SQL.

- The sproc is called x_utils.tableToHtml_v2
- The variable 'qry' stores the dynamic SQL. This command

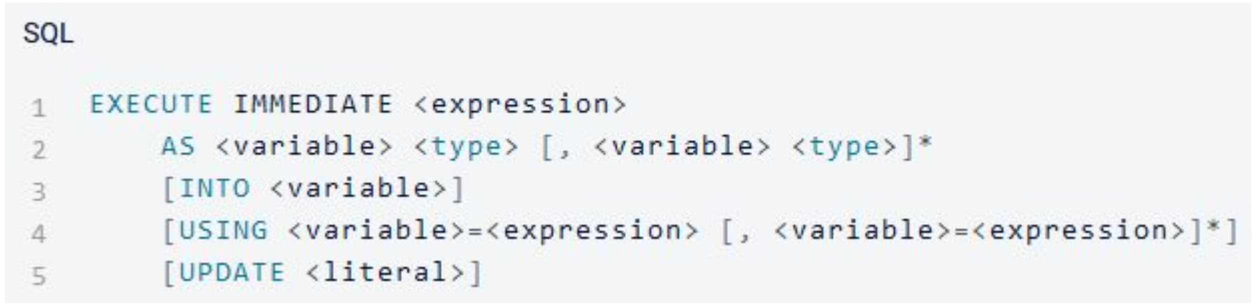
  execute IMMEDIATE qry as body xml into #hdr;

  is used to execute the dynamic SQL and store the results into a temp table.

Emailing data as an inline HTML table

Let’s dissect the command

execute IMMEDIATE qry as body xml into #hdr;



SQL

```
EXECUTE IMMEDIATE <expression>
    AS <variable> <type> [, <variable> <type>]*
    [INTO <variable>]
    [USING <variable>=<expression> [, <variable>=<expression>]*]
    [UPDATE <literal>]
```

qry is a variable with the dynamic SQL

body xml is the definition of the result set

into #hdr instructs the command to store the results in the temp table call **hdr**

data virtuality

Demo: Generating HTML reports using Python

Generating HTML reports using Python

To connect to Data Virtuality we'll be using the psycopg library and treating Data Virtuality as if it were a PostgreSQL database.

This example uses Jinja to render the HTML. Although any suitable rendering engine can be used.

Using a rendering engine requires a slightly different workflow. In our previous examples, the code loop was responsible for acquiring the data and creating all the HTML.

With Jinja, the code loops for placing the data is inside of the HTML template and is executed by the rendering engine.

The Python code is more of an orchestrator. Python connects to the database and retrieves the data. Then reshapes the data if needed. The data is then passed to the rendering engine. The rendering engine actually builds the HTML.

Summary

- Sending reports from within Data Virtuality
- Other reporting options

Any feedback / questions?

Thank you!

Please feel free to contact us at:
presales@datavirtuality.com

or

visit us at:
datavirtuality.com