CHANGE, IMPROVE AND CONNECT.
TOGETHER WE MAKE IT SIMPLE.
VASCO DA GAMA BRIDGE | 1998 | LISBON | 17 KM
VASCO. PORTUGUESE EXPLORER AND LEADER OF PORTUGAL'S FIRST EXPEDITION TO INDIA.

PROJECT

MATEO ALVAREZ

Data Engineer

Tecnologies:

Python
Cloud Storage API
Cloud Functions
BigQuery API
MatplotLib
Pandas
Numpy
Google Cloud Platform

OBJETIVE

A company in the survey business wants to store its data from their own platforms on a more organized matter, however, there are no data experts to support any implementation, so they hired Vasco, a well known company in the data market, to support and provide best practices on data engineering.

For the first iteration, they want to start with a simple survey dataset, as you've probably seen, there are two links on the RAW DATA section, being: a description of all columns and the raw data, both in .csv. They are going to receive this information on a daily basis, once per day. With that said, their main analysis is understanding which questions have the biggest variance and the relationship between Salary and Years Coding They also need help to design the overall architecture, being, the ETL Pipeline and Final table(s) schema.

There are a few business rules that any consultant should be aware of (please bear in mind they are from the business, so we might need to help them here):

- Respondents should be unique
- If Student is blank, it means NO
- If Employment is blank, consider a full time job.
- If multiple fields are not filled (more than 3), you can disregard, since it does not count as a valid response.
- They would like to see all salaries Yearly.
- There might be some issues with encoding, business people are unaware if they used UTF8/ANS. We need this fixed in order to see all characters!

DRAWBACKS

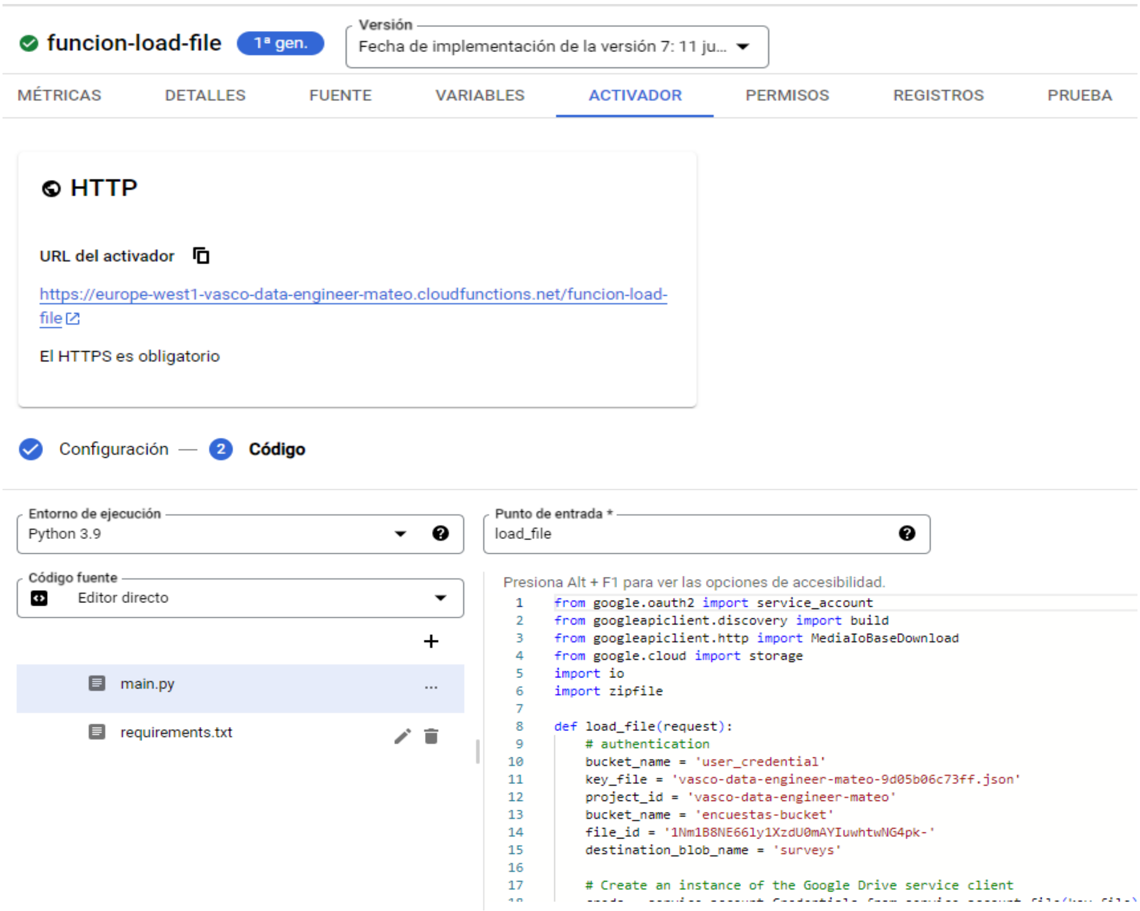
One of the obstacles I encountered while developing the project was that the original paper recommended using Microsoft Azure for ETL in the cloud. I tried many times and in many ways but I couldn't get the free account, so to avoid expenses I looked for other cloud technology options. Considering that it is a cloud in which I feel comfortable because I have both professional and academic experience, I turned to Google Cloud Platform.

When trying to do the data process, I reviewed and tried to do it from several approaches until I found the best one. One of them was Airflow on-premise for loading the file from the source (Google Drive) to Cloud Storage, but then I understood that if it was going to be a cloud-based project, the most neat and performant way was to do it in that environment from the beginning, so I decided to take the Cloud Function path which I will develop in more detail later.

Another approach was to do the transformations from Cloud Storage to load the data into BigQuery with DataFlow in a Python script with Apache Beam. Here I had several compatibility issues with Python 3.11, I tried installing and adding Python 3.9 to my environment variable which is the latest version compatible with Apache Beam. But I still had compatibility problems with Python libraries and functions that I was trying to use to read the csv in Cloud Storage. Therefore, since it was taking me quite some time to solve this, I opted for the option of using the Cloud Storage API, the BigQuery API in python to read the csv whit Pandas library, make the necessary transformations and subsequent loading. To automate everything, I ran everything in a Cloud Function function process which I will give more details about throughout this documentation.

PROCESS

The process starts with a survey file hosted on a Google Drive link in ZIP format (https://drive.google.com/file/d/1Nm1B8NE66ly1XzdU0mAYIuwhtwNG4pk-/view?usp=sharing) with the explanation of the fields in another file (https://drive.google.com/file/d/1Nm1B8NE66ly1XzdU0mAYIuwhtwNG4pk-/view?usp=sharing). As I mentioned earlier, I developed a script with python using the Google Drive API and the Cloud Storage API to extract and unzip the file to a temporary memory and then upload it to Storage (test.py). This .py file when executed does what I said in previous lines manually. So, I used Cloud Functions, included all the code in a function called “load_file” and pasted it into the Cloud Functions code reader, previously configured the function in that tool using a URL as a trigger (https://europe-west1-vasco-data-engineer-mateo.cloudfunctions.net/funcion-load-file), I also shared the drive file with the service account email of my project so that every time the file is updated at the source, that is, in Google Drive, the “load_file” function is automatically executed and the file is updated in Cloud Storage. It is worth clarifying that according to the instructions the file will be updated once a day.


funcion-load-file 1ª gen.

Versión: Fecha de implementación de la versión 7: 11 ju...

HTTP

URL del activador

https://europe-west1-vasco-data-engineer-mateo.cloudfunctions.net/funcion-load-file

El HTTPS es obligatorio

Entorno de ejecución: Python 3.9

Punto de entrada *: load_file

Código fuente: Editor directo

main.py

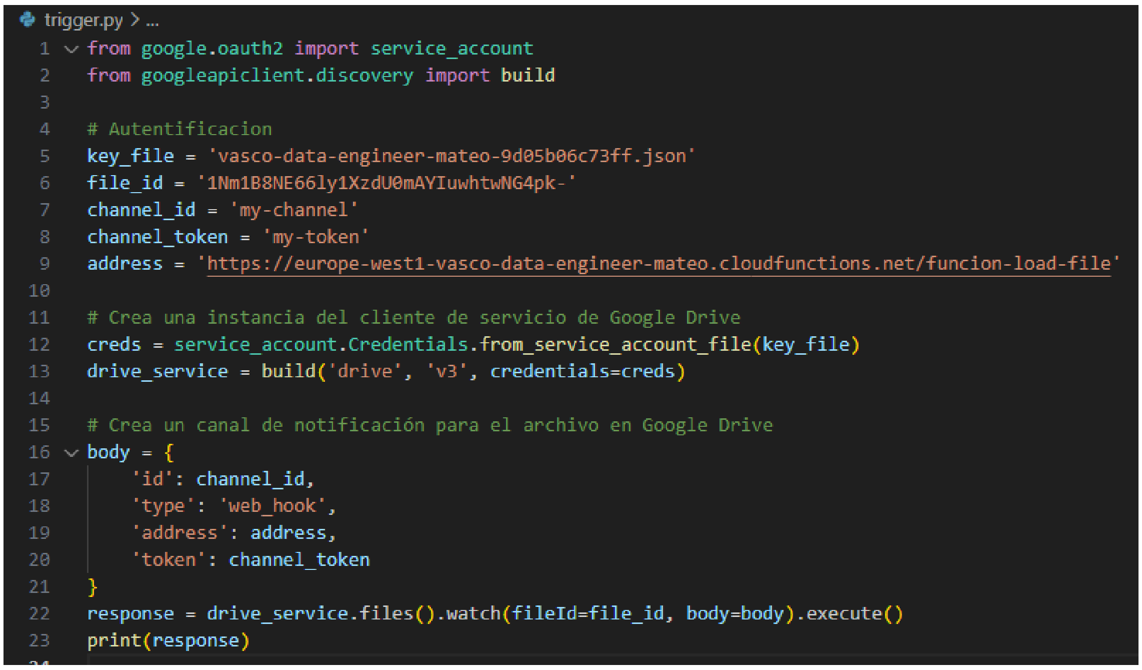
requirements.txt

Presiona Alt + F1 para ver las opciones de accesibilidad.

```python
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.http import MediaIoBaseDownload
from google.cloud import storage
import io
import zipfile

def load_file(request):
    # authentication
    bucket_name = 'user_credential'
    key_file = 'vasco-data-engineer-mateo-9d05b06c73ff.json'
    project_id = 'vasco-data-engineer-mateo'
    bucket_name = 'encuestas-bucket'
    file_id = '1Nm1B8NE66ly1XzdU0mAYIuwhtwNG4pk-'
    destination_blob_name = 'surveys'

    # Create an instance of the Google Drive service client
```

The complete code is in the file Drive_to_storage.py

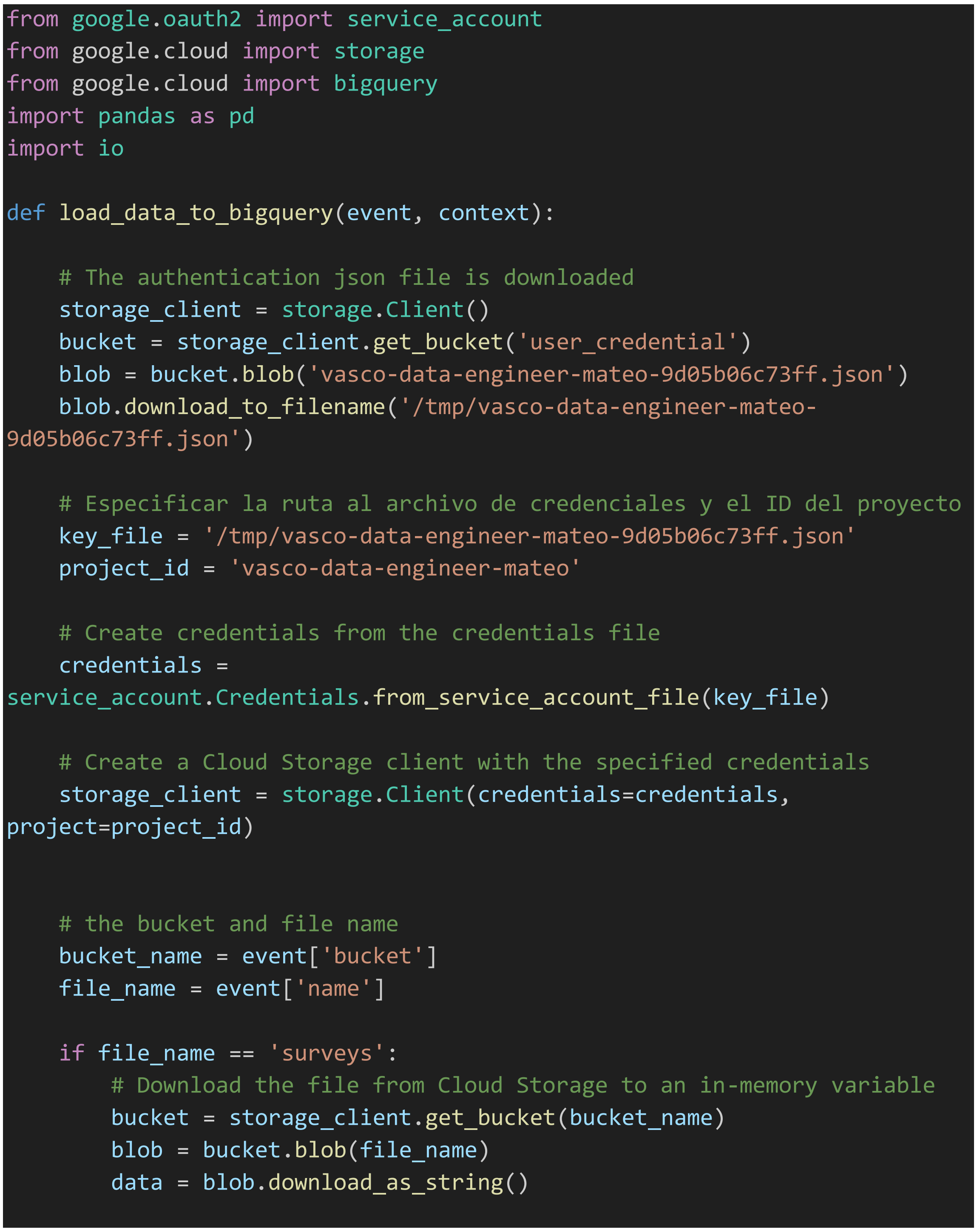
Trigger development code. trigger.py


trigger.py > ...

```python
from google.oauth2 import service_account
from googleapiclient.discovery import build

# Autentificacion
key_file = 'vasco-data-engineer-mateo-9d05b06c73ff.json'
file_id = '1Nm1B8NE66ly1XzdU0mAYIuwhtwNG4pk-'
channel_id = 'my-channel'
channel_token = 'my-token'
address = 'https://europe-west1-vasco-data-engineer-mateo.cloudfunctions.net/funcion-load-file'

# Crea una instancia del cliente de servicio de Google Drive
creds = service_account.Credentials.from_service_account_file(key_file)
drive_service = build('drive', 'v3', credentials=creds)

# Crea un canal de notificación para el archivo en Google Drive
body = {
    'id': channel_id,
    'type': 'web_hook',
    'address': address,
    'token': channel_token
}
response = drive_service.files().watch(fileId=file_id, body=body).execute()
print(response)
```

Then, after the problems already mentioned with the attempt to do it in DataFlow using a Python script with Apache Beam. Finally with the Python library, Pandas, the Cloud Storage API and the BigQuery API I read the csv, authentication through, and in the first instance I create a dataset with BigQuery called “Surveys_stage” where the raw data will be housed in the “surveys_stg” table, without transformation so that the data is safe and easily accessible in case of any inconvenience. In the second instance I make the transformations described by the client and create another dataset in this case called “Surveys_productivo” where the clean and ready-to-analyze dataset will be housed in the “surveys_prd” table.

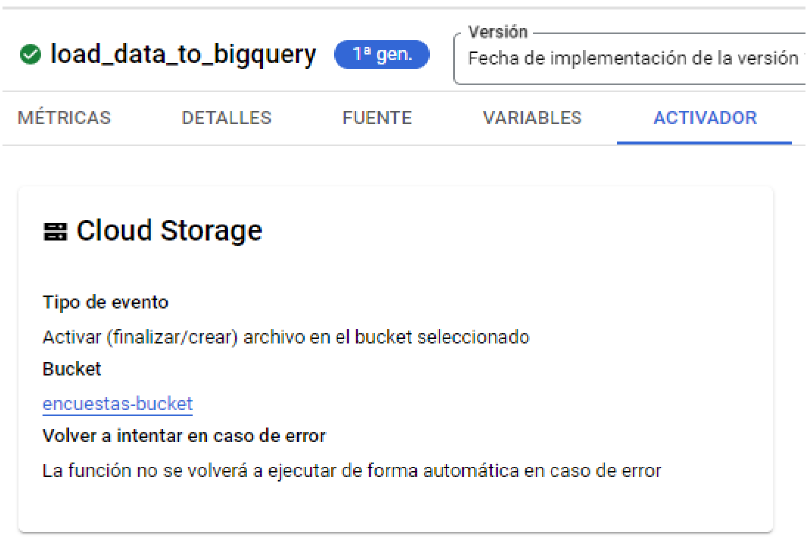
In the following image, the code of the Storage_to_bigquery.py file is shown.



```
from google.oauth2 import service_account
from google.cloud import storage
from google.cloud import bigquery
import pandas as pd
import io

def load_data_to_bigquery(event, context):

    # The authentication json file is downloaded
    storage_client = storage.Client()
    bucket = storage_client.get_bucket('user_credential')
    blob = bucket.blob('vasco-data-engineer-mateo-9d05b06c73ff.json')
    blob.download_to_filename('/tmp/vasco-data-engineer-mateo-
9d05b06c73ff.json')

    # Especificar la ruta al archivo de credenciales y el ID del proyecto
    key_file = '/tmp/vasco-data-engineer-mateo-9d05b06c73ff.json'
    project_id = 'vasco-data-engineer-mateo'

    # Create credentials from the credentials file
    credentials =
service_account.Credentials.from_service_account_file(key_file)

    # Create a Cloud Storage client with the specified credentials
    storage_client = storage.Client(credentials=credentials,
project=project_id)


    # the bucket and file name
    bucket_name = event['bucket']
    file_name = event['name']

    if file_name == 'surveys':
        # Download the file from Cloud Storage to an in-memory variable
        bucket = storage_client.get_bucket(bucket_name)
        blob = bucket.blob(file_name)
        data = blob.download_as_string()
        # Create a pandas DataFrame with the data
        df = pd.read_csv(io.BytesIO(data))

        # BigQuery credentials
        client = bigquery.Client(credentials=credentials,
project=project_id)

        # the dataset and the table into which the data will be loaded
        dataset_id = 'Surveys_stage'
        table_id = 'surveys_stg'

        # Create a reference to the table
        table_ref = client.dataset(dataset_id).table(table_id)

        # Load the DataFrame without transformations into the BigQuery
surveys_stg table
        job_config = bigquery.LoadJobConfig(
            write_disposition='WRITE_TRUNCATE'
        )

        job = client.load_table_from_dataframe(df, table_ref,
job_config=job_config)
        job.result()

        # Applies the necessary transformations to the DataFrame
        df = df.drop_duplicates()
        df['Student'] = df['Student'].fillna('No')
        df['Employment'] = df['Employment'].fillna('Full-time')
        df = df.dropna(thresh=len(df.columns)-3)

        # the dataset and the table into which the data will be loaded
        dataset_id = 'Surveys_productivo'
        table_id = 'surveys_prd'

        # Create a reference to the table
        table_ref = client.dataset(dataset_id).table(table_id)

        # Load the DataFrame into the BigQuery table with UTF 8 decoding
        job_config = bigquery.LoadJobConfig(
            encoding='UTF-8',
            write_disposition='WRITE_TRUNCATE'
        )
        job = client.load_table_from_dataframe(df, table_ref,
job_config=job_config)
        job.result()
```

As seen in the previous code, I also defined the code that searches for a file in the “encuestas_bucket” bucket which, through an if condition, indicates that it only acts if the file is called surveys. In that case, the previously indicated process is initiated. In summary, I created another function in Cloud Function with this same code that detects every time the file is updated to execute the code and automatically update the tables created in BigQuery. For this I had to choose another trigger for the function configuration options, in this case it was going to be Cloud Storage specifying the bucket id.

In the following images you will be able to verify this, and also you will be able to see a test that I carried out by manually executing the test.py file that loads the file manually to Cloud Storage so that the function in Cloud Function is activated which I called “load_data_to_bigquery”.



Cloud Storage

Tipo de evento

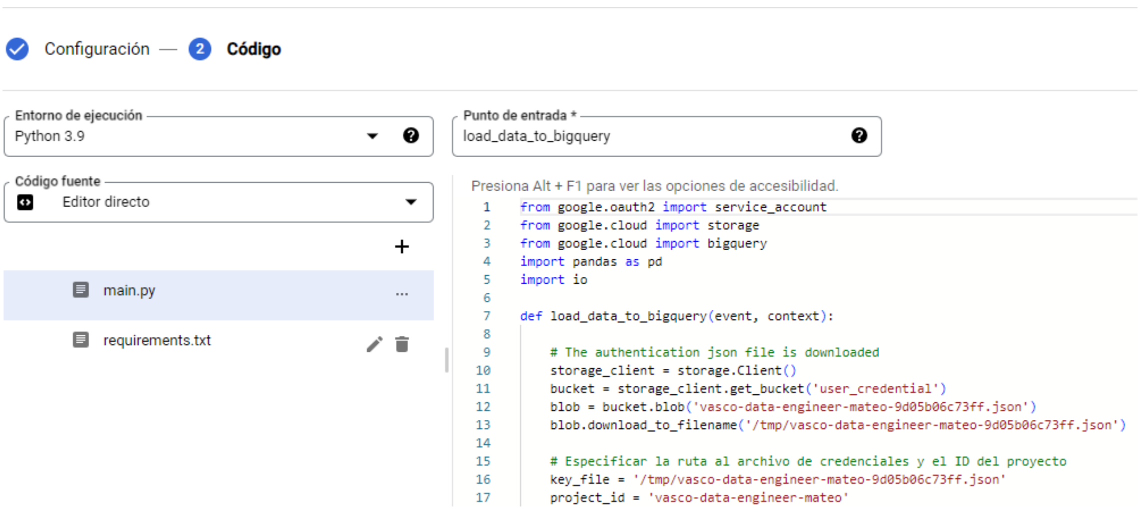
Activar (finalizar/crear) archivo en el bucket seleccionado

Bucket

encuestas-bucket

Volver a intentar en caso de error

La función no se volverá a ejecutar de forma automática en caso de error



Entorno de ejecución: Python 3.9

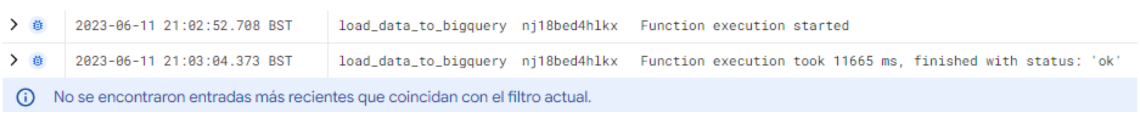
Punto de entrada *: load_data_to_bigquery

Código fuente: Editor directo

Presiona Alt + F1 para ver las opciones de accesibilidad.

```python
from google.oauth2 import service_account
from google.cloud import storage
from google.cloud import bigquery
import pandas as pd
import io

def load_data_to_bigquery(event, context):

    # The authentication json file is downloaded
    storage_client = storage.Client()
    bucket = storage_client.get_bucket('user_credential')
    blob = bucket.blob('vasco-data-engineer-mateo-9d05b06c73ff.json')
    blob.download_to_filename('/tmp/vasco-data-engineer-mateo-9d05b06c73ff.json')

    # Especificar la ruta al archivo de credenciales y el ID del proyecto
    key_file = '/tmp/vasco-data-engineer-mateo-9d05b06c73ff.json'
    project_id = 'vasco-data-engineer-mateo'
```

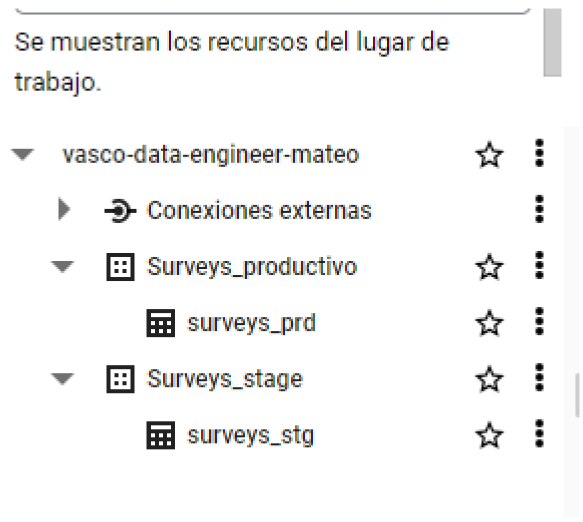
the code is identical to the one specified above



```
2023-06-11 21:02:52.708 BST   load_data_to_bigquery  nj18bed4hlkx   Function execution started
2023-06-11 21:03:04.373 BST   load_data_to_bigquery  nj18bed4hlkx   Function execution took 11665 ms, finished with status: 'ok'
```

No se encontraron entradas más recientes que coincidan con el filtro actual.

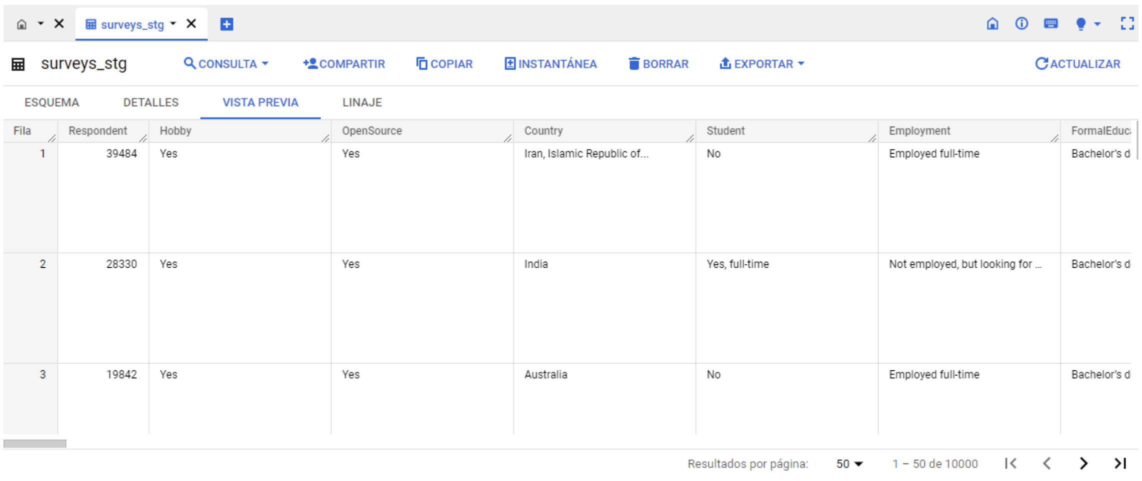
With these two functions executing correctly, the data that is updated every day at the source will automatically be updated both in Cloud Storage and in BigQuery, the stage table “surveys_stg” and with the necessary transformations in the table “surveys_prd”.


Se muestran los recursos del lugar de trabajo.

- vasco-data-engineer-mateo
  - Conexiones externas
  - Surveys_productivo
    - surveys_prd
  - Surveys_stage
    - surveys_stg

Table fragment since it is too big to be able to capture.

surveys_stg

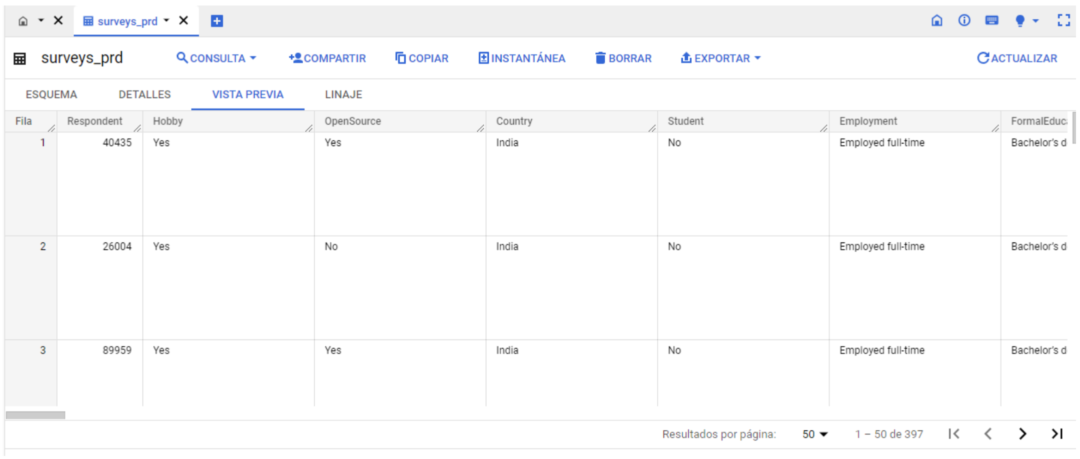


surveys_stg

Fila	Respondent	Hobby	OpenSource	Country	Student	Employment	FormalEduc
1	39484	Yes	Yes	Iran, Islamic Republic of...	No	Employed full-time	Bachelor's d
2	28330	Yes	Yes	India	Yes, full-time	Not employed, but looking for ...	Bachelor's d
3	19842	Yes	Yes	Australia	No	Employed full-time	Bachelor's d

Resultados por página: 50 1 – 50 de 10000

Surveys_prd



surveys_prd

Fila	Respondent	Hobby	OpenSource	Country	Student	Employment	FormalEduc
1	40435	Yes	Yes	India	No	Employed full-time	Bachelor's d
2	26004	Yes	No	India	No	Employed full-time	Bachelor's d
3	89959	Yes	Yes	India	No	Employed full-time	Bachelor's d

Resultados por página: 50 1 – 50 de 397

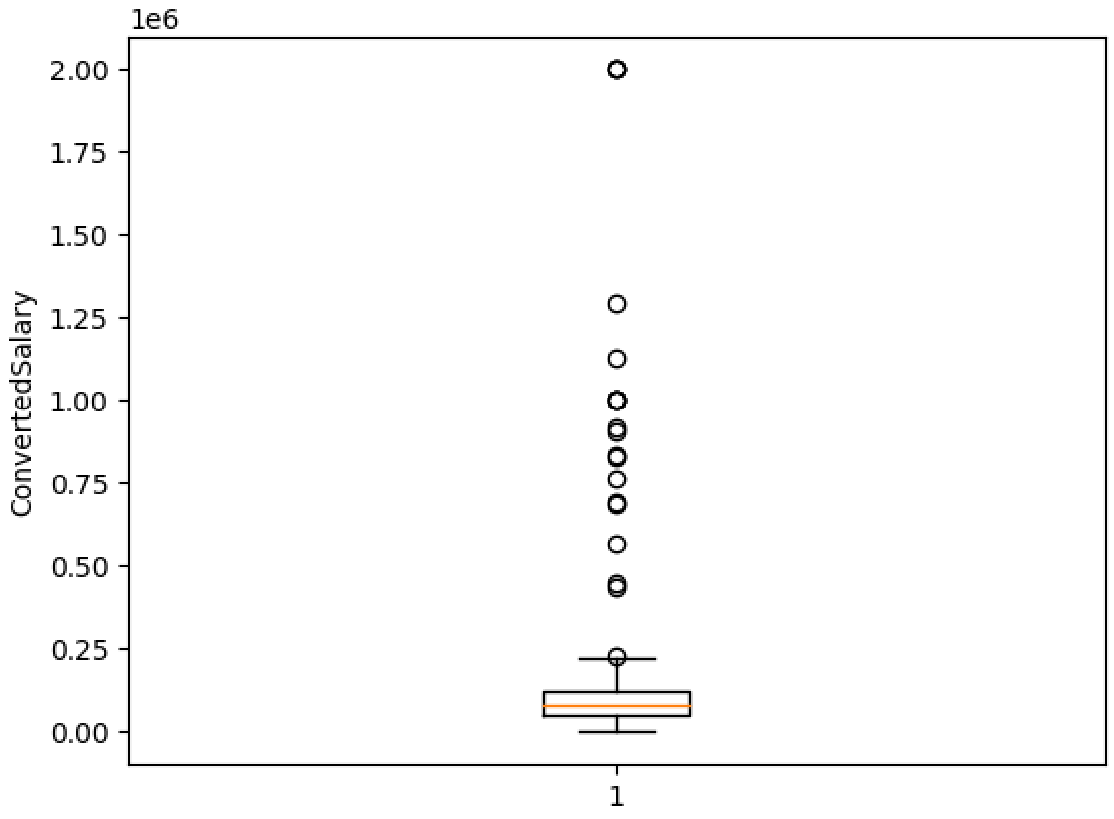
ANALYSIS

The analysis was done in a Jupyter notebook (analysis.ipynb) where in addition to the task of identifying the relationship between the annual salary of the surveyed developers and the years of coding, I decided to relate the salary also with the formal education achieved, satisfaction with their respective careers and the size of the company in which they collaborate. For this I used the BigQuery API for connection and data retrieval, the Pandas, Matplotlib for plotting, and Numpy libraries.

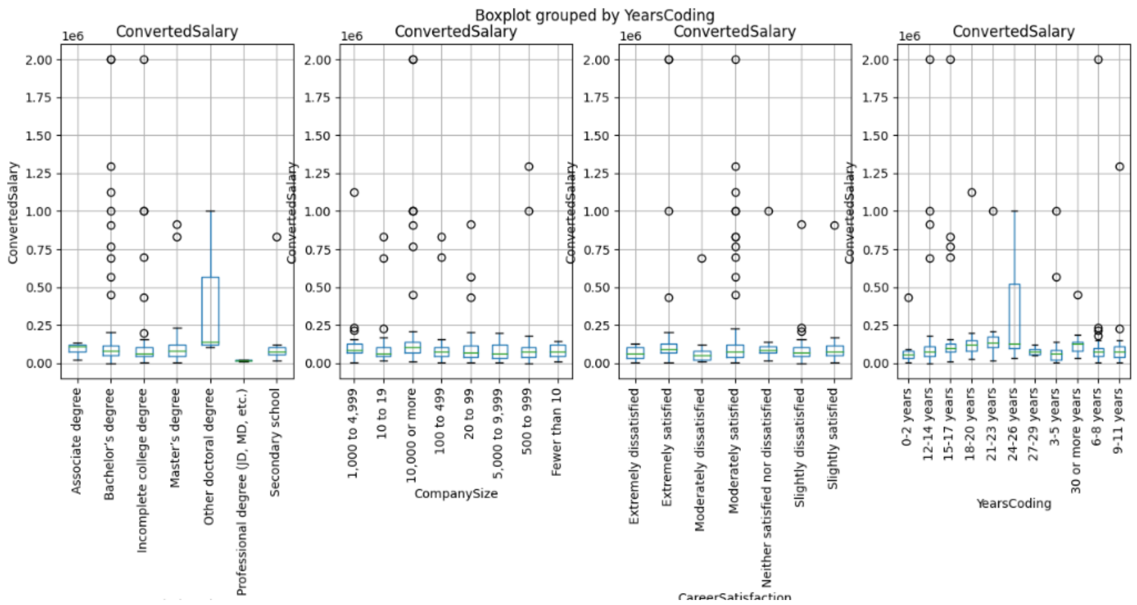
I detected outliers in salaries to try to prevent them from skewing the analysis, variation of data, dispersion for each field, distribution of each of them.

I decided to use both the mean and median total and partial as a reference for the graphs that can clarify what the data is telling us and draw conclusions. The mean was going to be directly influenced by outliers so I also included the median.

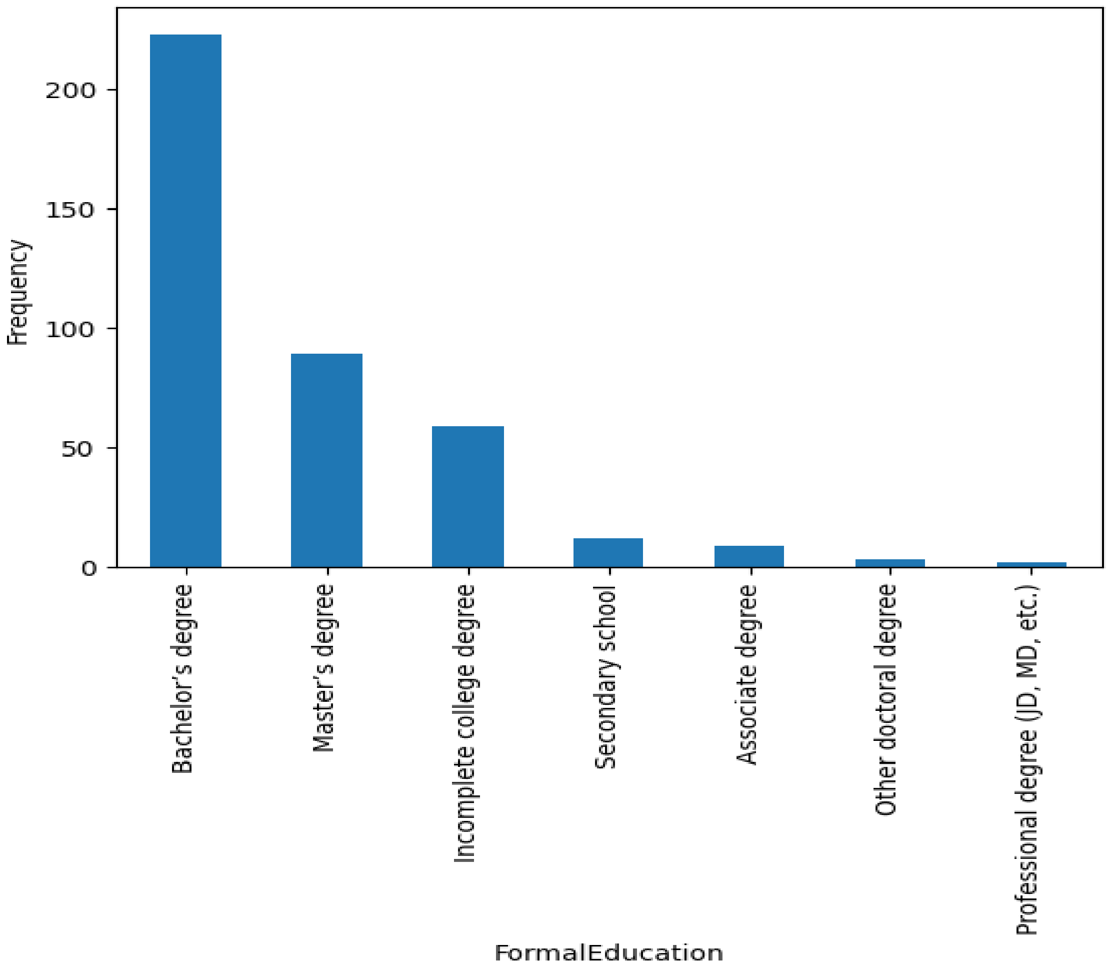
Outliers in the salaries.



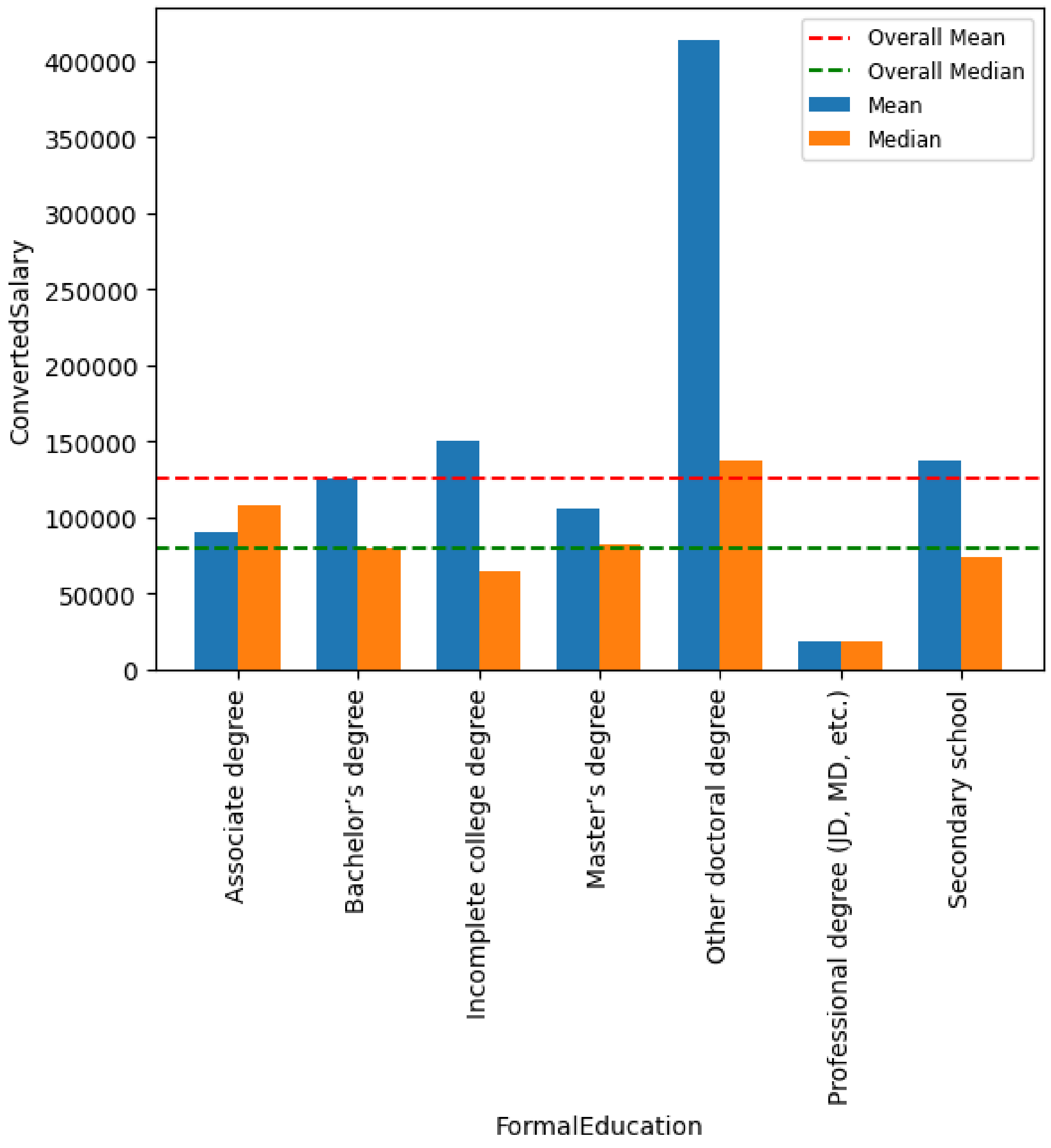
Dispersion between salaries and the fields 'FormalEducation', 'Company Size', 'Career Satisfaction' and 'YearsCoding'



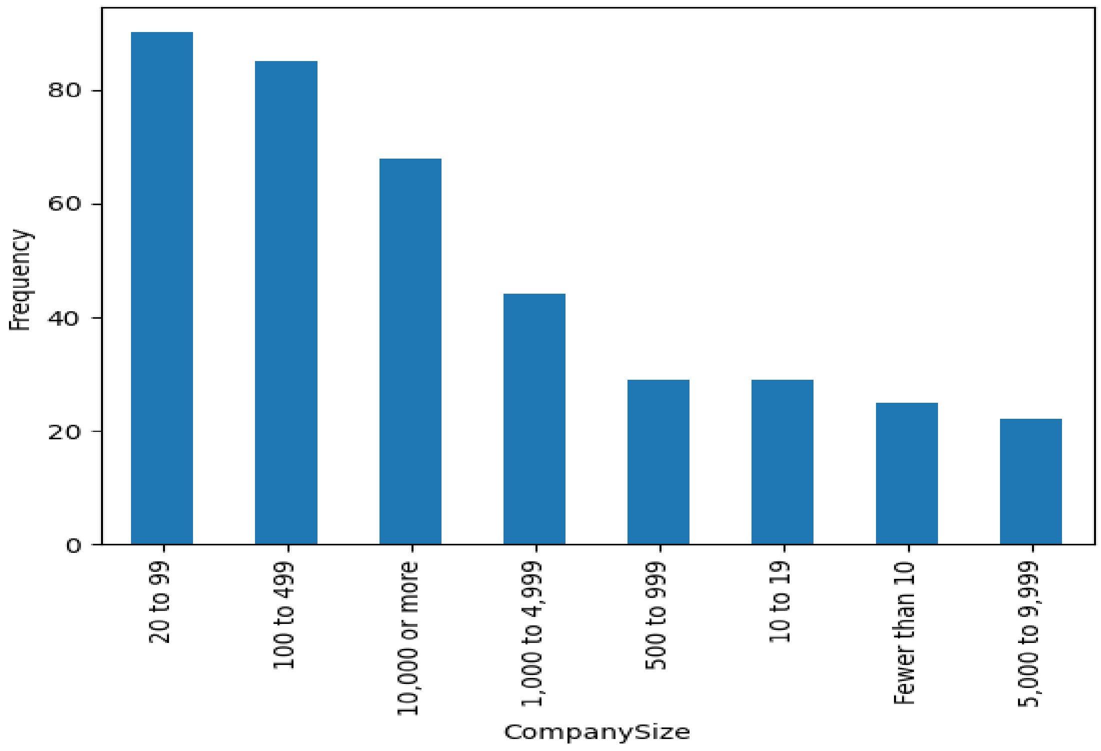
Distribution of the different levels of formal education



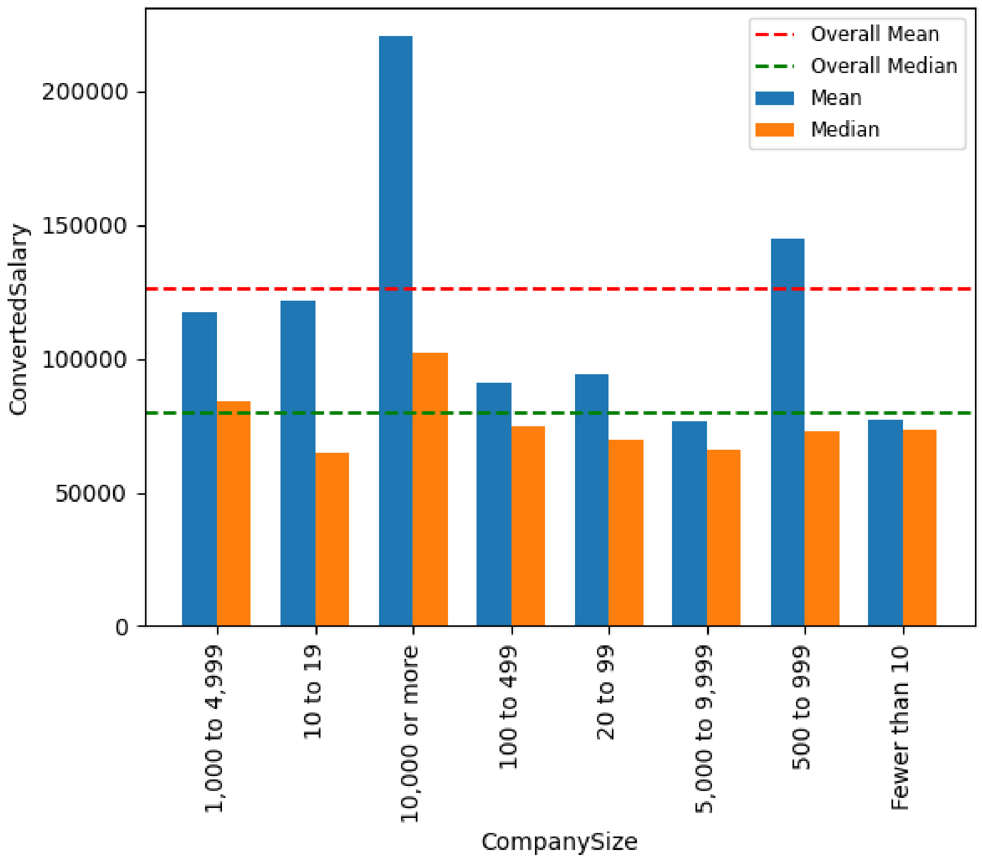
Relationship between formal education attained and annual wages, using the total mean and median as reference. And the median and mean of each value in the field with respect to wages.



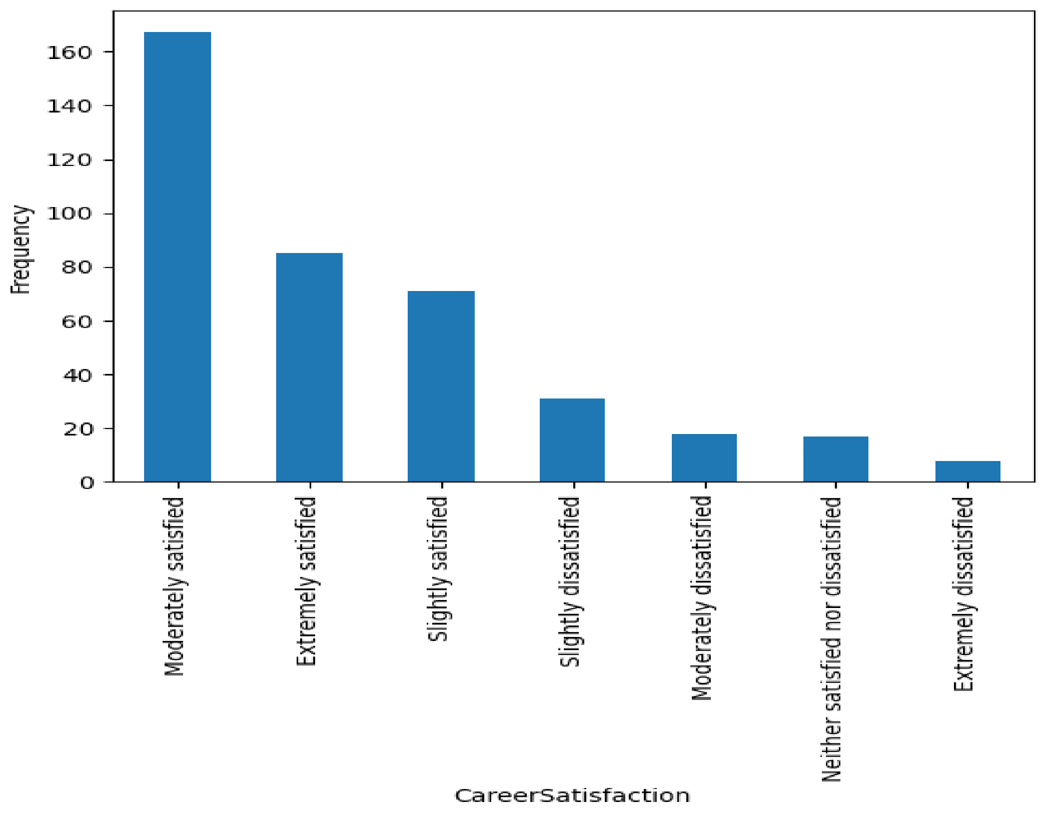
Distribution of the sizes of the companies where the developer collaborates



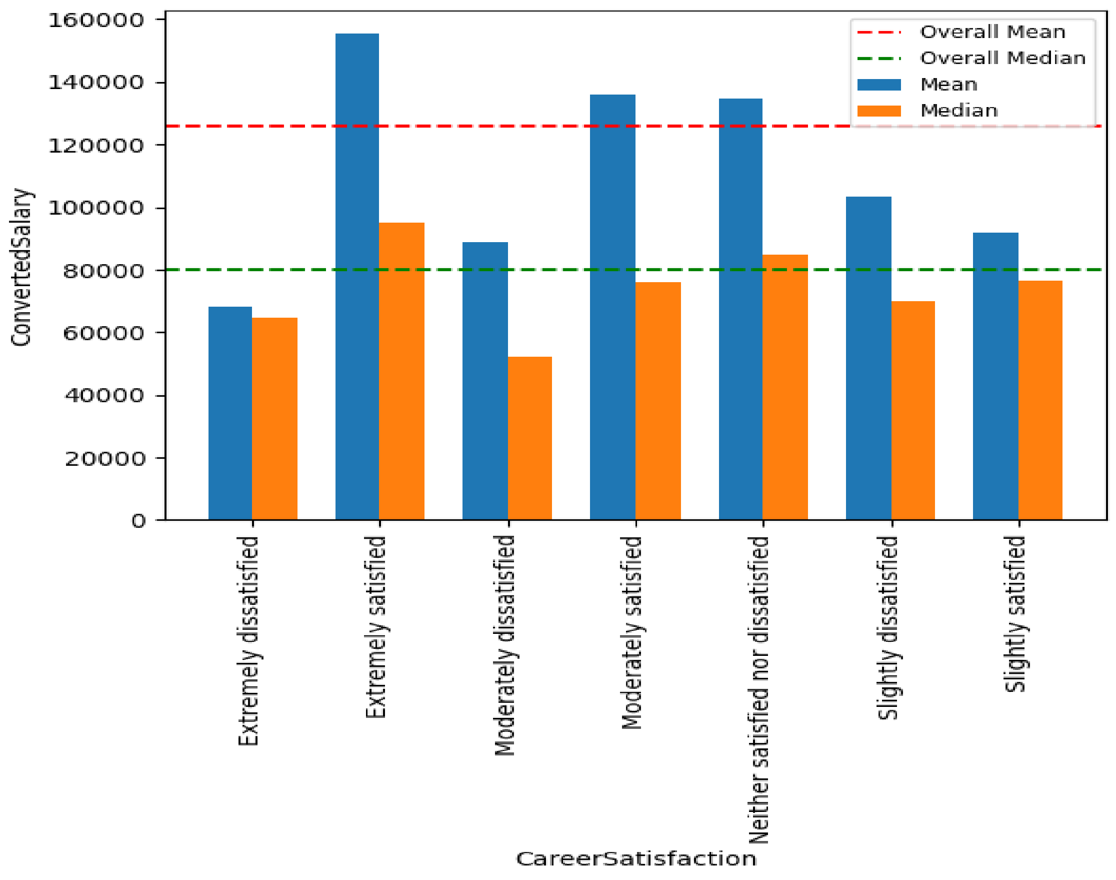
Relationship between company size and salary



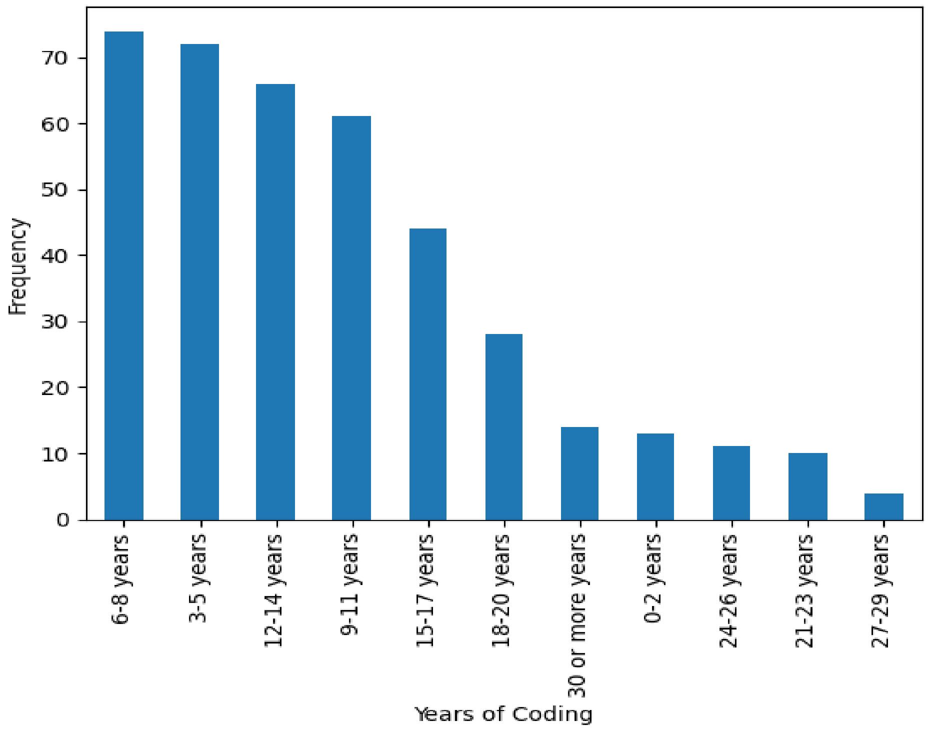
Distribution of career satisfaction



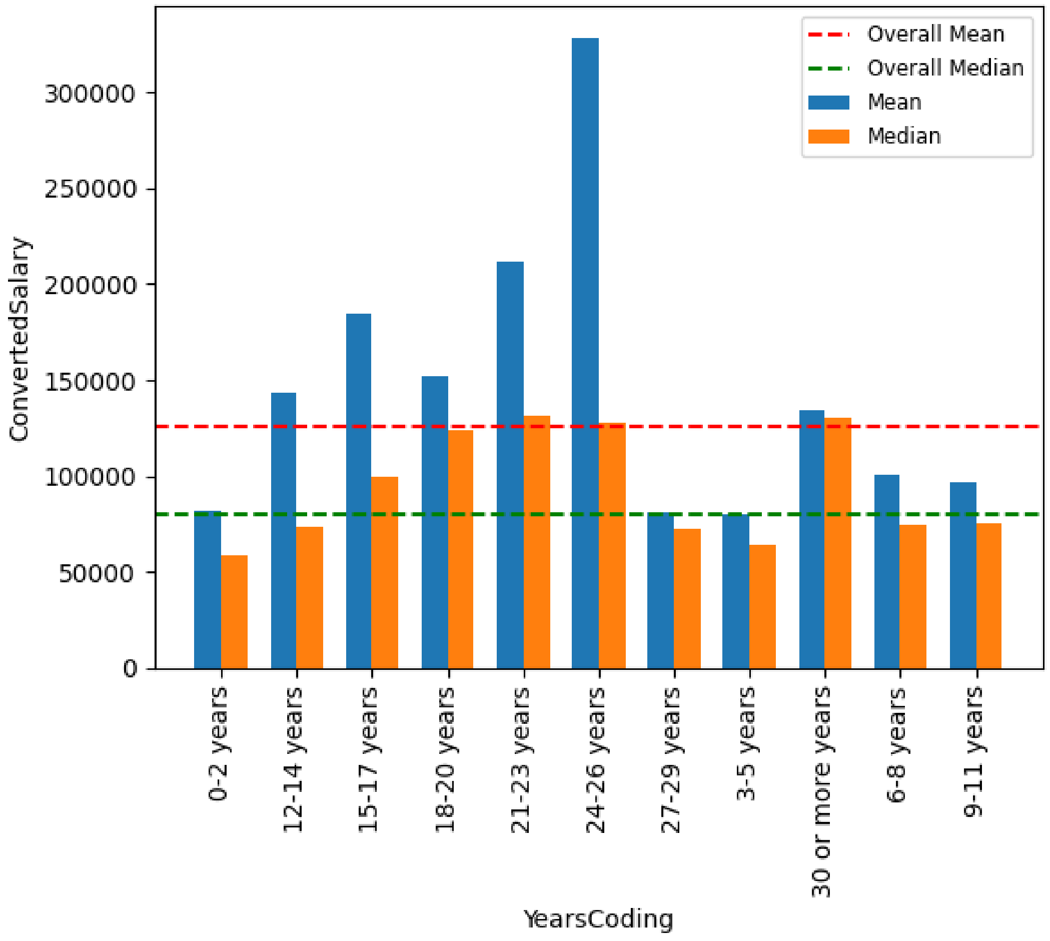
Relationship between satisfaction and salary



Distribution of years of code



Relationship between years hobnobbing and salary



To see the complete code, I invite you to see the analysis.ipynb file at the GitHub link

Thanks to this analysis I can determine some conclusions:

- Those with more years of experience tend to have the best salary.
- Those who are least satisfied with their career tend to be those who earn the least salary and are below both the mean and median.
- It is not so necessary to have complete university studies to earn a salary above the mean and median.
- The largest companies are the ones that pay the best salaries.

Thank you for getting this far. I am available for any questions.