ANALYTICS AND VISUALIZATION DESIGN DOCUMENT (DRAFT V 1.1)

By Akash Shah & Aakansha Rathore.

TABLE OF CONTENTS

CHANGE LOG

Author	Item	Date	Comments
Akash Shah	Initial Design	28/2/2019	Unreleased
Akash Shah	Addition of the Visualization Layer Decision	2/3/2019	Unreleased
Akash Shah	Added change log, table of contents and versioning for the document	6/3/2019	Released v1.0
Akash Shah	Design for Thumbnail Definition and added Analytics Queries in Appendix	12/3/2019	Released v1.1

OVERVIEW

The vision is to have KPI's dynamically calculated and visualized, instead of hardcoding deployment of views and tables in the Analytics DB.

REQUIREMENTS

- BigQuery KPI Definition Table
- BigQuery Visualization Definition Table
- Analytics Processing Python Program: **gpb_ds_bq_unk_analytics_load.py**
- Visualization Processing Python Program: **gpb_ds_bq_unk_visualize.py [In Progress]**

TABLE SCHEMAS

1. **BigQuery KPI Definition Table**

gpb_ds_bq_kpi_definition

Schema Details Preview

Field name	Type	Mode	Description
kpi_id	INTEGER	NULLABLE	
object	STRING	NULLABLE	
formula	STRING	NULLABLE	
condition	STRING	NULLABLE	
grouping	STRING	NULLABLE	
orderBy	STRING	NULLABLE	

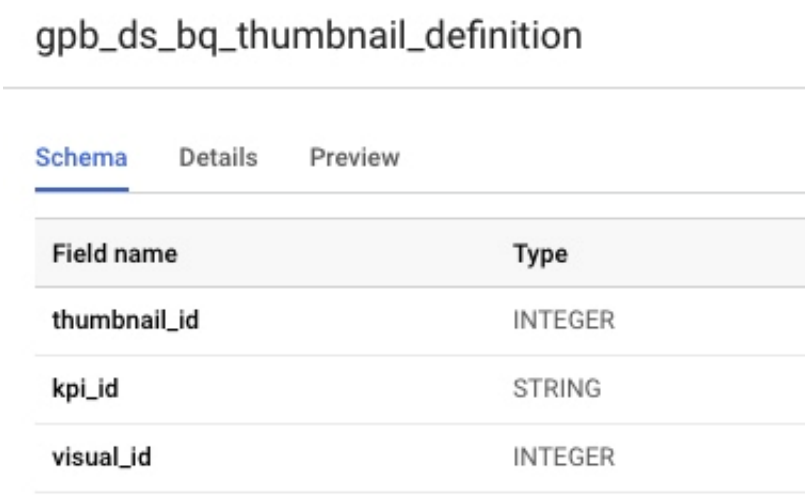
2. **BigQuery Visualization Definition Table**

gpb_ds_bq_visual_definition

Schema Details Preview

Field name	Type	Mode	Description
kpi_id	STRING	NULLABLE	
plot_type	STRING	NULLABLE	
x	STRING	NULLABLE	
y	STRING	NULLABLE	
z	STRING	NULLABLE	
groupBy	STRING	NULLABLE	
color	STRING	NULLABLE	
additionalInfo	STRING	NULLABLE	

3. **BigQuery Thumbnail Definition Table**



As we can see, the definitions are present as a **star schema**, with thumbnail definition at the centre.

PROCESS OR WORKFLOW

We expect our event object to have a minimum of the following (but not limited to) :

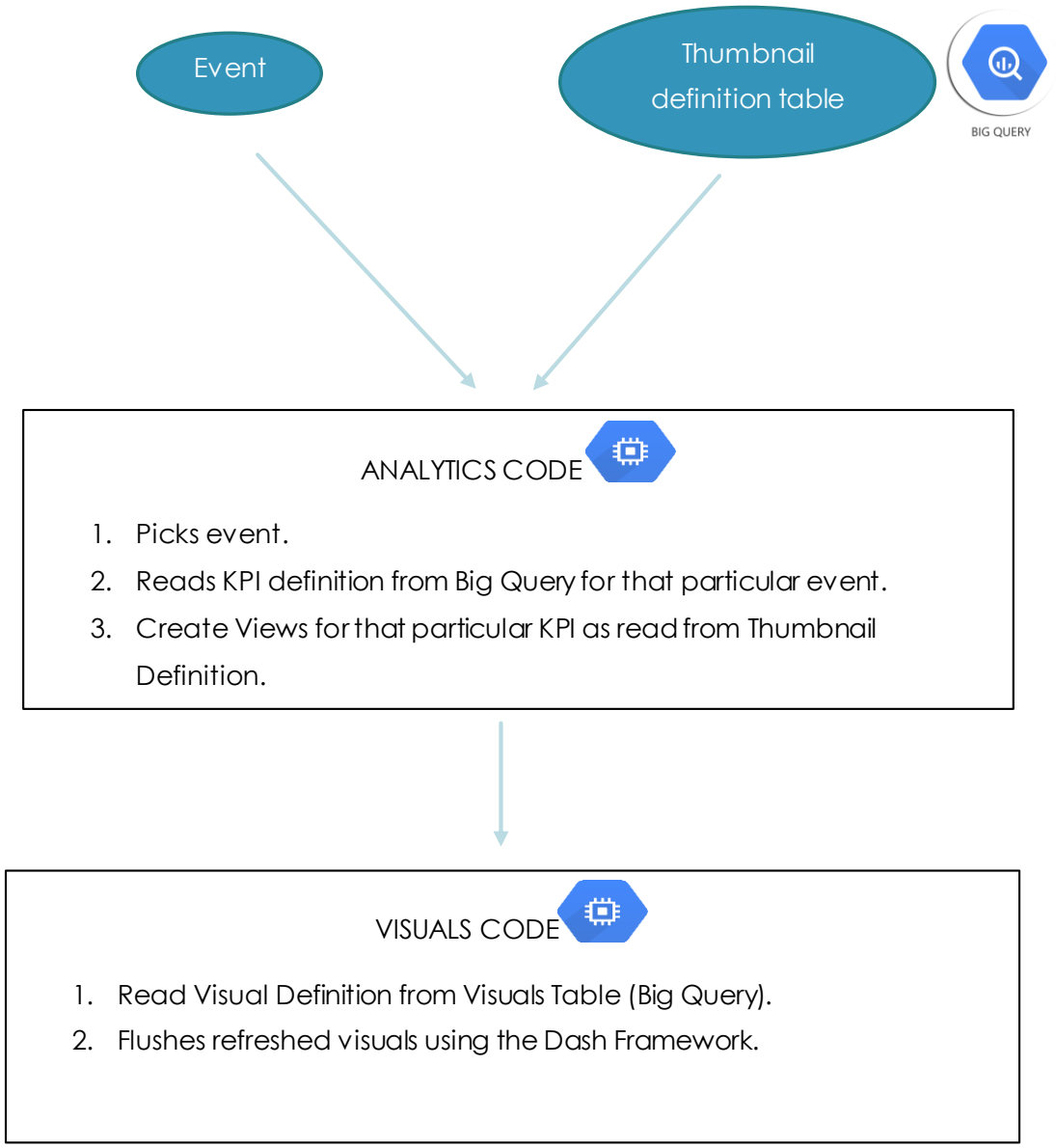
event = {

'projectId': 'asia-northeast1-231705 ' ,

'dataset_id_from': 'transformationDataset'

'interval': 12*5000

}



VISUALISATION TOOL SELECTION

In the absence of Data Studio, we need a Visualization Layer that relies ***purely on python code***. This also helps us achieve the cross platform architecture we aim to build.

We took into consideration 3 tools and our observations can be seen below:

- Matplotlib
- Plotly

- Dash

COMPONENTS	MATPLOTLIB	PLOTLY	DESCRIPTION
Produce static websites	✗	✓	Plotly forces us to produces .html files , thus providing interactive behavior.
Security Concerns	✗	✓	Plotly html source files contained data as lists in it.
Abstraction	✗	✓	Matplotlib forced us to give every minute detail, for example the positions of the bars in bar graphs. Plotly's abstraction made it easier to implement .
Creating PDFs + Images from charts	✓	✗	Charts must first be rendered in Plotly for them to be used as pdfs or images
Used with Jupyter Notebook	✓	✓	Both can be used with Jupyter Notebooks for Data Scientists
Ease of Integration with Frameworks	✗	✓	Dash is built over plotly giving us the ability to
Available as a product	✗	✓	Enterprise ready Dashboard building tool available for plotly.

Our major focus when it comes to dashboards is:

1. Breadth in Visuals offered.
2. Interactive Behavior.
4. Near real-time updates.

Dash fits perfectly when it comes to the points mentioned above. It also addresses the security concern raised by plotly, where we could see data at the source.

DASH

INTRODUCTION

Dash is a Python framework for building analytical web applications. **No JavaScript required.**

Built on top of Plotly.js, React, and Flask, Dash ties modern UI elements like dropdowns, sliders, and graphs to your analytical Python code.

HOW IT WORKS

The Dash server sends the remote client a JSON serialized representation of the layout tree which is hydrated into HTML by React in the browser. It wouldn't take much to write an HTML export function for a Python layout tree consisting only of the HTML components, however the Dash core components correspond to Dash/React components so the potentially complex custom HTML generation logic for these is handled on the client side.

CODE SNIPPET FOR REAL TIME UPDATES

```
app.layout = html.Div(children=[
        html.H3(children='GPB Data Services Analytics Dashboard', style={'textAlign':'center'}),
        dcc.Graph( id='example-graph'), dcc.Interval(
        id='interval-component', interval = 1*5000, n_intervals = 0)
    ])


@app.callback(Output('example-graph' ,'figure'),[Input('interval-component','n_intervals')])
def getDataframe(n):
        #Your logic here.
```

REFERENCES

Main Dashboard Code

Analytics Code

https://alm-confluence.systems.uk.hsbc/confluence/display/GSDS/Analytics+%3A+Insert+query+results+in+analytics_db

Visualization Code

https://alm-confluence.systems.uk.hsbc/confluence/display/GSDS/Visualize+%3A+Create+charts+and+save+in+cloud+storage+bucket

Dash references

https://plot.ly/products/dash

APPENDIX

USEFUL BIG QUERY QUERIES

Visual Definition Addition.

```
insert into `asia-northeast1-231705.analyticsdataset.gpb_ds_bq_visual_definition`
(kpi_id,plot_type,x,y,z,groupBy,color,additionalInfo,title,visual_id,legend,xlabel,ylabel) values ('1','multi-bar','transaction_date','sum_amount',NULL,'transaction_type_code','green',NULL,'TRANSACTION AMOUNTS',1,'TRANSACTION_TYPE','DATE (BY MONTH)','SUM(AMOUNTS)');
```

Thumbnail Definition addition.

```
insert into`asia-northeast1-231705.analyticsdataset.gpb_ds_bq_thumbnail_definition`
(thumbnail_id,kpi_id,visual_id) values (1,'1',1)
```

How many thumbnails are present in each Tab?

```
select tabNo,count(tabNo) as thumbnailCount
from
(SELECT CASE WHEN thumbnail_id >= 1 and thumbnail_id <= 8  THEN 'one' WHEN thumbnail_id >= 9 and thumbnail_id <= 16 THEN 'two' WHEN thumbnail_id >= 17 and thumbnail_id <= 24 THEN 'three' WHEN thumbnail_id >= 9 and thumbnail_id <= 16 then 'four'
END AS tabNo  from `asia-northeast1-231705.analyticsdataset.gpb_ds_bq_thumbnail_definition`) group by tabNo
```

AUTHORS

Akash Shah (44079135)

Aakansha Rathore (44131438)