5_Create_Data_Dictionary

May 9, 2023

1 How to create a data dictionary for your FDM

1.1 Prerequisites

Before you do anything else, if you haven't already, please open a terminal on the Jupyter Launcher and run the below (you'll only have to do this once):

conda install -c conda-forge scipy

and then:

conda install -c conda-forge tqdm

Import the packages you've already installed

[ ]:
```
# Intsall required packages
from google.cloud import bigquery
import pandas as pd
from tqdm import tqdm
```

Please read the below before you run the script

Before you run the script in the cell below, please scroll down and change the name of your dataset_id in the cell beneath this one. This is the part in brackets after 'create_data_dict' Your dataset_id is the name of the dataset - the part that follows: `yhcr-prd-phm-bia-core.`

[6]:
```
def get_num_description_column(col_name, table_name):
    """
    Takes a column name and a table name, returning a string with descriptive
    statistics for the column specified.

    Calculates the mean, median, max, min and IQR for the specified column␣
 ↪using
    BigQuery SQL and returns a string with the results concatenated together.

    Args:
        col_name (str): The name of a the numeric column.
        table_name (str): The name of the table containing the specified column.
```

```
    Returns:
        str: A string with the mean, median, max, min and IQR for the specified column.
    """
    sql_query = f"""
        WITH stats AS (
            SELECT
                AVG({col_name}) AS mean,
                MAX({col_name}) AS max,
                MIN({col_name}) AS min,
                APPROX_QUANTILES({col_name}, 2)[OFFSET(1)] AS median,
                APPROX_QUANTILES({col_name}, 4)[OFFSET(1)] AS q1,
                APPROX_QUANTILES({col_name}, 4)[OFFSET(3)] AS q3
            FROM `{table_name}`
        )
        SELECT CONCAT('Mean: ', CAST(mean AS STRING),
                      ', Median: ', CAST(median AS STRING),
                      ', Max: ', CAST(max AS STRING),
                      ', Min: ', CAST(min AS STRING),
                      ', IQR: ', CAST(q3 - q1 AS STRING))
        FROM stats
    """
    return pd.read_gbq(sql_query).iloc[0,0]


def get_date_description_column(col_name, table_name):
    """
    Takes a column name and a table name, returning a string with the min and max
    dates.

    Calculates the min and max dates for the specified column using BigQuery SQL
    and returns a string with the results concatenated together.

    Args:
        col_name (str): The name of a date column.
        table_name (str): The name of the table containing the specified column.

    Returns:
        str: A string with the min and max dates for the specified column.
    """
    sql_query = f"""
        WITH stats AS (
            SELECT
                MAX({col_name}) AS max_date,
```

```
                MIN({col_name}) AS min_date
            FROM `{table_name}`
        )
        SELECT
            CONCAT('From: ', CAST(min_date AS STRING),
                   ' To: ', CAST(max_date AS STRING))
        FROM stats
    """
    return pd.read_gbq(sql_query).iloc[0,0]


def get_bool_description_column(col_name, table_name):
    """
    Takes a column name and a table name, returning a the count of `True` and
    `False` values.

    Calculates the count of `True` and `False` values for the specified column
    using BigQuery SQL and returns a string with the results concatenated
    together.

    Args:
        col_name (str): The name of the boolean column.
        table_name (str): The name of the table containing the specified column.

    Returns:
        str: A string with the count of `True` and `False` values for the
             specified column.
    """
    sql_query = f"""
        WITH stats AS (
            SELECT
                COUNTIF({col_name} = TRUE) AS true_count,
                COUNTIF({col_name} = FALSE) AS false_count
            FROM `{table_name}`
        )
        SELECT
            CONCAT('False: ', CAST(false_count AS STRING),
                   ', True: ', CAST(true_count AS STRING))
        FROM stats
    """
    return pd.read_gbq(sql_query).iloc[0,0]


def get_string_description_column(col_name, table_name):
    sql_query = f"""
        WITH top_entries AS (
            SELECT {col_name}, COUNT(*) AS count
```

```
            FROM `{table_name}`
            GROUP BY {col_name}
            ORDER BY count DESC
            LIMIT 5
        ),
        total_entries AS (
            SELECT COUNT(DISTINCT {col_name}) AS total_count
            FROM `{table_name}`
        )
        SELECT IF((SELECT total_count FROM total_entries) > 5,
                  CONCAT('Top 5: ', STRING_AGG(CONCAT({col_name}, ': ',
                         CAST(count AS STRING)), ', ')),
                  STRING_AGG(CONCAT({col_name}, ': ',
                             CAST(count AS STRING)), ', '))
        FROM top_entries
    """
    return pd.read_gbq(sql_query).iloc[0,0]


def create_data_dict(dataset_id):

    """
    Create a data dictionary table for a BigQuery dataset.

    Takes the ID of a BigQuery dataset and creates a data_dict table in the␣
 ↪same
    dataset. `data_dict` contains information about tables in the
    dataset with names prefixed " tbl_" or "cb_":
    table name, column name, data type, and a summary
    description of each column. `description` column includes summary␣
 ↪statistics
    for numeric columns (mean, median, IQR, min,  max), the number of unique
    values and top 5 values for string columns, the  date range for date
    columns, and the count of True and False values for boolean columns.

    Args:
        dataset_id (str): The ID of the BigQuery dataset.

    Output:
        None - `data_dict` table is uploaded to bigquery dataset at "dataset_id"
    """

    client = bigquery.Client()
    dataset_ref = client.dataset(dataset_id)
    tables = list(client.list_tables(dataset_ref))
    rows = []
    table_count = 0
```

```
    output_dict = {
        "table_name": [],
        "column_name": [],
        "data_type": [],
        "description": []
    }
    for table in tables:
        if table.table_id.startswith("tbl_") or table.table_id.
↪startswith("cb_"):
            table_count += 1
            print(f"Processing table {table_count} of {len(tables)}: {table.
↪table_id}")
            table_ref = dataset_ref.table(table.table_id)
            table = client.get_table(table_ref)
            for schema_field in tqdm(table.schema):
                output_dict["table_name"].append(table.table_id)
                output_dict["column_name"].append(schema_field.name)
                output_dict["data_type"].append(schema_field.field_type)
                full_table_id = f"{dataset_id}.{table.table_id}"
                if schema_field.field_type == "STRING":
                    output_dict["description"].append(
                        get_string_description_column(schema_field.name,
                                                      full_table_id)
                    )
                elif schema_field.field_type in ["INTEGER", "FLOAT", "NUMERIC"]:
                    output_dict["description"].append(
                        get_num_description_column(schema_field.name,
                                                   full_table_id)
                    )
                elif schema_field.field_type in ["DATE", "TIMESTAMP",␣
↪"DATETIME"]:
                    output_dict["description"].append(
                        get_date_description_column(schema_field.name,
                                                    full_table_id)
                    )
                elif schema_field.field_type in ["BOOL", "BOOLEAN"]:
                    output_dict["description"].append(
                        get_bool_description_column(schema_field.name,
                                                    full_table_id)
                    )
    output_df = pd.DataFrame(output_dict)
    output_df.to_gbq(f"{dataset_id}.data_dictionary", progress_bar=False)
    print("Finished creating data_dict table")
```

Change the below cell before you do anything else

[7]:
```
create_data_dict("CB_FDM_ChildrensSocialCare")
```

```
Processing table 1 of 8: tbl_Assessments
100%|      | 11/11 [00:10<00:00,  1.03it/s]
Processing table 2 of 8: tbl_CPP
100%|      | 10/10 [00:07<00:00,  1.42it/s]
Processing table 3 of 8: tbl_CiC
100%|      | 9/9 [00:05<00:00,  1.66it/s]
Processing table 4 of 8: tbl_CiNP
100%|      | 9/9 [00:09<00:00,  1.06s/it]
Processing table 5 of 8: tbl_FactorLookup
100%|      | 4/4 [00:03<00:00,  1.24it/s]
Finished creating data_dict table
```

Now print the data dictionary to view it The data_dictionary is now available to use and view in wihtin your dataset. If you would like to display the data dictionary in Vertex, follow the instructions below

[13]:
```
%load_ext google.cloud.bigquery
```

```
The google.cloud.bigquery extension is already loaded. To reload it, use:
  %reload_ext google.cloud.bigquery
```

Print table Make sure to change the name of your dataset_id below before running this cell

Tip: if you just want to dispaly a few rows to see how it looks, remove the '#' by 'LIMIT 10' under the below query

[14]:
```
%%bigquery
SELECT
*
FROM
`yhcr-prd-phm-bia-core.CB_FDM_ChildrensSocialCare.data_dictionary`
#LIMIT 10
```

```
Query is running:   0%|          |
Downloading:   0%|          |
```

[14]:

	table_name	column_name	data_type	\
0	tbl_Assessments	StartDate	DATE	
1	tbl_Assessments	EndDate	DATE	
2	tbl_CPP	StartDate	DATE	
3	tbl_CPP	EndDate	DATE	
4	tbl_CiC	StartDate	DATE	
5	tbl_CiC	EndDate	DATE	

6	tbl_CiNP	StartDate	DATE
7	tbl_CiNP	EndDate	DATE
8	tbl_Assessments	EthnicOrigin	STRING
9	tbl_Assessments	FactorID	STRING
10	tbl_Assessments	PCArea	STRING
11	tbl_CPP	CPP_Category	STRING
12	tbl_CPP	EthnicOrigin	STRING
13	tbl_CPP	PCArea	STRING
14	tbl_CiC	EthnicOrigin	STRING
15	tbl_CiC	PCArea_Home	STRING
16	tbl_CiNP	EthnicOrigin	STRING
17	tbl_CiNP	PCArea	STRING
18	tbl_FactorLookup	FactorID	STRING
19	tbl_FactorLookup	Description	STRING
20	tbl_FactorLookup	Category	STRING
21	tbl_FactorLookup	Subcategory	STRING
22	tbl_Assessments	person_id	INTEGER
23	tbl_Assessments	AssessmentID	INTEGER
24	tbl_Assessments	YearOfBirth	INTEGER
25	tbl_Assessments	validNHS	INTEGER
26	tbl_CPP	person_id	INTEGER
27	tbl_CPP	YearOfBirth	INTEGER
28	tbl_CPP	validNHS	INTEGER
29	tbl_CiC	person_id	INTEGER
30	tbl_CiC	YearOfBirth	INTEGER
31	tbl_CiC	validNHS	INTEGER
32	tbl_CiNP	person_id	INTEGER
33	tbl_CiNP	YearOfBirth	INTEGER
34	tbl_CiNP	validNHS	INTEGER
35	tbl_Assessments	tbl_assessments_start_date	DATETIME
36	tbl_Assessments	tbl_assessments_end_date	DATETIME
37	tbl_CPP	tbl_CPP_start_date	DATETIME
38	tbl_CPP	tbl_CPP_end_date	DATETIME
39	tbl_CiC	tbl_CiC_start_date	DATETIME
40	tbl_CiC	tbl_CiC_end_date	DATETIME
41	tbl_CiNP	tbl_CiNP_start_date	DATETIME
42	tbl_CiNP	tbl_CiNP_end_date	DATETIME

	description
0	From: 2019-04-01 To: 2021-06-09
1	From: 2019-04-01 To: 2021-06-09
2	From: 2017-04-03 To: 2021-05-18
3	From: 2017-05-10 To: 2021-06-09
4	From: 2017-04-04 To: 2021-05-09
5	From: 2017-04-20 To: 2021-06-09
6	From: 2019-04-01 To: 2021-05-21
7	From: 2019-04-10 To: 2021-12-22

```
8   Top 5: White - British: 29311, Asian/British A…
9   Top 5: 3B: 7774, 4B: 6155, 17A: 5201, 20: 5178…
10  Top 5: BD5 : 5692, BD4 : 5611, BD3 : 5568, BD7…
11  Emotional Abuse: 1558, Neglect: 803, Physical …
12  Top 5: White - British: 1463, Asian/British As…
13  Top 5: BD5 : 373, BD3 : 346, BD4 : 337, BD7 : …
14  Top 5: White - British: 397, Asian/British Asi…
15  Top 5: BD3 : 110, BD5 : 109, BD4 : 92, BD7 : 8…
16  Top 5: White - British: 1824, Asian/British As…
17  Top 5: BD5 : 492, BD3 : 467, BD4 : 439, BD7 : …
18        Top 5: 11A: 1, 20: 1, 21: 1, 12A: 1, 10A: 1
19  Top 5: Child sexual exploitation: concerns tha…
20  Top 5: Privately Fostered: 5, Alcohol Misuse: …
21  Top 5: Of another person in household: 4, Of C…
22  Mean: 12060193.364455627, Median: 13016638, Ma…
23  Mean: 3684413.254429372, Median: 3781736, Max:…
24  Mean: 2010.6956610794275, Median: 2010, Max: 2…
25         Mean: 1, Median: 1, Max: 1, Min: 1, IQR: 0
26  Mean: 11867141.547368431, Median: 12992108, Ma…
27  Mean: 2010.7792869269892, Median: 2011, Max: 2…
28         Mean: 1, Median: 1, Max: 1, Min: 1, IQR: 0
29  Mean: 11757382.196648052, Median: 12974318, Ma…
30  Mean: 2008.9541899441342, Median: 2008, Max: 2…
31         Mean: 1, Median: 1, Max: 1, Min: 1, IQR: 0
32  Mean: 11948171.21228369, Median: 13009577, Max…
33  Mean: 2011.000974896413, Median: 2011, Max: 20…
34         Mean: 1, Median: 1, Max: 1, Min: 1, IQR: 0
35  From: 2019-04-01 00:00:00 To: 2021-06-09 00:00:00
36  From: 2019-04-01 00:00:00 To: 2021-06-09 00:00:00
37  From: 2017-04-03 00:00:00 To: 2021-05-18 00:00:00
38  From: 2017-05-10 00:00:00 To: 2021-06-09 00:00:00
39  From: 2017-04-04 00:00:00 To: 2021-05-09 00:00:00
40  From: 2017-04-20 00:00:00 To: 2021-06-09 00:00:00
41  From: 2019-04-01 00:00:00 To: 2021-05-21 00:00:00
42  From: 2019-04-10 00:00:00 To: 2021-12-22 00:00:00
```