Project Report

A Dataflow Journey
from PubSub to BigQuery and Data Studio

Google Cloud Pipeline
Data Engineering 2

Elnaz Dehkharghani
Student Number: 11015404
M.Sc. Big Data and Business Analytics

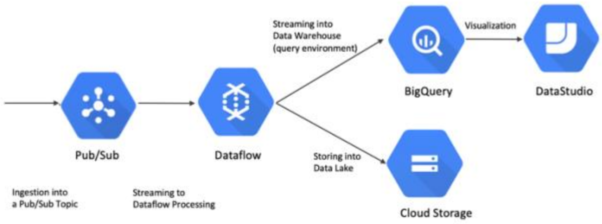
Prof. Dr. Frank Schulz

SRH Heidelberg University
16th Jun 2021

Introduction.

In Data Engineering one we made our data Pipeline in a manual way. These kinds of pipelines are great for doing some quick prototypes and exploratory analysis or collecting data in a fast way from somewhere. In contrast, this time in Data Engineering two, the goal is to look at a larger scenario and production qualities. We would have a scalable pipeline and as a result, will more consider the big data. In this schema we guaranty the on timing and availability of the system while people are using it. The whole process will be done in the cloud. So, the main goal of this project is to move from prototype level to production level.

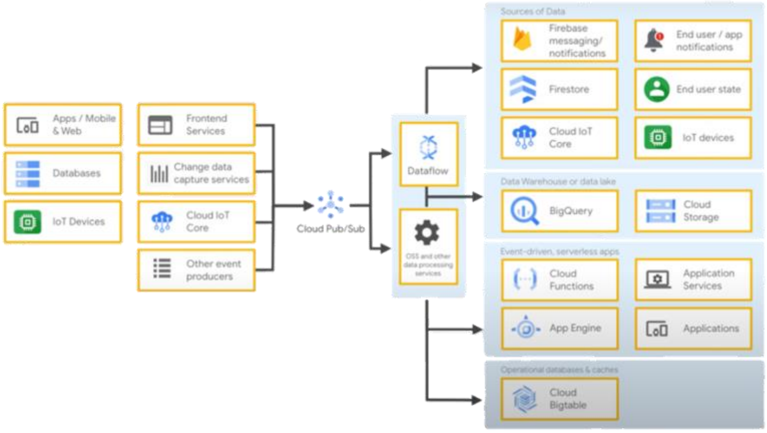
The schema of the desired pipeline:



Before deep-diving into data flowing and storytelling, we will have a brief description of all the pipeline components.

1. **Pub/Sub:** This is our messaging component. If you are ingesting large amount of data for analysis, or looking to simplify the development of event-driven microservices then you need cloud pub/sub. It is similar to Kafka that has topics. Pub/Sub helps to build robust, scalable systems of applications by integrating them asynchronously. Cloud pub/sub is a fully managed, real-time messaging service that allows you to send and receive messages between independent applications. Once an event is successfully published as a message, it becomes the job of the service to ensure that all the systems that need to react to this event get it. This architecture is more scalable and reduces dependencies. Pub/Sub can store messages for seven days, so in case of some temporary failures in some parts, they can receive it yet. As the name pub/sub indicates, cloud pub/sub supports a publisher-subscriber model. A publisher application creates and sends messages to a topic which is named resource and the messages are stored until acknowledged by all the subscribers. To receive these messages a subscriber application creates a subscription to a topic. The subscriber receives a message by either cloud pub/sub pushing them to the subscribers chosen endpoint or by subscriber pulling them from the service. When the message is acknowledged by the subscriber it will remove from the subscription backlog and not delivered again. Communicating can be one-to-many (fan-out), many-to-one (fan-in), and many-to-many.

    - **Publishers:** can be any application that can make https request to Google APIs.com.
    - **Subscribers:** Pull subscribers can also be any application although push subscribers must be a webhook endpoint that can accept post requests.



Example of incoming data sources to Cloud Pub/Sub and its targets

2. **Dataflow:** In this pipeline, dataflow is actually Apache Beam that is an open-source tool. It can do many tasks such as select, transfer, transformation, and enriching data with additional information in the streaming way "Record base". We have to consider streaming here does not mean that our data is live because they are actually stored in a pub/sub topics. So, dataflow sends data message by message or record by record. This part actually acts as the consumer. Based on Apache Beam, this Google Cloud service is used for data processing both in batch or streaming mode using the same code, providing horizontal scalability to calibrate the resources needed. The service provides many templates ready to use. Among them, there is one, that follows the pipeline schema that I am going to elaborate on later: reading messages from Pub/Sub and writing them to BigQuery.

3. **BigQuery:** Storing and querying massive datasets can be time-consuming and expensive without the right hardware and infrastructure. BigQuery is an enterprise data warehouse that solves this problem by enabling super-fast SQL queries using the processing power of Google's infrastructure.
   It is possible to control access to both the project and data based on business needs, such as giving others the ability to view or query your data. BigQuery is accessible by using the cloud console, by using the bq command-line tool, or by making calls to the BigQuery REST API using a variety of client libraries such as Java, .NET, or Python. There are also a variety of third-party tools that you can use to interact with BigQuery, such as visualizing the data.

4. **Cloud Storage:** For the longtime storing data we need this component as a data lake. We can also use some alternative cloud data lakes.

5. **Data Studio:** For the visualization aspect, we can use Google Data Studio. This is one environment that we can easily create our dashboards and visualize our data.

Advantages of using this Pipeline:

- Very fast implementation and setup
- Scalable
- Availability
- Robust
- Working with larger data

Description of Data Source and Dataset: London Crime 2008-2016

Now that we have a brief overview of our data pipeline, it is time to make familiarize ourselves with our data and making some user stories to know how to get useful insights from our data.

For this project, I am going to use London crime data between the years 2008 to 2016, which is in CSV file format and has around 13.5 million rows and 7 columns. The dataset is downloaded from the Kaggle website. Data is already cleaned and ready for doing analysis. This data counts the number of crimes at two different geographic levels of London (Lower Layer Super Output Area "LSOA" and Boroughs) by year, according to crime type. This is telling us, if the table naming convention is sensible, that this data is going to be grouped primarily by the area where the crime/s happened.

Columns are as following:

- **lsoa_code:** This represents a policing area
- **borough:** The London borough for which the statistic is related
- **major_category:** The major crime category
- **minor_category:** The minor crime category
- **value:** The count of the crime for that particular borough, in that particular month
- **year**: The year of the summary statistic
- **month**: The month of the summary statistic

User story telling:

1. How many boroughs are there in London?
2. What is the rate of crimes in each borough?
3. How many LSOA codes are there per borough?
4. How many crimes have happened per year in London from 2008 to 2016?
5. What is the overall volume of the different crimes in London?
6. How are the crimes in Croydon over the whole period?
7. What is the order of major crime categories for Croydon in 2012 that most of the crimes have happened in that year?
8. What type of crimes is growing fastest?
9. What are the different types of crimes and their quantity per year?
10. What are the minor categories in Theft and Handling?
11. What is the change in the number of crime incidents from 2011 to 2016?
12. What are the top 3 crimes per borough?

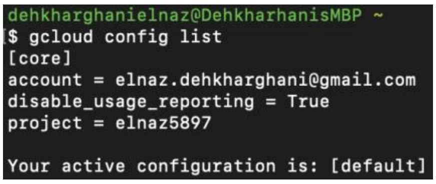
Steps for Making the Data Pipeline in Google Cloud Console

Step One: Creating a GCP Account

It is really simple just by having one Gmail account everybody can have a Google Cloud account as well. There is a good option for every individual when joining Google Cloud to have 300 USD free credit for 90 days to play around, gaining experience, and then upgrade their accounts into payable ones.

Interacting with GCP account:

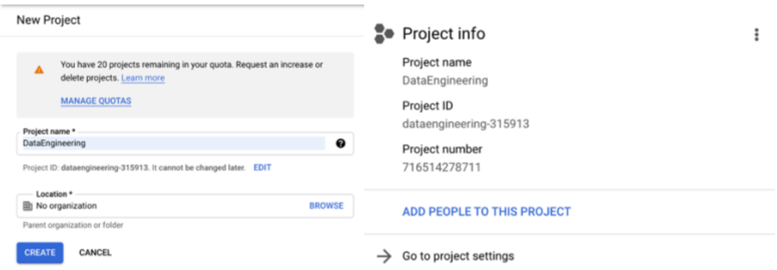
1. **Web Interface**
2. **Command Line Interface:** For using the CLI should install Google Cloud SDK base on your machine hardware name then initialize it and continue using by gcloud utility, with connecting it to your google account. For the BigQuery you can also use another utility called bq. Here I have successfully installed the GCP SDK and initialized it.



```
dehkharghanielnaz@DehkharhanisMBP ~
$ gcloud config list
[core]
account = elnaz.dehkharghani@gmail.com
disable_usage_reporting = True
project = elnaz5897

Your active configuration is: [default]
```

Step Two: Make a Project in GCP

Creating the project that we are going to deal with it in the whole process here named as DataEngineering. It is very important to save the Project ID somewhere since we will need it in further steps.



New Project

You have 20 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name *: DataEngineering

Project ID: dataengineering-315913. It cannot be changed later. EDIT

Location *: No organization BROWSE

Parent organization or folder

Project info

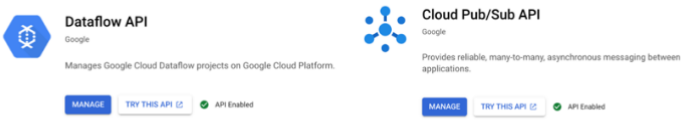
Project name: DataEngineering

Project ID: dataengineering-315913

Project number: 716514278711

Step Three: Enabling the Required APIs

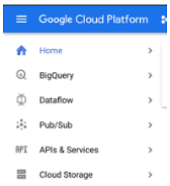
For this project, we have to enable the Dataflow API and Cloud Pub/Sub API.



Dataflow API
Google
Manages Google Cloud Dataflow projects on Google Cloud Platform.

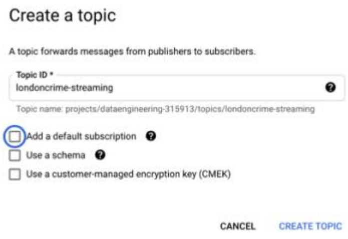
Cloud Pub/Sub API
Google
Provides reliable, many-to-many, asynchronous messaging between applications.

Since Google Cloud Platform has lots of tools, it is better to pin your required components at the beginning to have better access of them.



Step Four: Create a Topic

In this step, it is time to create our topic for sending and receiving the messages. We do this by referring to the Pub/Sub section. When we create our topic, below of it a long topic name is generated that is always accessible in the Pub/Sub section. We will need this also in further steps to make our dataflow. For the sake of this project, we do not need to make subscriptions but if we had some subscriber app, we need to define them in the subscription menu as many as that we have subscribers.



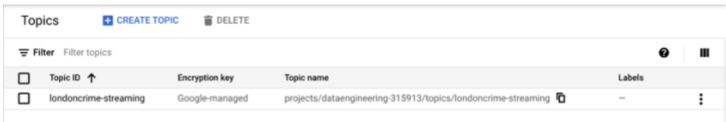
Create a topic

A topic forwards messages from publishers to subscribers.

Topic ID *: londoncrime-streaming

Topic name: projects/dataengineering-315913/topics/londoncrime-streaming

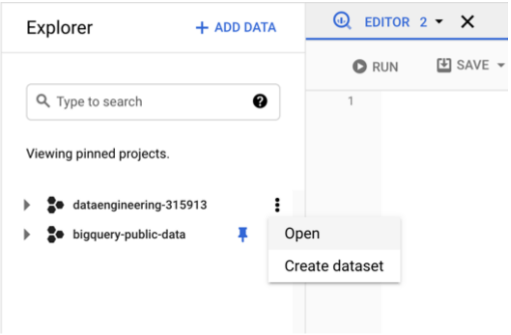
□ Add a default subscription
□ Use a schema
□ Use a customer-managed encryption key (CMEK)

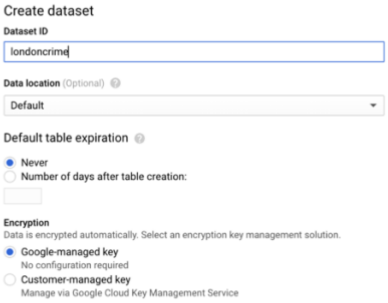


□	Topic ID ↑	Encryption key	Topic name	Labels
□	londoncrime-streaming	Google-managed	projects/dataengineering-315913/topics/londoncrime-streaming	–

Step Five: Create a Dataset and Tables

We do this by referring to the BigQuery section and add our dataset and tables there.





Create dataset

Dataset ID: londoncrime

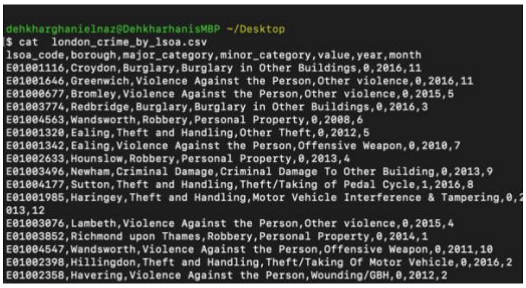
Data location (Optional): Default

Default table expiration
● Never
○ Number of days after table creation:

Encryption
Data is encrypted automatically. Select an encryption key management solution.
● Google-managed key
No configuration required
○ Customer-managed key
Manage via Google Cloud Key Management Service

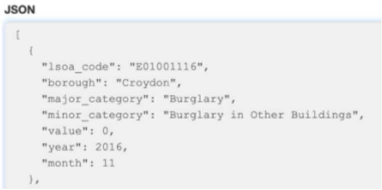
When we are going to make our table, it is very important to have a good understanding of our table schema otherwise during the sending messages will encounter problems. So, to do a better job we can use this command in the terminal and see our CSV file records there:

```
$ cat  london_crime_by_lsoa.csv
```



```
dehkharghanielnaz@DehkharhanisMBP ~/Desktop
$ cat  london_crime_by_lsoa.csv
lsoa_code,borough,major_category,minor_category,value,year,month
E01001116,Croydon,Burglary,Burglary in Other Buildings,0,2016,11
E01001646,Greenwich,Violence Against the Person,Other violence,0,2016,11
E01000677,Bromley,Violence Against the Person,Other violence,0,2015,5
E01003774,Redbridge,Burglary,Burglary in Other Buildings,0,2016,3
E01004563,Wandsworth,Robbery,Personal Property,0,2008,6
E01001320,Ealing,Theft and Handling,Other Theft,0,2012,5
E01001342,Ealing,Violence Against the Person,Offensive Weapon,0,2010,7
E01002633,Hounslow,Robbery,Personal Property,0,2013,4
E01003496,Newham,Criminal Damage,Criminal Damage To Other Building,0,2013,9
E01004177,Sutton,Theft and Handling,Theft/Taking of Pedal Cycle,1,2016,8
E01001985,Haringey,Theft and Handling,Motor Vehicle Interference & Tampering,0,2
013,12
E01003076,Lambeth,Violence Against the Person,Other violence,0,2015,4
E01003852,Richmond upon Thames,Robbery,Personal Property,0,2014,1
E01004547,Wandsworth,Violence Against the Person,Offensive Weapon,0,2011,10
E01002398,Hillingdon,Theft and Handling,Theft/Taking Of Motor Vehicle,0,2016,2
E01002358,Havering,Violence Against the Person,Wounding/GBH,0,2012,2
```

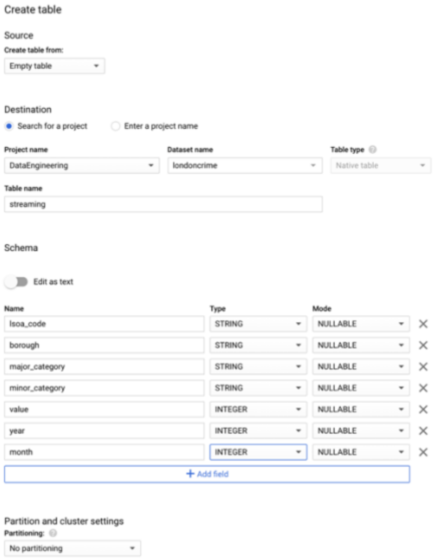
We can copy some of the records and paste them into the JSON convertor website to see that how our messages will be sent in the JSON format.



JSON

```
[
  {
    "lsoa_code": "E01001116",
    "borough": "Croydon",
    "major_category": "Burglary",
    "minor_category": "Burglary in Other Buildings",
    "value": 0,
    "year": 2016,
    "month": 11
  },
```

First four of the columns should be in string type and the last three should be in integer. Now that we have enough confidence about our data types it is time to make table schema. Since in this step I am going to send one-sample message to my table I chose an empty table with its required schema.



Create table

Source

Create table from: Empty table

Destination

● Search for a project ○ Enter a project name

Project name: DataEngineering

Dataset name: londoncrime

Table type: Native table

Table name: streaming

Schema

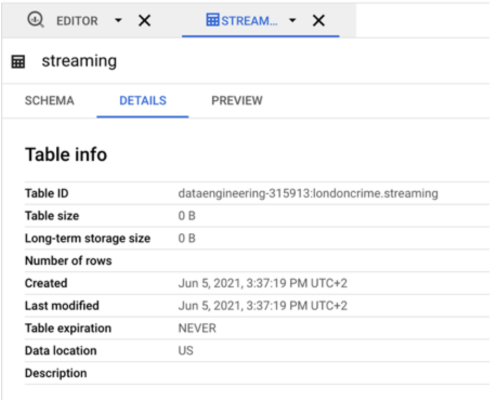
Edit as text

Name	Type	Mode
lsoa_code	STRING	NULLABLE
borough	STRING	NULLABLE
major_category	STRING	NULLABLE
minor_category	STRING	NULLABLE
value	INTEGER	NULLABLE
year	INTEGER	NULLABLE
month	INTEGER	NULLABLE

Partition and cluster settings

Partitioning: No partitioning

As soon as that we make our table, one table ID is generated for us which is very important to use in the dataflow configuration step.



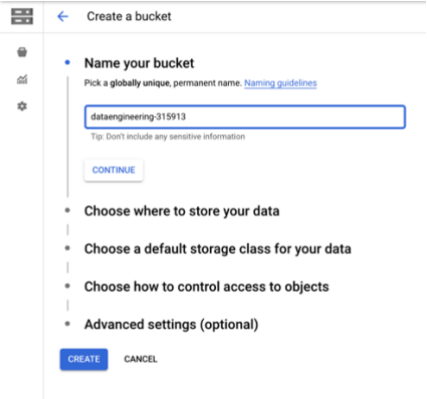
streaming

Table info

Table ID	dataengineering-315913:londoncrime.streaming
Table size	0 B
Long-term storage size	0 B
Number of rows	
Created	Jun 5, 2021, 3:37:19 PM UTC+2
Last modified	Jun 5, 2021, 3:37:19 PM UTC+2
Table expiration	NEVER
Data location	US
Description	

Step Six: Create a Bucket in Google Cloud Storage

The only thing that we need is just choosing a bucket name nothing else. I chose it the same as my project name. There are some more options for handling your bucket but this will affect the monthly fees.



Name your bucket

Pick a **globally unique**, permanent name. Naming guidelines

dataengineering-315913

Tip: Don't include any sensitive information

Choose where to store your data

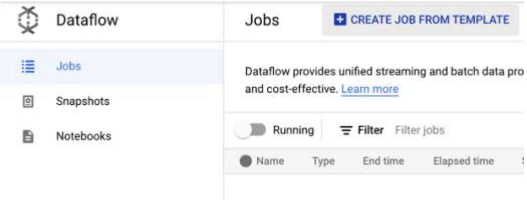
Choose a default storage class for your data

Choose how to control access to objects

Advanced settings (optional)

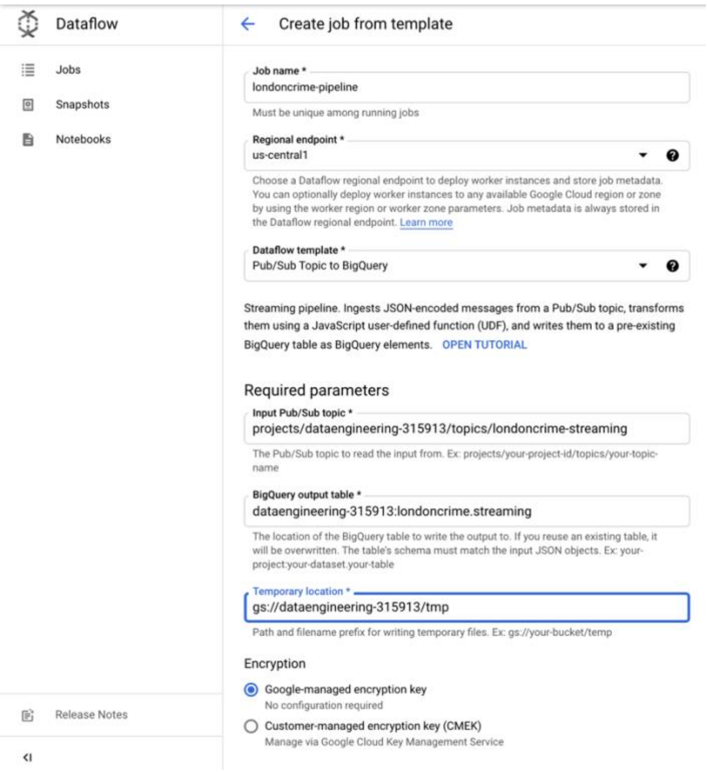
Step Seven: Create a Dataflow

Dataflow can be assumed as our consumer. We set this with the provided template as Pub/Sub topic to BigQuery.



Dataflow provides unified streaming and batch data pro and cost-effective. Learn more

It is exactly here that we need to add our topic name, BigQuery table ID and our bucket address location.



Job name *
londoncrime-pipeline
Must be unique among running jobs

Regional endpoint *
us-central1
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

Dataflow template *
Pub/Sub Topic to BigQuery

Streaming pipeline. Ingests JSON-encoded messages from a Pub/Sub topic, transforms them using a JavaScript user-defined function (UDF), and writes them to a pre-existing BigQuery table as BigQuery elements.

Required parameters

Input Pub/Sub topic *
projects/dataengineering-315913/topics/londoncrime-streaming
The Pub/Sub topic to read the input from. Ex: projects/your-project-id/topics/your-topic-name

BigQuery output table *
dataengineering-315913:londoncrime.streaming
The location of the BigQuery table to write the output to. If you reuse an existing table, it will be overwritten. The table's schema must match the input JSON objects. Ex: your-project:your-dataset.your-table

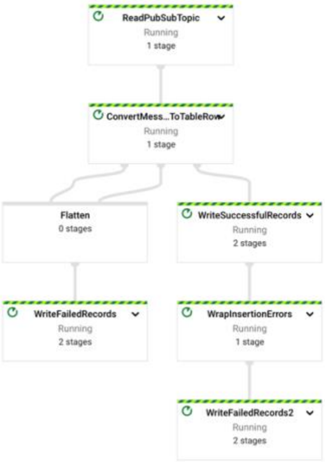
Temporary location *
gs://dataengineering-315913/tmp
Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp

Encryption

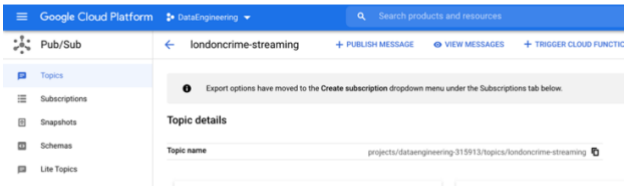
● Google-managed encryption key
No configuration required

○ Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service

When we run the job, we will get this dataflow chart that is ready to listen to our publisher and store our data into BigQuery.



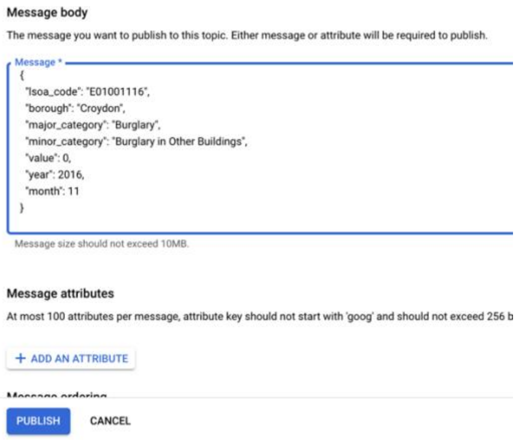
To examine if everything works well, we send one message manually in Pub/Sub "Publish Message" section. The message should be in JSON format.



Export options have moved to the **Create subscription** dropdown menu under the Subscriptions tab below.

Topic details

Topic name projects/dataengineering-315913/topics/londoncrime-streaming



Message body

The message you want to publish to this topic. Either message or attribute will be required to publish.

Message *

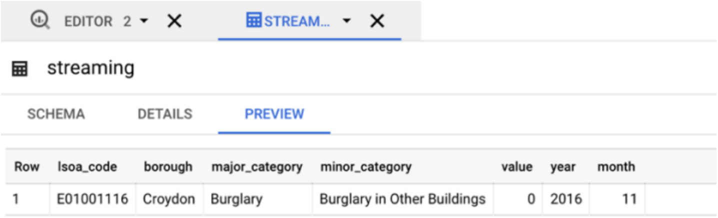
```
{
  "lsoa_code": "E01001116",
  "borough": "Croydon",
  "major_category": "Burglary",
  "minor_category": "Burglary in Other Buildings",
  "value": 0,
  "year": 2016,
  "month": 11
}
```

Message size should not exceed 10MB.

Message attributes

At most 100 attributes per message, attribute key should not start with 'goog' and should not exceed 256 b

Now we refer to the BigQuery section to see if our message is there, and yes, it is there.

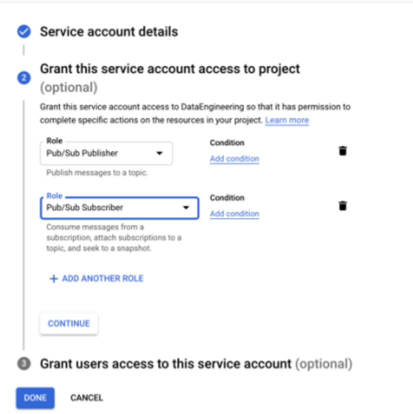


Row	lsoa_code	borough	major_category	minor_category	value	year	month
1	E01001116	Croydon	Burglary	Burglary in Other Buildings	0	2016	11

As far as everything works well, it is time to either write a python script as our publisher to send data from there or store our data file locally in Google Cloud Storage and transferring them by another job template “Text Cloud Storage to PubSub Topic”.

Making Publisher Application in Python: Building a One-to-One Pub/Sub System

In the case of making a publisher app, we need to set up a service account and Cloud IAM. This is important to our application to use it for authentication and set up Cloud IAM permissions. For this purpose, we navigate to the IAM & Admin section then select the Service accounts. We choose our service account name and click create. After that, should set the roles that we need for our apps either Pub/Sub Publisher or Pub/Sub Subscriber or maybe both. For this example, the service account needs both publishing and subscribing permissions.



Service account details

2 Grant this service account access to project (optional)

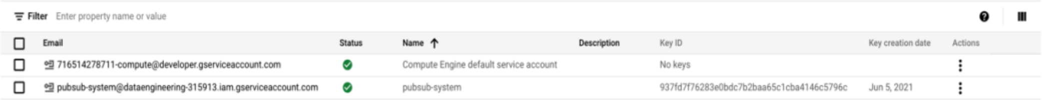
Grant this service account access to DataEngineering so that it has permission to complete specific actions on the resources in your project. Learn more

Role: Pub/Sub Publisher — Condition: Add condition
Publish messages to a topic.

Role: Pub/Sub Subscriber — Condition: Add condition
Consume messages from a subscription, attach subscriptions to a topic, and seek to a snapshot.

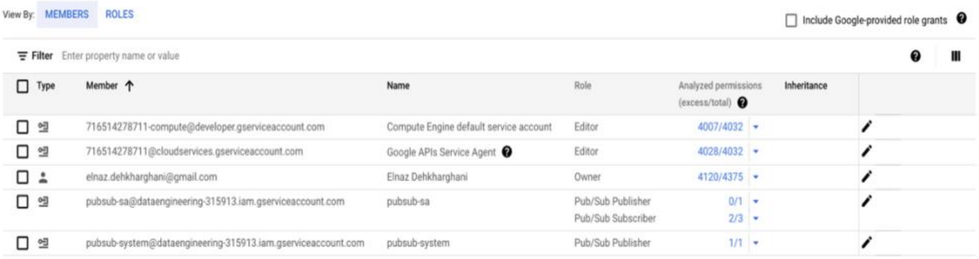
3 Grant users access to this service account (optional)

By having the service account, can go inside its link to create a key. This key will be used by the client library to access the Cloud Pub/Sub API. We select JSON and create it.



Filter Enter property name or value

☐	Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
☐	716514278711-compute@developer.gserviceaccount.com	✓	Compute Engine default service account		No keys		⋮
☐	pubsub-system@dataengineering-315913.iam.gserviceaccount.com	✓	pubsub-system		937fd7f76283e0bdc7b2baa65c1cba4146c5796c	Jun 5, 2021	⋮



View By: MEMBERS ROLES

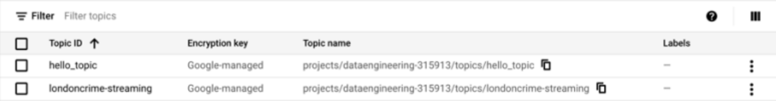
☐ Include Google-provided role grants

Filter Enter property name or value

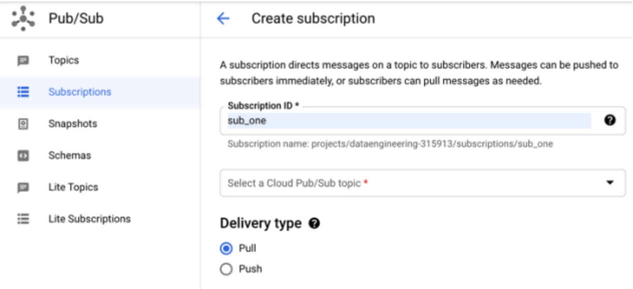
☐ Type	Member ↑	Name	Role	Analyzed permissions (excess/total)	Inheritance
☐	716514278711-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor	4007/4032	
☐	716514278711@cloudservices.gserviceaccount.com	Google APIs Service Agent	Editor	4028/4032	
☐	elnaz.dehkharghani@gmail.com	Elnaz Dehkharghani	Owner	4120/4375	
☐	pubsub-sa@dataengineering-315913.iam.gserviceaccount.com	pubsub-sa	Pub/Sub Publisher Pub/Sub Subscriber	0/1 2/3	
☐	pubsub-system@dataengineering-315913.iam.gserviceaccount.com	pubsub-system	Pub/Sub Publisher	1/1	

From this step to further it is very important to have our Google Cloud SDK installed. In continue for sending and receiving sample messages through pub.py and sub.py we should clone the repository of the Google Cloud in our specified folder and then create a virtual environment and activate it.

To examine this python publisher and subscriber I have made a new topic named “hello_topic” and made a subscriber for it as “sub_one” that pulls the messages from the topic.



□	Topic ID ↑	Encryption key	Topic name	Labels
□	hello_topic	Google-managed	projects/dataengineering-315913/topics/hello_topic	–
□	londoncrime-streaming	Google-managed	projects/dataengineering-315913/topics/londoncrime-streaming	–



A subscription directs messages on a topic to subscribers. Messages can be pushed to subscribers immediately, or subscribers can pull messages as needed.

Subscription ID *: sub_one

Subscription name: projects/dataengineering-315913/subscriptions/sub_one

Select a Cloud Pub/Sub topic *

Delivery type

● Pull
○ Push

Further Steps:

1. Download or clone the required files in local folder:
   $ git clone https://github.com/googleapis/python-pubsub.git

2. Make a virtual environment and activate it.

3. Download the required files:
   $ pip install --upgrade google-cloud-pubsub

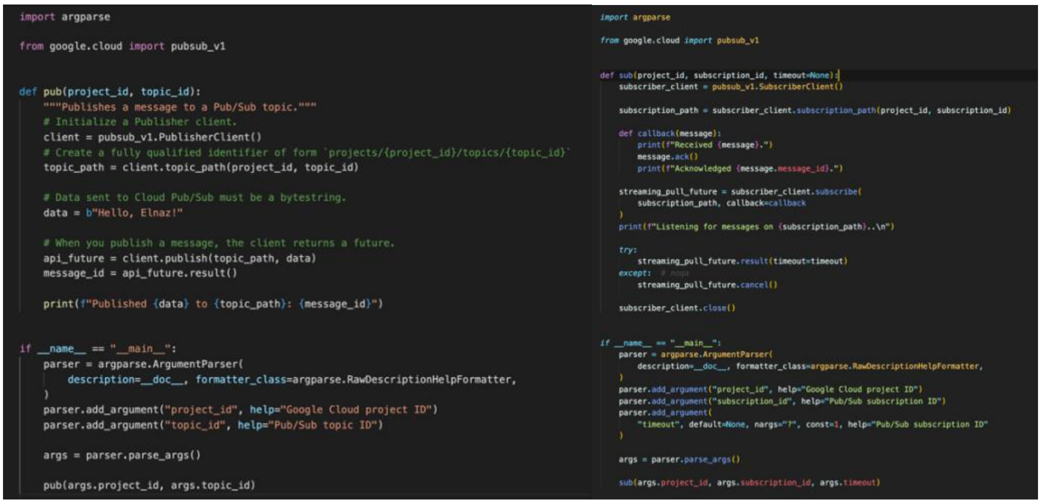
4. Set the project that are going to use:
   $ gcloud config set project dataengineering-315913

5. Watching all topics and subscriptions inside the project:
   $ gcloud pubsub topics list
   $ gcloud pubsub subscriptions list

6. Doing the Google Credentials:
   $ export GOOGLE_APPLICATION_CREDENTIALS=~/Desktop/Data_Engineering/keys/key_test.json

7. Setting the value:
   $ export PROJECT=`gcloud config get-value project`

8. Opening two new terminal that have the pub.py and sub.py inside it.

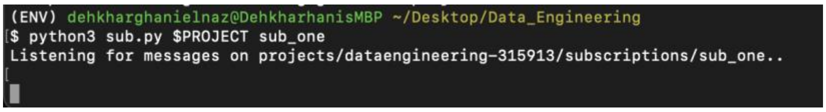


```python
import argparse

from google.cloud import pubsub_v1


def pub(project_id, topic_id):
    """Publishes a message to a Pub/Sub topic."""
    # Initialize a Publisher client.
    client = pubsub_v1.PublisherClient()
    # Create a fully qualified identifier of form `projects/{project_id}/topics/{topic_id}`
    topic_path = client.topic_path(project_id, topic_id)

    # Data sent to Cloud Pub/Sub must be a bytestring.
    data = b"Hello, Elnaz!"

    # When you publish a message, the client returns a future.
    api_future = client.publish(topic_path, data)
    message_id = api_future.result()

    print(f"Published {data} to {topic_path}: {message_id}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter,
    )
    parser.add_argument("project_id", help="Google Cloud project ID")
    parser.add_argument("topic_id", help="Pub/Sub topic ID")

    args = parser.parse_args()

    pub(args.project_id, args.topic_id)
```

```python
import argparse

from google.cloud import pubsub_v1


def sub(project_id, subscription_id, timeout=None):
    subscriber_client = pubsub_v1.SubscriberClient()

    subscription_path = subscriber_client.subscription_path(project_id, subscription_id)

    def callback(message):
        print(f"Received {message}.")
        message.ack()
        print(f"Acknowledged {message.message_id}.")

    streaming_pull_future = subscriber_client.subscribe(
        subscription_path, callback=callback
    )
    print(f"Listening for messages on {subscription_path}..\n")

    try:
        streaming_pull_future.result(timeout=timeout)
    except:  # noqa
        streaming_pull_future.cancel()

    subscriber_client.close()


if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter,
    )
    parser.add_argument("project_id", help="Google Cloud project ID")
    parser.add_argument("subscription_id", help="Pub/Sub subscription ID")
    parser.add_argument(
        "timeout", default=None, nargs="?", const=1, help="Pub/Sub subscription ID"
    )

    args = parser.parse_args()

    sub(args.project_id, args.subscription_id, args.timeout)
```

Python Programs for Publishing and Receiving Messages

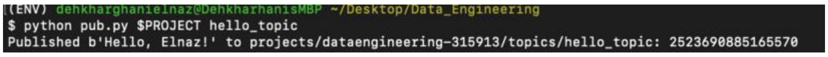
9. Running the subscriber program:
   $ python3 sub.py $PROJECT sub_one



```
(ENV) dehkharghanielnaz@DehkharhanisMBP ~/Desktop/Data_Engineering
$ python3 sub.py $PROJECT sub_one
Listening for messages on projects/dataengineering-315913/subscriptions/sub_one..
```

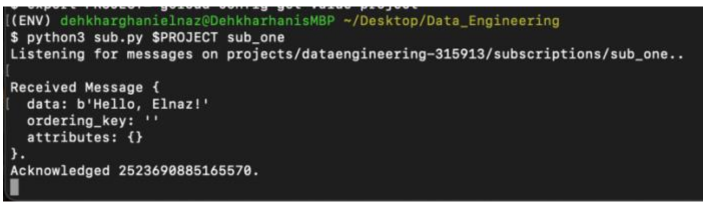
Subscriber is ready to listen to messages.

10. Finally in another terminal run the publisher program:
    $ python pub.py $PROJECT hello_topic



```
(ENV) dehkharghanielnaz@DehkharhanisMBP ~/Desktop/Data_Engineering
$ python pub.py $PROJECT hello_topic
Published b'Hello, Elnaz!' to projects/dataengineering-315913/topics/hello_topic: 2523690885165570
```

Cloud pub/Sub receive the message ID that indicate the subscriber has received the message and do not need to resend it.



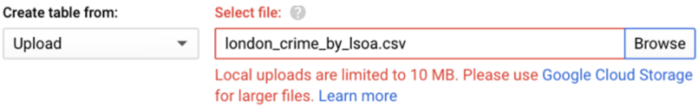
```
(ENV) dehkharghanielnaz@DehkharhanisMBP ~/Desktop/Data_Engineering
$ python3 sub.py $PROJECT sub_one
Listening for messages on projects/dataengineering-315913/subscriptions/sub_one..

Received Message {
  data: b'Hello, Elnaz!'
  ordering_key: ''
  attributes: {}
}.
Acknowledged 2523690885165570.
```

Now that we have a short experience in using python programs for sending and receiving messages through Pub/Sub it is time to send our big data from Pub/Sub topic to BigQuery. We can do this with a little change in our python program to read our CSV file record by record or in JSONL format, and by keep running one job template just like before as “PubSub Topic to BigQuery” write our real messages in BigQuery.

Step Eight: Accessing to Our London Crime Data for Doing the BigQuery in Four Different Attempts Through GCP Console

8.1 First Attempt - Uploading the Dataset from Local System to BigQuery:

In the previous steps, we have seen that I have already created my “londoncrime dataset” so the only thing that we need is to make a new table. Since my data file with 13.5 million records is 932,8 MB an error message is pop up that indicates to use google cloud storage for your large files. Here I decided to make my file shorter and try this option from my local system and after that continue with my origin file from google cloud storage.

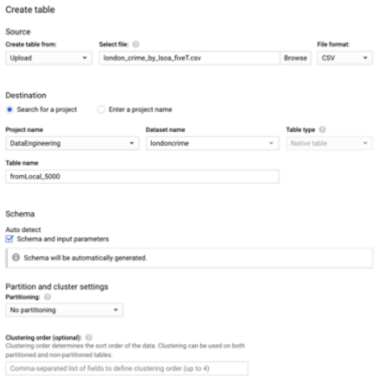


Create table from: Upload

Select file: london_crime_by_lsoa.csv

Local uploads are limited to 10 MB. Please use Google Cloud Storage for larger files. Learn more

Therefore, I selected only 5000 rows from the total records and continue my job:



Create table

Source

Create table from: Upload

Select file: london_crime_by_lsoa_fiveT.csv

File format: CSV

Destination

● Search for a project ○ Enter a project name

Project name: DataEngineering

Dataset name: londoncrime

Table type: Native table

Table name: fromLocal_5000

Schema

Auto detect

☑ Schema and input parameters

Schema will be automatically generated.

Partition and cluster settings

Partitioning: No partitioning

Clustering order (optional):

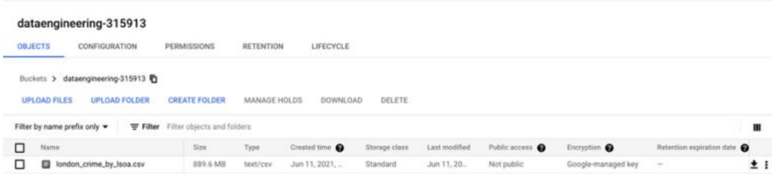
Clustering order determines the sort order of the data. Clustering can be used on both partitioned and non-partitioned tables.

Comma-separated list of fields to define clustering order (up to 4)

This time schema and file format is automatically detected. The data is inserted successfully and ready for doing queries. As I have decided to make my queries with larger amount of data in the next attempt, I will upload my origin data to Google Cloud and from there insert them to BigQuery.

8.2 Second Attempt - Uploading the CSV to Google Cloud Storage and ingesting in BigQuery:

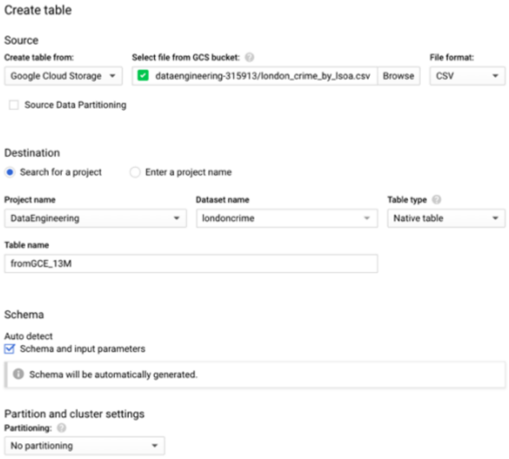
For this purpose, we refer to the created bucket with the name of dataengineering-315913 and upload our CSV file with 13.5 rows there.



dataengineering-315913

Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption	Retention expiration date
london_crime_by_lsoa.csv	889.6 MB	text/csv	Jun 11, 2021, ...	Standard	Jun 11, 20...	Not public	Google-managed key	–

Now the CSV file is actually there and ready for ingesting by BigQuery. Further, navigate to the BigQuery section and make a new table. This time instead of the upload option we select the Google Cloud Storage to upload our original file with 13.5 million rows.



Create table

Source

Create table from: Google Cloud Storage

Select file from GCS bucket: dataengineering-315913/london_crime_by_lsoa.csv

File format: CSV

□ Source Data Partitioning

Destination

● Search for a project ○ Enter a project name

Project name: DataEngineering

Dataset name: londoncrime

Table type: Native table

Table name: fromGCE_13M

Schema

Auto detect

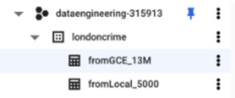
☑ Schema and input parameters

Schema will be automatically generated.

Partition and cluster settings

Partitioning: No partitioning

The file format and table schema are recognized automatically and we do not need to do anything. We have our data in two tables, one from the local uploading and the other from Google Cloud Storage:



dataengineering-315913

londoncrime

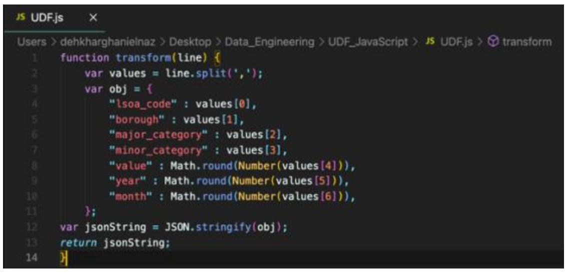
fromGCE_13M

fromLocal_5000

8.3 Attempt Three - Transferring Data to BigQuery by The Help of one Dataflow and GCS:

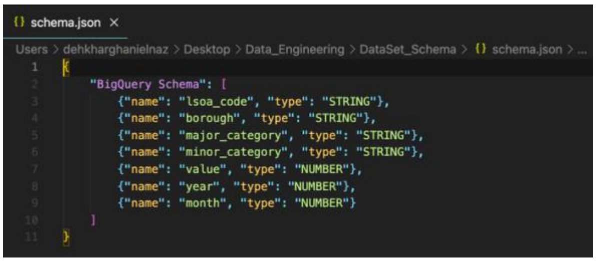
For this option we have to do a little programming in JavaScript to make the UDF function. Since I am using the free version of google cloud, in this step will use my smaller file with only 5000 rows.

- **8.3.1 Making a small UDF program in JavaScript:**


UDF.js
Users > dehkharghanielnaz > Desktop > Data_Engineering > UDF_JavaScript > UDF.js > transform

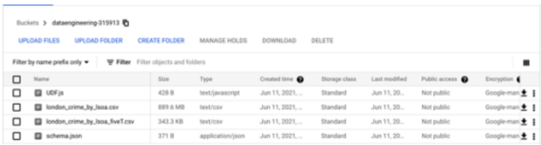
```
function transform(line) {
    var values = line.split(',');
    var obj = {
        "lsoa_code" : values[0],
        "borough" : values[1],
        "major_category" : values[2],
        "minor_category" : values[3],
        "value" : Math.round(Number(values[4])),
        "year" : Math.round(Number(values[5])),
        "month" : Math.round(Number(values[6])),
    };
    var jsonString = JSON.stringify(obj);
    return jsonString;
}
```

- **8.3.2 Making the JSON schema of the CSV file:**


schema.json
Users > dehkharghanielnaz > Desktop > Data_Engineering > DataSet_Schema > schema.json > ...

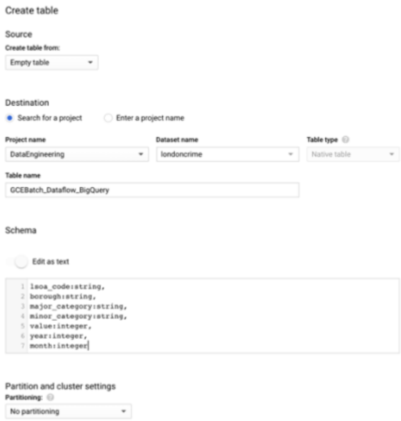
```
{
    "BigQuery Schema": [
        {"name": "lsoa_code", "type": "STRING"},
        {"name": "borough", "type": "STRING"},
        {"name": "major_category", "type": "STRING"},
        {"name": "minor_category", "type": "STRING"},
        {"name": "value", "type": "NUMBER"},
        {"name": "year", "type": "NUMBER"},
        {"name": "month", "type": "NUMBER"}
    ]
}
```

- **8.3.3 Uploading the UDF, JSON Schema and my small CSV file without header to the GCS.**


Buckets > dataengineering-315913
UPLOAD FILES
UPLOAD FOLDER
CREATE FOLDER
MANAGE HOLDS
DOWNLOAD
DELETE
Filter by name prefix only
Filter
Filter objects and folders

Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption
UDF.js	428 B	text/javascript	Jun 11, 2021, ...	Standard	Jun 11, 20...	Not public	Google-man
london_crime_by_lsoa.csv	889.6 MB	text/csv	Jun 11, 2021, ...	Standard	Jun 11, 20...	Not public	Google-man
london_crime_by_lsoa_fiveT.csv	343.3 KB	text/csv	Jun 11, 2021, ...	Standard	Jun 11, 20...	Not public	Google-man
schema.json	371 B	application/json	Jun 11, 2021, ...	Standard	Jun 11, 20...	Not public	Google-man

- **8.3.4 Creating a target empty table with defining its schema:**



Create table

Source

Create table from: Empty table

Destination

● Search for a project ○ Enter a project name

Project name: DataEngineering

Dataset name: londoncrime

Table type: Native table

Table name: GCEBatch_Dataflow_BigQuery

Schema

Edit as text

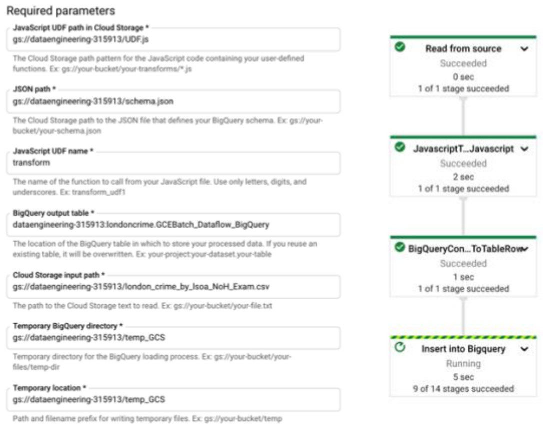
```
lsoa_code:string,
borough:string,
major_category:string,
minor_category:string,
value:integer,
year:integer,
month:integer
```

Partition and cluster settings

Partitioning: No partitioning

- **8.3.5 Create a job template from dataflow section:** Text Files on Cloud Storage to BigQuery (Batch)

  We select “Text Files on Cloud Storage to BigQuery (Batch)” template and then address the location of our UDF function, name of the function, schema of our table in JSON format, targeted BigQuery Table, our supposed CSV file without headers and two temporary directories then run the job. In this process everything goes well except inserting data into BigQuery table.



Required parameters

JavaScript UDF path in Cloud Storage *
gs://dataengineering-315913/UDF.js
The Cloud Storage path pattern for the JavaScript code containing your user-defined functions. Ex: gs://your-bucket/your-transforms/*.js

JSON path *
gs://dataengineering-315913/schema.json
The Cloud Storage path to the JSON file that defines your BigQuery schema. Ex: gs://your-bucket/your-schema.json

JavaScript UDF name *
transform
The name of the function to call from your JavaScript file. Use only letters, digits, and underscores. Ex: transform_udf1

BigQuery output table *
dataengineering-315913:londoncrime.GCEBatch_Dataflow_BigQuery
The location of the BigQuery table in which to store your processed data. If you reuse an existing table, it will be overwritten. Ex: your-project:your-dataset.your-table

Cloud Storage input path *
gs://dataengineering-315913/london_crime_by_lsoa_NoH_Exam.csv
The path to the Cloud Storage text to read. Ex: gs://your-bucket/your-file.txt

Temporary BigQuery directory *
gs://dataengineering-315913/temp_GCS
Temporary directory for the BigQuery loading process. Ex: gs://your-bucket/your-files/temp-dir

Temporary location *
gs://dataengineering-315913/temp_GCS
Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp

Read from source
Succeeded
0 sec
1 of 1 stage succeeded

JavascriptT...Javascript
Succeeded
2 sec
1 of 1 stage succeeded

BigQueryCon...ToTableRow
Succeeded
1 sec
1 of 1 stage succeeded

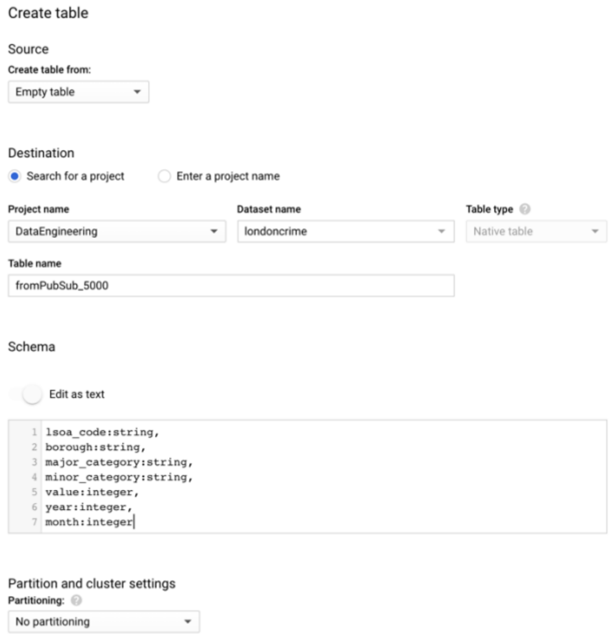
Insert into Bigquery
Running
5 sec
9 of 14 stages succeeded

8.4 Attempt Four – Transferring Data to BigQuery with the help of Two Dataflow Using the Pub/Sub:

At this point with the help of Pub/Sub I can insert my data to BigQuery table successfully.

Steps for this purpose are as followings:

- **8.4.1 Making a new empty table with its defined schema:**



Create table

Source

Create table from: Empty table

Destination

● Search for a project ○ Enter a project name

Project name: DataEngineering

Dataset name: londoncrime

Table type: Native table

Table name: fromPubSub_5000

Schema

Edit as text

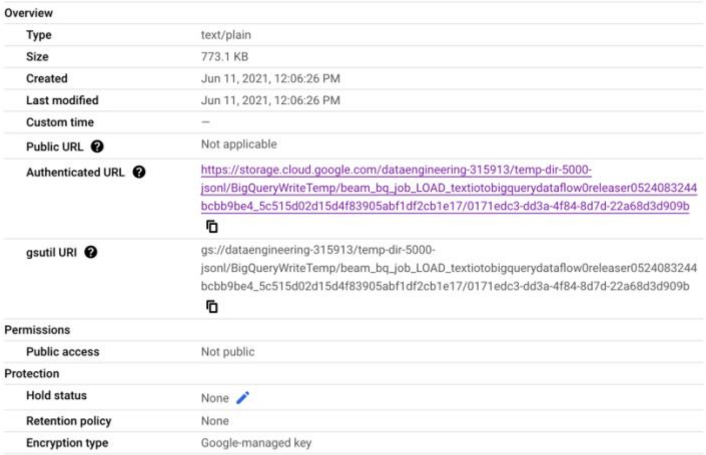
```
lsoa_code:string,
borough:string,
major_category:string,
minor_category:string,
value:integer,
year:integer,
month:integer
```

Partition and cluster settings

Partitioning: No partitioning

- **8.4.2 Using the JSONL file from the last step:**

The previous step (8.3-Attempt three) has generated a beautiful JSON line file in our temporary folder to use it in this step.



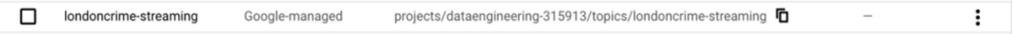
Overview	
Type	text/plain
Size	773.1 KB
Created	Jun 11, 2021, 12:06:26 PM
Last modified	Jun 11, 2021, 12:06:26 PM
Custom time	–
Public URL	Not applicable
Authenticated URL	https://storage.cloud.google.com/dataengineering-315913/temp-dir-5000-jsonl/BigQueryWriteTemp/beam_bq_job_LOAD_textiotobigquerydataflow0releaser0524083244bcbb9be4_5c515d02d15d4f83905abf1df2cb1e17/0171edc3-dd3a-4f84-8d7d-22a68d3d909b
gsutil URI	gs://dataengineering-315913/temp-dir-5000-jsonl/BigQueryWriteTemp/beam_bq_job_LOAD_textiotobigquerydataflow0releaser0524083244bcbb9be4_5c515d02d15d4f83905abf1df2cb1e17/0171edc3-dd3a-4f84-8d7d-22a68d3d909b
Permissions	
Public access	Not public
Protection	
Hold status	None
Retention policy	None
Encryption type	Google-managed key

We refer to the provided URL and copy the generated JSONL file into our local system with a .json extension. It is important that for this step and even the previous step omit the header of the files. After that we upload this JSON file to the Google Cloud Storage.

- **8.4.3 Create two different dataflow job templates:**

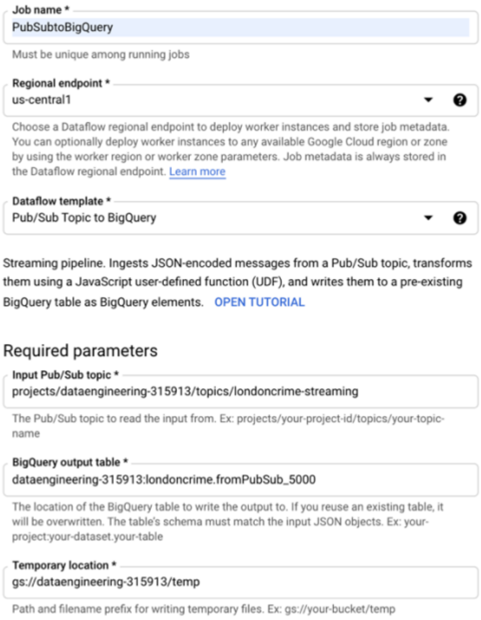
8.4.3.1 First Dataflow: Pub/Sub Topic to BigQuery

Here we fill the fields with our topic which we have created it at the beginning of this document.



☐	londoncrime-streaming	Google-managed	projects/dataengineering-315913/topics/londoncrime-streaming	–

Then address the new table and finally make a new temporary folder for it and run the job.



Job name *
PubSubtoBigQuery
Must be unique among running jobs

Regional endpoint *
us-central1
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

Dataflow template *
Pub/Sub Topic to BigQuery

Streaming pipeline. Ingests JSON-encoded messages from a Pub/Sub topic, transforms them using a JavaScript user-defined function (UDF), and writes them to a pre-existing BigQuery table as BigQuery elements. OPEN TUTORIAL

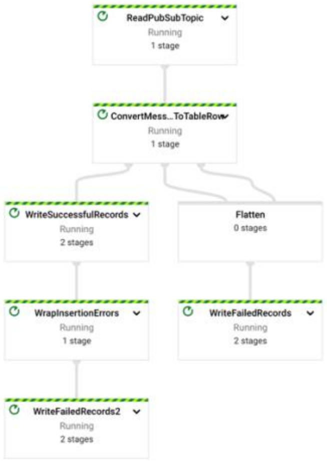
Required parameters

Input Pub/Sub topic *
projects/dataengineering-315913/topics/londoncrime-streaming
The Pub/Sub topic to read the input from. Ex: projects/your-project-id/topics/your-topic-name

BigQuery output table *
dataengineering-315913:londoncrime.fromPubSub_5000
The location of the BigQuery table to write the output to. If you reuse an existing table, it will be overwritten. The table's schema must match the input JSON objects. Ex: your-project:your-dataset.your-table

Temporary location *
gs://dataengineering-315913/temp
Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp

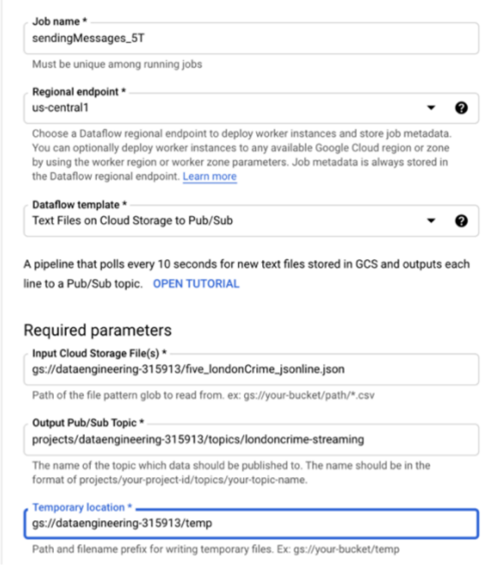
We have to keep this job running to can send our messages through topic.



8.4.3.2 Second Dataflow: Text File on Cloud Storage to Pub/Sub Topic

We have our listener, so it is time for sending our messages and we do it by making a new job template from dataflow. This time we choose "Text File on Cloud Storage to Pub/Sub" template to send all our 5000 messages through it. It is important that from the GCS select our JSONL without header otherwise we will encounter with errors in the BigQuery that cannot serialize our messages in the proper way.

As soon as that we run this job 200 messages from the whole are visible in the BigQuery table. For finalizing the message sending I had to keep this job running about 2 hours to see all 5000 rows in my BigQuery table.



Job name *
sendingMessages_5T
Must be unique among running jobs

Regional endpoint *
us-central1
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

Dataflow template *
Text Files on Cloud Storage to Pub/Sub

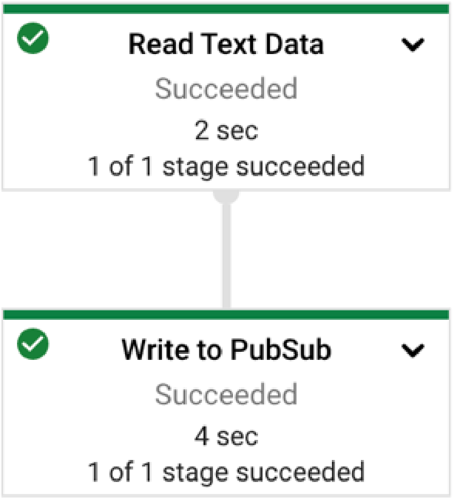
A pipeline that polls every 10 seconds for new text files stored in GCS and outputs each line to a Pub/Sub topic. OPEN TUTORIAL

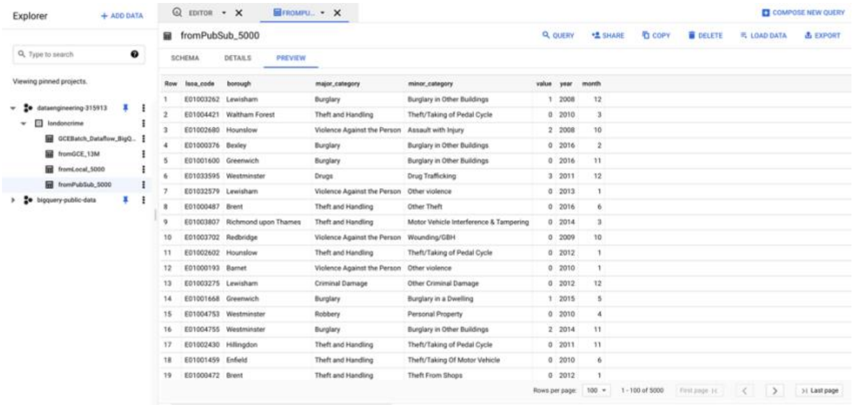
Required parameters

Input Cloud Storage File(s) *
gs://dataengineering-315913/five_londonCrime_jsonline.json
Path of the file pattern glob to read from. ex: gs://your-bucket/path/*.csv

Output Pub/Sub Topic *
projects/dataengineering-315913/topics/londoncrime-streaming
The name of the topic which data should be published to. The name should be in the format of projects/your-project-id/topics/your-topic-name.

Temporary location *
gs://dataengineering-315913/temp
Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp





fromPubSub_5000

Row	lsoa_code	borough	major_category	minor_category	value	year	month
1	E01003262	Lewisham	Burglary	Burglary in Other Buildings	1	2008	12
2	E01004421	Waltham Forest	Theft and Handling	Theft/Taking of Pedal Cycle	0	2010	3
3	E01002680	Hounslow	Violence Against the Person	Assault with Injury	2	2008	10
4	E01000376	Bexley	Burglary	Burglary in Other Buildings	0	2016	2
5	E01001600	Greenwich	Burglary	Burglary in Other Buildings	0	2016	11
6	E01033595	Westminster	Drugs	Drug Trafficking	3	2011	12
7	E01032579	Lewisham	Violence Against the Person	Other violence	0	2013	1
8	E01000487	Brent	Theft and Handling	Other Theft	0	2016	6
9	E01003807	Richmond upon Thames	Theft and Handling	Motor Vehicle Interference & Tampering	0	2014	3
10	E01003702	Redbridge	Violence Against the Person	Wounding/GBH	0	2009	10
11	E01002602	Hounslow	Theft and Handling	Theft/Taking of Pedal Cycle	0	2012	1
12	E01000193	Barnet	Violence Against the Person	Other violence	0	2010	1
13	E01003275	Lewisham	Criminal Damage	Other Criminal Damage	0	2012	12
14	E01001668	Greenwich	Burglary	Burglary in a Dwelling	1	2015	5
15	E01004753	Westminster	Robbery	Personal Property	0	2010	4
16	E01004755	Westminster	Burglary	Burglary in Other Buildings	2	2014	11
17	E01002430	Hillingdon	Theft and Handling	Theft/Taking of Pedal Cycle	0	2011	11
18	E01001459	Enfield	Theft and Handling	Theft/Taking Of Motor Vehicle	0	2010	6
19	E01000472	Brent	Theft and Handling	Theft From Shops	0	2012	1

Rows per page: 100 ▾ 1 - 100 of 5000 First page |< < > >| Last page

Step Nine: BigQuery

BigQuery and answering the predefined user stories:

Now that we could successfully store our data in four different ways into BigQuery it is time to do different queries and visualize them to have a good insight of our data. We do these queries on the data that we inserted from our GCS to BigQuery which it has 13.490.604 rows in total. We can immediately see that there are more than one lsoa_code per borough telling us that this represents a smaller more confined geographical area within a borough; this may seem straightforward and obvious, but the relationship between the variables here might help us pose some questions that we can later answer using the information available to us.

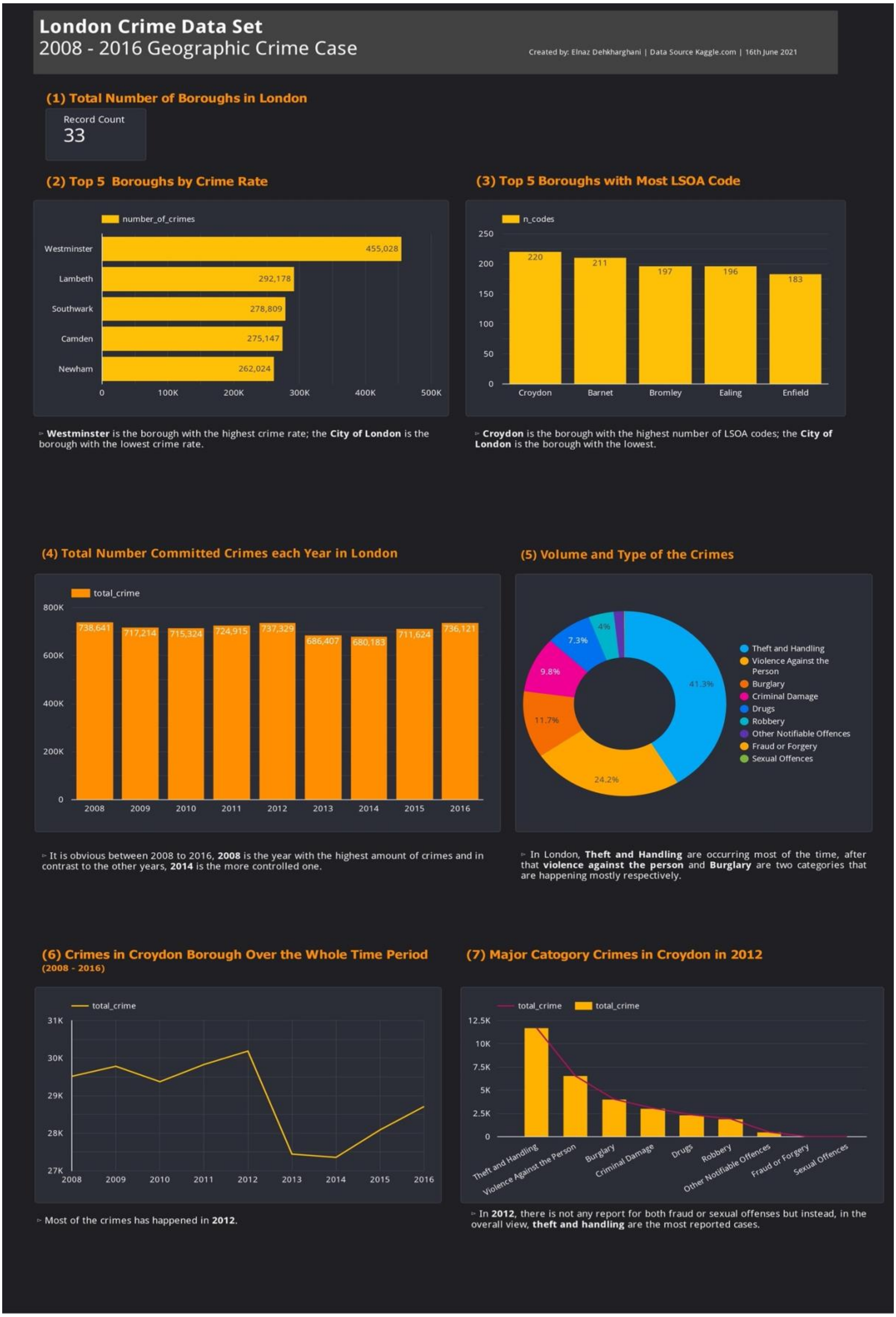
```
-- 1. How many boroughs are there in London ?

SELECT DISTINCT borough

FROM `dataengineering-315913.londoncrime.fromGCE_13M`;


-- 2. What is the rate of crimes in each borough?

SELECT borough, SUM(value) AS number_of_crimes

FROM `dataengineering-315913.londoncrime.fromGCE_13M`

GROUP BY borough

ORDER BY number_of_crimes DESC;


-- 3. How many LSOA codes are there per borough?

SELECT borough, count(DISTINCT lsoa_code) as n_codes

FROM `dataengineering-315913.londoncrime.fromGCE_13M`

GROUP BY borough

ORDER BY count(DISTINCT lsoa_code) DESC;


-- 4. How many crimes have happend per year in London from 2008 to 2016?

SELECT year, SUM(value) AS `total_crime`

FROM `dataengineering-315913.londoncrime.fromGCE_13M`

WHERE year BETWEEN 2008 AND 2016

GROUP BY year

ORDER BY total_crime DESC;


-- 5. What is the overall volume of the different crimes in London?
```

```
SELECT year, major_category, SUM(value) AS `total_crime`
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
GROUP BY year, major_category
ORDER BY year, total_crime DESC;
-- 6. How are the crimes in Croydon over the whole time period?
SELECT year, SUM(value) AS `total_crime`
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
WHERE borough = 'Croydon'
GROUP BY year
ORDER BY total_crime DESC;

-- 7. What is the order of major crime categories for Croydon in 2012 that most of the crimes has happened in that year?
SELECT major_category, sum(value) AS `total_crime`
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
WHERE borough = 'Croydon' AND year = 2012
GROUP BY major_category
ORDER BY total_crime DESC;

-- 8. What type of crimes are growing fastest? (by finding the total crime counts for the major categories across all the years in the data set)
SELECT year, major_category, SUM(value) AS `total_crime`
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
GROUP BY major_category, year
ORDER BY major_category, year;

-- 9. What are the different types of crimes and their quantity per year?
SELECT year, major_category, SUM(value) AS `total_crime`
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
WHERE year BETWEEN 2008 AND 2016
GROUP BY major_category, year
ORDER BY major_category, year;

-- 10. What are the minor categories in Theft and Handling?
SELECT major_category, minor_category, year,SUM(value) AS total_possesions
FROM `dataengineering-315913.londoncrime.fromGCE_13M`
WHERE major_category='Theft and Handling'
GROUP BY 1,2,3
ORDER BY 3 DESC;

-- 11. What is the change in the number of crime incidents from 2011 to 2016?
SELECT
  borough,
  no_crimes_2011,
  no_crimes_2016,
  no_crimes_2016 - no_crimes_2011 AS change,
  ROUND(((no_crimes_2016 - no_crimes_2011) / no_crimes_2016) * 100, 2) AS perc_change
FROM (
  SELECT
    borough,
    SUM(IF(year=2011, value, NULL)) no_crimes_2011,
    SUM(IF(year=2016, value, NULL)) no_crimes_2016
  FROM
    `dataengineering-315913.londoncrime.fromGCE_13M`
  GROUP BY
    borough )
ORDER BY
  perc_change ASC;

-- 12. What are the top 3 crimes per borough?
SELECT
  borough,
  major_category,
  rank_per_borough,
  no_of_incidents
FROM (
  SELECT
    borough,
    major_category,
    RANK() OVER(PARTITION BY borough ORDER BY SUM(value) DESC) AS rank_per_borough,
    SUM(value) AS no_of_incidents
  FROM
    `dataengineering-315913.londoncrime.fromGCE_13M`
  GROUP BY
    borough,
    major_category )
WHERE
  rank_per_borough <= 3
ORDER BY
  borough,
  rank_per_borough;
```

Step Ten: Data Visualization with Data Studio:

Visualization of the data is possible in two different ways. First making query and do the visualization at the same time in the data studio in the Explore Data section, the other way is to initially refer to the Google Data Studio make a blank report and from there connecting your BigQuery and project then do different kind of queries in a single report as your dashboard. There is also one option to share your work with others and managed it with different kind of permissions. In the following I did my visualization in data studio report and answered to all above queries with numbering them.

Link to the Dashboard: https://datastudio.google.com/reporting/ac4051cd-f515-4209-992d-f19b3e741916



London Crime Data Set

2008 - 2016 Geographic Crime Case

Created by: Elnaz Dehkharghani | Data Source Kaggle.com | 16th June 2021

(1) Total Number of Boroughs in London

Record Count

33

(2) Top 5 Boroughs by Crime Rate

▹ **Westminster** is the borough with the highest crime rate; the **City of London** is the borough with the lowest crime rate.

(3) Top 5 Boroughs with Most LSOA Code

▹ **Croydon** is the borough with the highest number of LSOA codes; the **City of London** is the borough with the lowest.

(4) Total Number Committed Crimes each Year in London

▹ It is obvious between 2008 to 2016, **2008** is the year with the highest amount of crimes and in contrast to the other years, **2014** is the more controlled one.

(5) Volume and Type of the Crimes

▹ In London, **Theft and Handling** are occurring most of the time, after that **violence against the person** and **Burglary** are two categories that are happening mostly respectively.

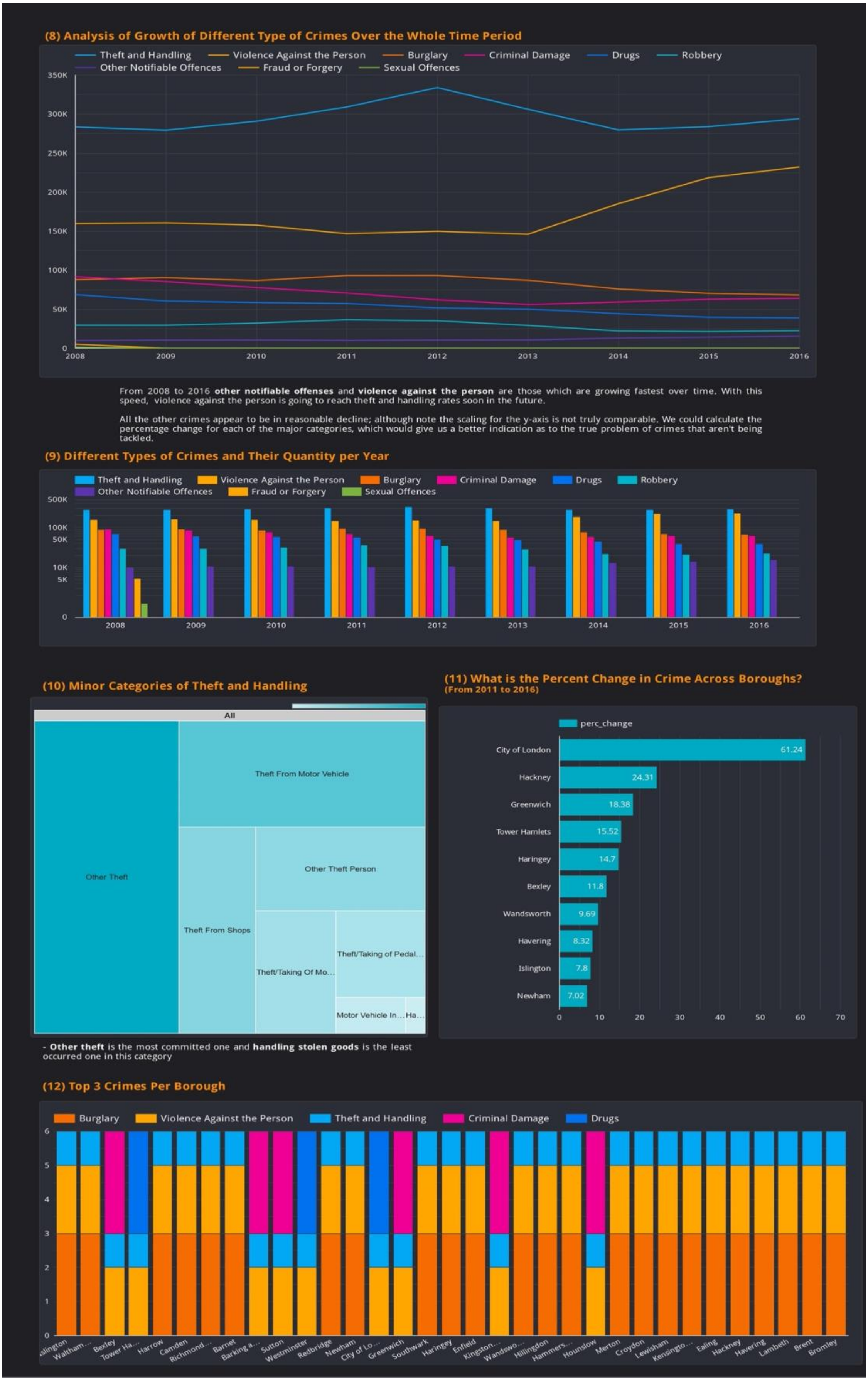
(6) Crimes in Croydon Borough Over the Whole Time Period

(2008 - 2016)

▹ Most of the crimes has happened in **2012**.

(7) Major Catogory Crimes in Croydon in 2012

▹ In **2012**, there is not any report for both fraud or sexual offenses but instead, in the overall view, **theft and handling** are the most reported cases.



(8) Analysis of Growth of Different Type of Crimes Over the Whole Time Period

From 2008 to 2016 **other notifiable offenses** and **violence against the person** are those which are growing fastest over time. With this speed, violence against the person is going to reach theft and handling rates soon in the future.

All the other crimes appear to be in reasonable decline; although note the scaling for the y-axis is not truly comparable. We could calculate the percentage change for each of the major categories, which would give us a better indication as to the true problem of crimes that aren't being tackled.

(9) Different Types of Crimes and Their Quantity per Year

(10) Minor Categories of Theft and Handling

- **Other theft** is the most committed one and **handling stolen goods** is the least occurred one in this category

(11) What is the Percent Change in Crime Across Boroughs?
(From 2011 to 2016)

(12) Top 3 Crimes Per Borough

Conclusion:

Google's Pub/Sub platform is great for handling large amounts of data and decoupling the various components of our architecture. While this project was not a real-life use case, but could have a very good experience and intertwined different Google Cloud Platform services. With a pipeline like this, you can easily migrate your data from applications to serverless, managed services on the cloud. Once the data is in BigQuery, you can do additional analytics and generate visualizations in Google Data Studio.

The End

Resources:

[1] Getting Started With Authentication:
https://cloud.google.com/docs/authentication/getting-started
[2] QuickStart: Using Client Libraries:
https://cloud.google.com/pubsub/docs/quickstart-client-libraries
[3] Building a One-to-Many Pub/Sub System:
https://cloud.google.com/pubsub/docs/building-pubsub-messaging-system
[4] What is BigQuery?
https://cloud.google.com/bigquery/docs/introduction?hl=en_US
[5] Installing Google Cloud SDK:
https://cloud.google.com/sdk/docs/install
[6] Pushing Messages to the Topic:
https://cloud.google.com/pubsub/docs/publisher
[7] Google Cloud Platform Python Samples:
https://github.com/GoogleCloudPlatform/python-docs-samples
[8] Error Handling for Google Package:
https://stackoverflow.com/questions/47011713/importerror-cannot-import-name-pubsub-v1
[9] A Dataflow Journey: From PubSub to BigQuery:
https://medium.com/codex/a-dataflow-journey-from-pubsub-to-bigquery-68eb3270c93
[10] Cloud PubSub Python Sample:
https://github.com/GoogleCloudPlatform/cloud-pubsub-samples-python/blob/master/gce-cmdline-publisher/traffic_pubsub_generator.py
[11] Setting Up a GCP Pub/Sub Integration With Python
http://www.theappliedarchitect.com/setting-up-gcp-pub-sub-integration-with-python/
[12] CSV Streamer with PubSub and BigQuery
https://github.com/thundercomb/csv-pubsub-bq-streamer
[13] Writing data from PubSub to BigQuery via Apache Beam
http://abitdeployed.com/2020/06/08/writing-data-from-pubsub-to-bigquery-via-apache-beam/