Data Engineering | Creation and configuration of an environment for analysis

Preparing a Data Warehouse with BigQuery for the Autoforce Technical Challenge with *Google Cloud Platform* - GCP.

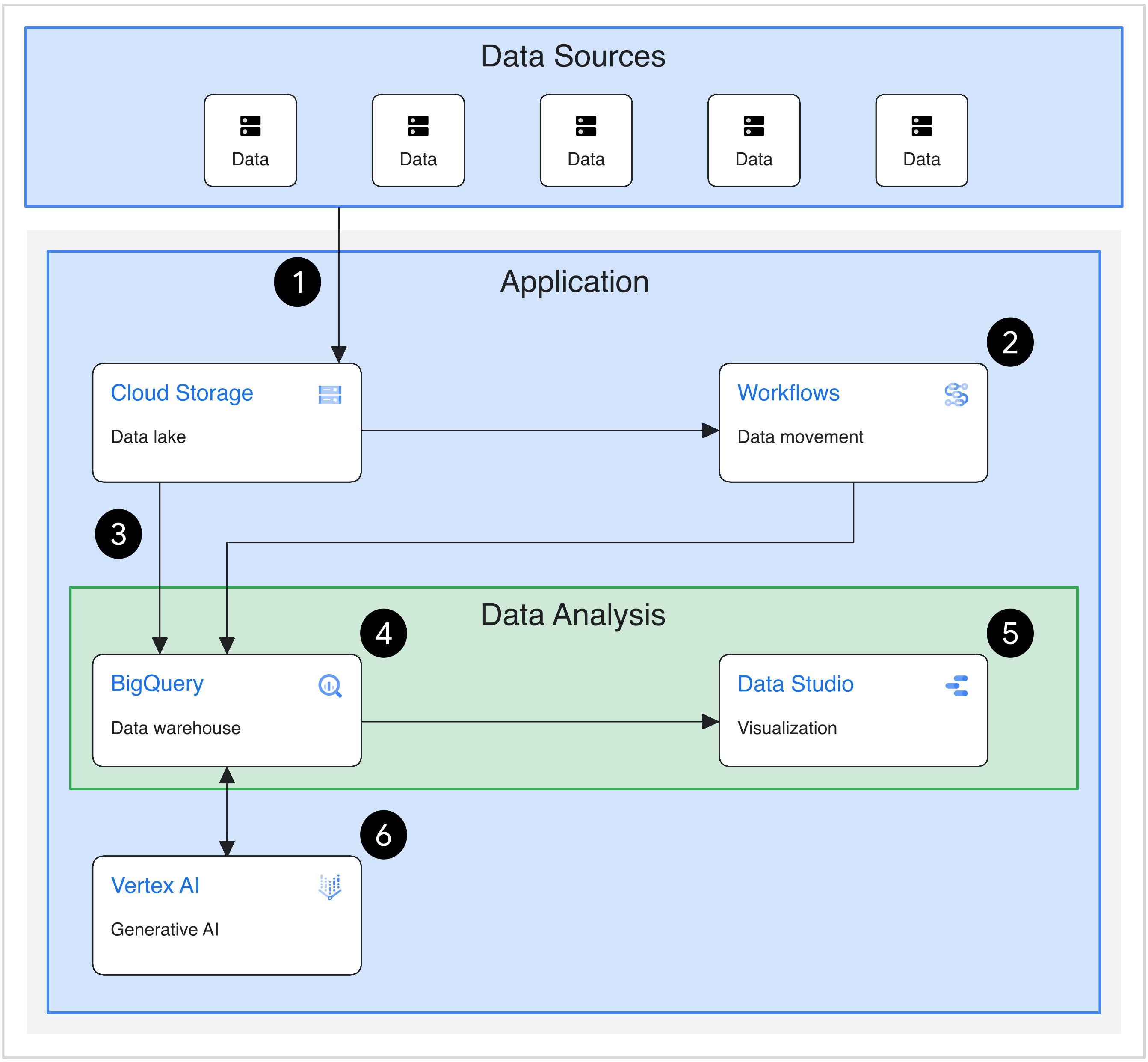
---

1. Tools Used in Data Structure Implementation

1. **Kaggle**:
    - For dataset acquisition.
2. **Google Cloud Storage**:
    - Creating the data lake and storing CSV files.
3. **BigQuery Studio**:
    - Data warehousing and SQL querying.
4. **SQL**:
    - Language for data manipulation in BigQuery.
5. **Looker Studio**:
    - For data visualization and creating BI dashboards.

---

2. Data Lake Preparation and Analysis Using Google Cloud

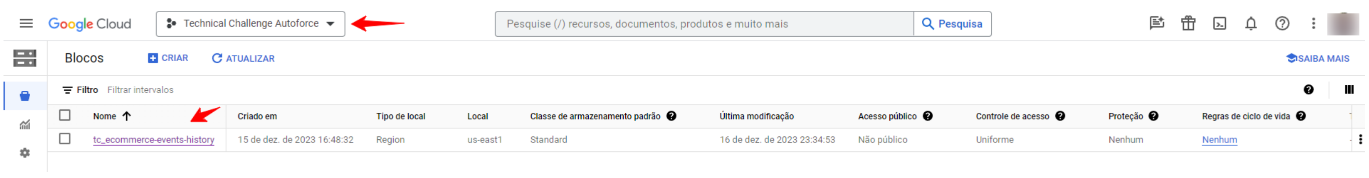


Implementation scheme for the Technical Challenge on GCP.

Data Lake Preparation on Google Cloud Storage

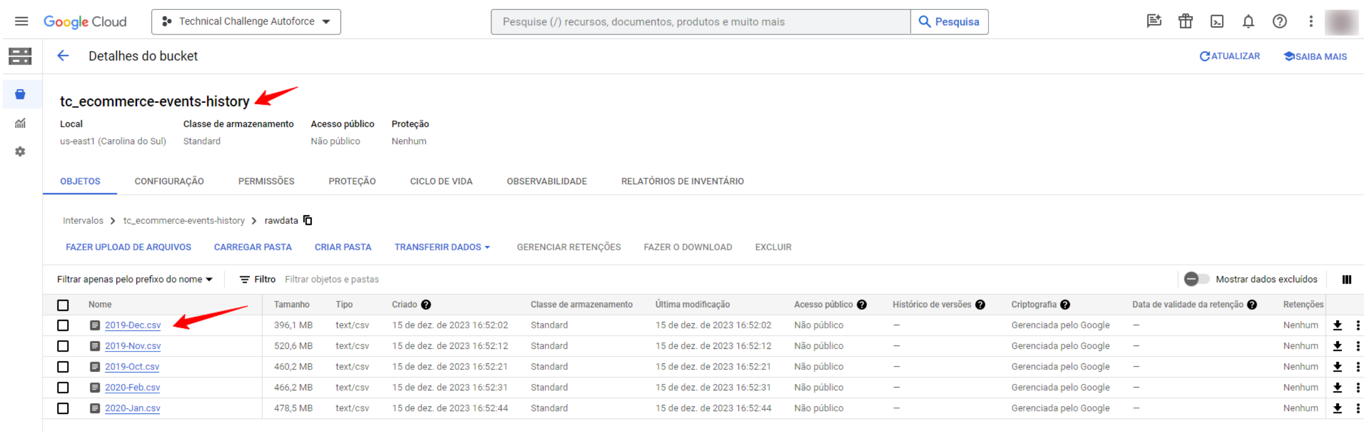
After downloading all the data from Kaggle, a data engineering stage was set up for analysis. The data was then transferred to a **Google Cloud Storage** bucket in Lakehouse, which was created specifically for this challenge. Subsequently, this dataset underwent extensive manipulation and transformation within a **BigQuery Studio** project, aptly named *Technical Challenge Autoforce*.

1. **Data Acquisition**:
    - Downloaded dataset from Kaggle.
    - Dataset includes five CSV files with events data from Oct 2019 to Feb 2020.
2. **Google Cloud Storage Bucket Creation**:
    - Created a bucket `tc_ecommerce-events-history` for data storage.


Google Cloud
Technical Challenge Autoforce
Pesquise (/) recursos, documentos, produtos e muito mais
Pesquisa
Blocos
CRIAR
ATUALIZAR
SAIBA MAIS
Filtro
Filtrar intervalos

Nome	Criado em	Tipo de local	Local	Classe de armazenamento padrão	Última modificação	Acesso público	Controle de acesso	Proteção	Regras de ciclo de vida
tc_ecommerce-events-history	15 de dez. de 2023 16:48:32	Region	us-east1	Standard	16 de dez. de 2023 23:34:53	Não público	Uniforme	Nenhum	Nenhum

3. **Data Upload**:
    - Uploaded the CSV files to the `tc_ecommerce-events-history` bucket.



tc_ecommerce-events-history

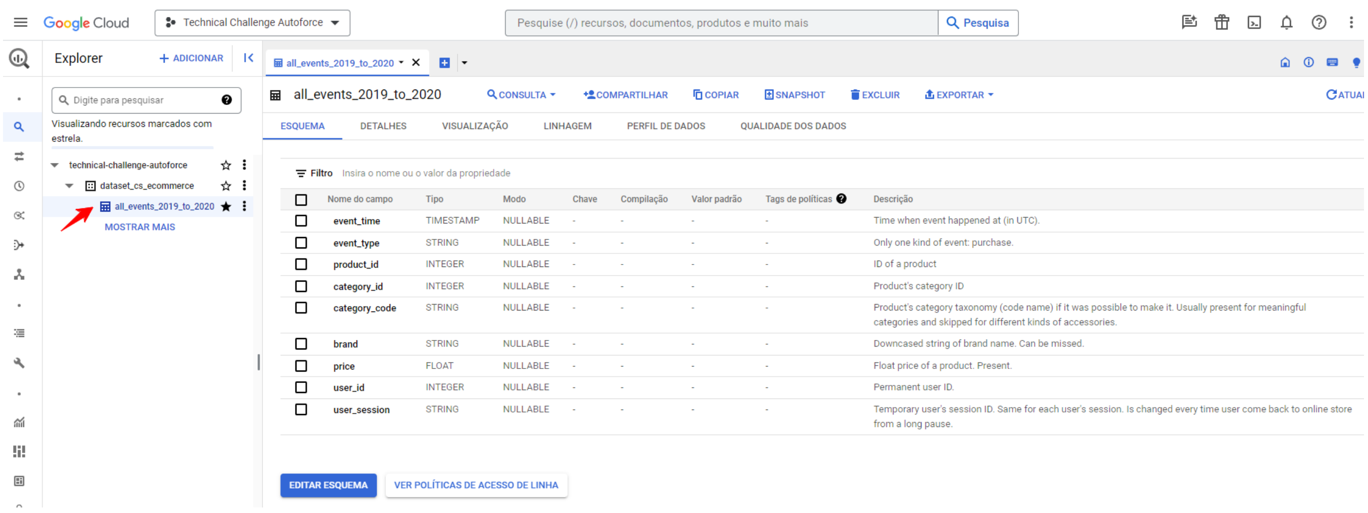
Local	Classe de armazenamento	Acesso público	Proteção
us-east1 (Carolina do Sul)	Standard	Não público	Nenhum

Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público	Histórico de versões	Criptografia	Data de validade da retenção	Retenções
2019-Dec.csv	396,1 MB	text/csv	15 de dez. de 2023 16:52:02	Standard	15 de dez. de 2023 16:52:02	Não público	–	Gerenciada pelo Google	–	Nenhum
2019-Nov.csv	520,6 MB	text/csv	15 de dez. de 2023 16:52:12	Standard	15 de dez. de 2023 16:52:12	Não público	–	Gerenciada pelo Google	–	Nenhum
2019-Oct.csv	460,2 MB	text/csv	15 de dez. de 2023 16:52:21	Standard	15 de dez. de 2023 16:52:21	Não público	–	Gerenciada pelo Google	–	Nenhum
2020-Feb.csv	466,2 MB	text/csv	15 de dez. de 2023 16:52:31	Standard	15 de dez. de 2023 16:52:31	Não público	–	Gerenciada pelo Google	–	Nenhum
2020-Jan.csv	478,5 MB	text/csv	15 de dez. de 2023 16:52:44	Standard	15 de dez. de 2023 16:52:44	Não público	–	Gerenciada pelo Google	–	Nenhum

Data Warehousing and Analysis in BigQuery

Using **Google Cloud** tools and servers is the best option because it combines the resources of a *data lake* and a *data warehouse* into a single platform for storing, processing, and analyzing structured and unstructured data. This integration facilitates efficient storage, processing, and analysis of diverse datasets, encompassing both structured and unstructured data forms. Moreover, **Google Cloud's** seamless interoperability with **Looker Studio** significantly enhances its utility, enabling the creation of sophisticated, insightful dashboards essential for the visualization and interpretation of critical business intelligence.

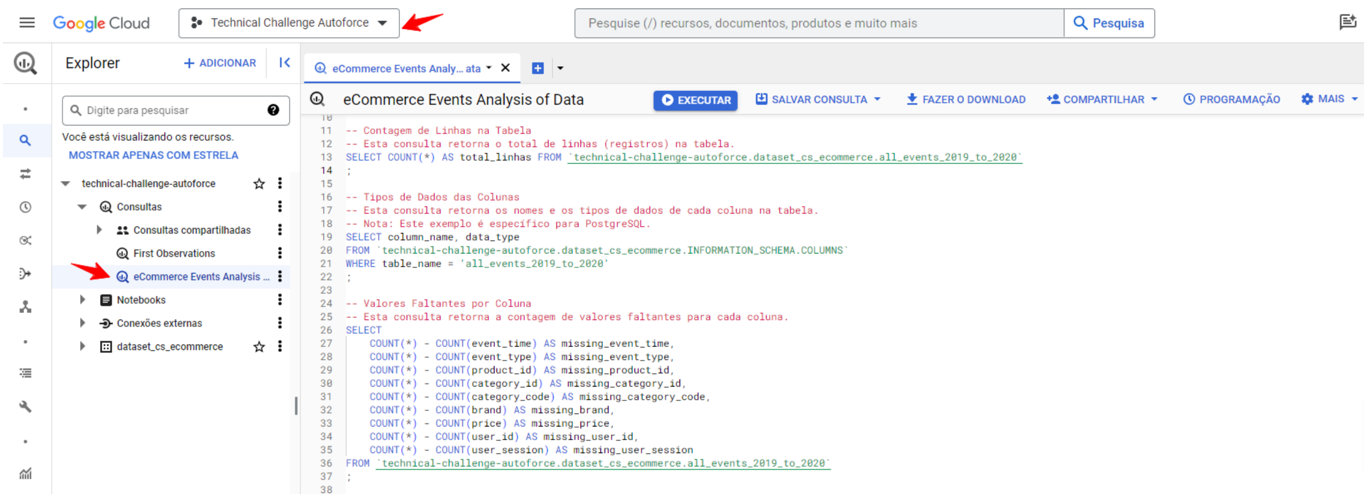
1. **BigQuery Studio Project Setup**:
    - Established the "Technical Challenge Autoforce" project in BigQuery.
2. **Data Import into BigQuery**:
    - Imported CSV files from Google Cloud Storage to BigQuery.
3. **Data Transformation and Unification**:
    - Merged individual tables into a single comprehensive table.



all_events_2019_to_2020

Nome do campo	Tipo	Modo	Chave	Compilação	Valor padrão	Tags de políticas	Descrição
event_time	TIMESTAMP	NULLABLE	-	-	-	-	Time when event happened at (in UTC).
event_type	STRING	NULLABLE	-	-	-	-	Only one kind of event: purchase.
product_id	INTEGER	NULLABLE	-	-	-	-	ID of a product
category_id	INTEGER	NULLABLE	-	-	-	-	Product's category ID
category_code	STRING	NULLABLE	-	-	-	-	Product's category taxonomy (code name) if it was possible to make it. Usually present for meaningful categories and skipped for different kinds of accessories.
brand	STRING	NULLABLE	-	-	-	-	Downcased string of brand name. Can be missed.
price	FLOAT	NULLABLE	-	-	-	-	Float price of a product. Present.
user_id	INTEGER	NULLABLE	-	-	-	-	Permanent user ID.
user_session	STRING	NULLABLE	-	-	-	-	Temporary user's session ID. Same for each user's session. Is changed every time user come back to online store from a long pause.

4. **SQL Queries and Analysis**:
   - Conducted SQL-based analysis for in-depth insights.



eCommerce Events Analysis of Data

```sql
-- Contagem de Linhas na Tabela
-- Esta consulta retorna o total de linhas (registros) na tabela.
SELECT COUNT(*) AS total_linhas FROM `technical-challenge-autoforce.dataset_cs_ecommerce.all_events_2019_to_2020`
;

-- Tipos de Dados das Colunas
-- Esta consulta retorna os nomes e os tipos de dados de cada coluna na tabela.
-- Nota: Este exemplo é específico para PostgreSQL.
SELECT column_name, data_type
FROM `technical-challenge-autoforce.dataset_cs_ecommerce.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'all_events_2019_to_2020'
;

-- Valores Faltantes por Coluna
-- Esta consulta retorna a contagem de valores faltantes para cada coluna.
SELECT
    COUNT(*) - COUNT(event_time) AS missing_event_time,
    COUNT(*) - COUNT(event_type) AS missing_event_type,
    COUNT(*) - COUNT(product_id) AS missing_product_id,
    COUNT(*) - COUNT(category_id) AS missing_category_id,
    COUNT(*) - COUNT(category_code) AS missing_category_code,
    COUNT(*) - COUNT(brand) AS missing_brand,
    COUNT(*) - COUNT(price) AS missing_price,
    COUNT(*) - COUNT(user_id) AS missing_user_id,
    COUNT(*) - COUNT(user_session) AS missing_user_session
FROM `technical-challenge-autoforce.dataset_cs_ecommerce.all_events_2019_to_2020`
;
```

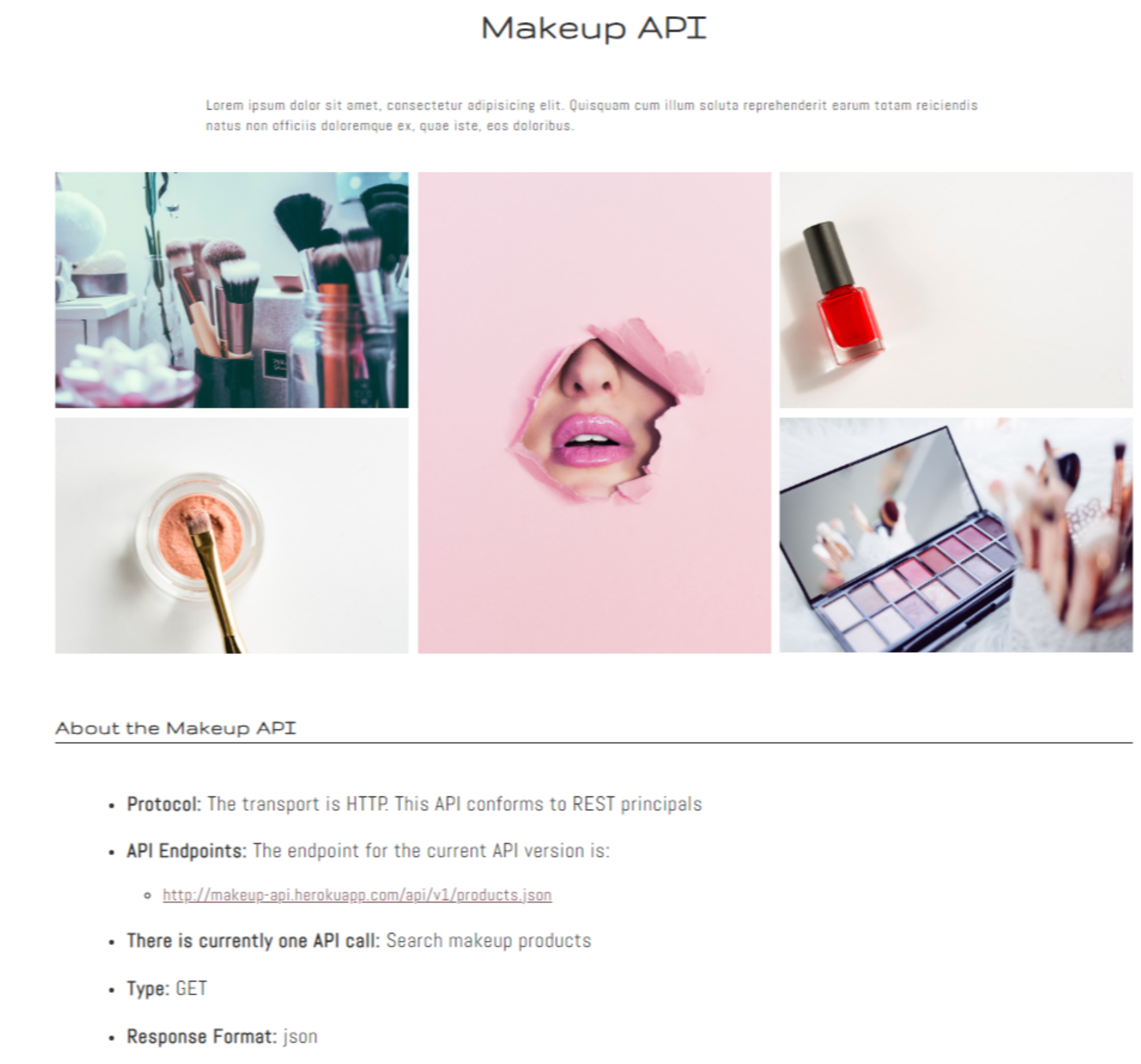
5. **Integration with Colab & Looker Studio**:
   - Leveraged Colab to analysis and Looker Studio for data visualization and BI insights.

---

3. Second part of the Technical Challenge Analysis - New dataset Makeup API

Continuing the Case study with a New Challenge to Technical Analysis:

> "Arbitrarily cross-reference product identifiers with the https://makeup-api.herokuapp.com/ API so that we have data about products and their respective categories."



Makeup API

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam cum illum soluta reprehenderit earum totam reiciendis natus non officiis doloremque ex, quae iste, eos doloribus.

About the Makeup API

- **Protocol:** The transport is HTTP. This API conforms to REST principals
- **API Endpoints:** The endpoint for the current API version is:
  - http://makeup-api.herokuapp.com/api/v1/products.json
- **There is currently one API call:** Search makeup products
- **Type:** GET
- **Response Format:** json

So, to continue the analysis within **Google's BigQuery** environment, we needed to utilize **Google Cloud Function** to extract the bucket we created (`tc_ecommerce-events-history`). The dataset could then be exported to a table, similar to the process used with the other CSV files. And then later the data in the tables can be crossed.

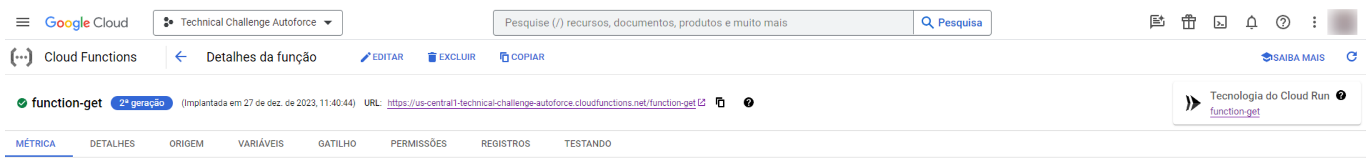
To create an API with **Google Cloud** services that access the endpoint https://makeup-api.herokuapp.com/api/v1/products.json and downloads the complete dataset in CSV format, follow the steps below:

1. **Activation of required APIs on Google Cloud:**

- **Cloud Functions** to create a serverless function that will download the data.
- **Cloud Storage** if you want to store the generated CSV file.

2. **Creating a function in Cloud Functions:**

- Write a Node.js function supported by **Cloud Functions** that sends a GET request to the makeup API endpoint and converts the JSON response to CSV.



function-get 2ª geração (Implantada em 27 de dez. de 2023, 11:40:44) URL: https://us-central1-technical-challenge-autoforce.cloudfunctions.net/function-get

Tecnologia do Cloud Run
function-get

Which was to `main` file:

```
const { Storage } = require('@google-cloud/storage');
const axios = require('axios');
const { Readable } = require('stream');

const storage = new Storage();

exports.downloadMakeupData = async (req, res) => {
  try {
    // Makes request to the makeup API
    const response = await axios.get('https://makeup-
api.herokuapp.com/api/v1/products.json');
    const products = response.data;

    // Convert the response to CSV
    const csvData = convertToCSV(products);

    // Bucket and file name in Cloud Storage
    const bucketName = 'tc_ecommerce-events-history';
    const destinationBlobName = 'source/makeup_data.csv';

    // Bucket and file reference in Cloud Storage
    const bucket = storage.bucket(bucketName);
    const blob = bucket.file(destinationBlobName);

    // Convert CSV string to readable stream
    const csvStream = new Readable();
    csvStream.push(csvData);
    csvStream.push(null);

    // Upload CSV to Cloud Storage
    csvStream.pipe(blob.createWriteStream({
      resumable: false,
      contentType: 'text/csv',
    }))
    .on('error', (error) => {
      res.status(500).send('Error uploading the CSV file: ' + error.message);
    })
    .on('finish', () => {
      res.status(200).send(`Successfully created ${destinationBlobName} in
${bucketName}`);
    });
  } catch (error) {
    res.status(500).send('Error downloading makeup data: ' + error.message);
  }
};
```

```
function convertToCSV(objArray) {
   // Specify the columns you want to include
  const columns = ["id", "brand", "name", "price", "price_sign", "currency",
"rating", "category", "product_type", "tag_list", "created_at", "updated_at"];

   // Create the header with the selected columns
  let str = columns.join(",") + '\r\n';

   // Maps the data to the selected columns and joins with the separator
  return objArray.reduce((accumulator, nextObj) => {
    const row = columns.map(column => {
      let value = nextObj[column] === undefined || nextObj[column] === null ? "" :
nextObj[column].toString();
      if (value.includes(',') || value.includes('"') || value.includes('\n') ||
value.includes('\r')) {
        value = `"${value.replace(/"/g, '""')}"`;
      }
      return value;
    }).join(",");

    return accumulator + row + '\r\n';
  }, str);
}
```

and as `dependencies` file:

```
{
  "dependencies": {
    "@google-cloud/functions-framework": "^3.0.0",
    "@google-cloud/storage": "^7.7.0", // Substitua pelo número da versão mais
recente
    "axios": "^1.6.2" // Substitua pelo número da versão mais recente
  }
}
```

- For this code to work, you needed to install the `@google-cloud` and `axios` dependencies in your **Node.js** project.

In this version of the function in `Node.js`, the CSV file will be created within the `tc_ecommerce-events-history` bucket in the `source` directory with the name `makeup_data.csv`.

3. **Function deployment in Cloud Functions:**

- The function was implemented using the **Google Cloud** console or the CLI (*gcloud*).

4. **Call the created function:**

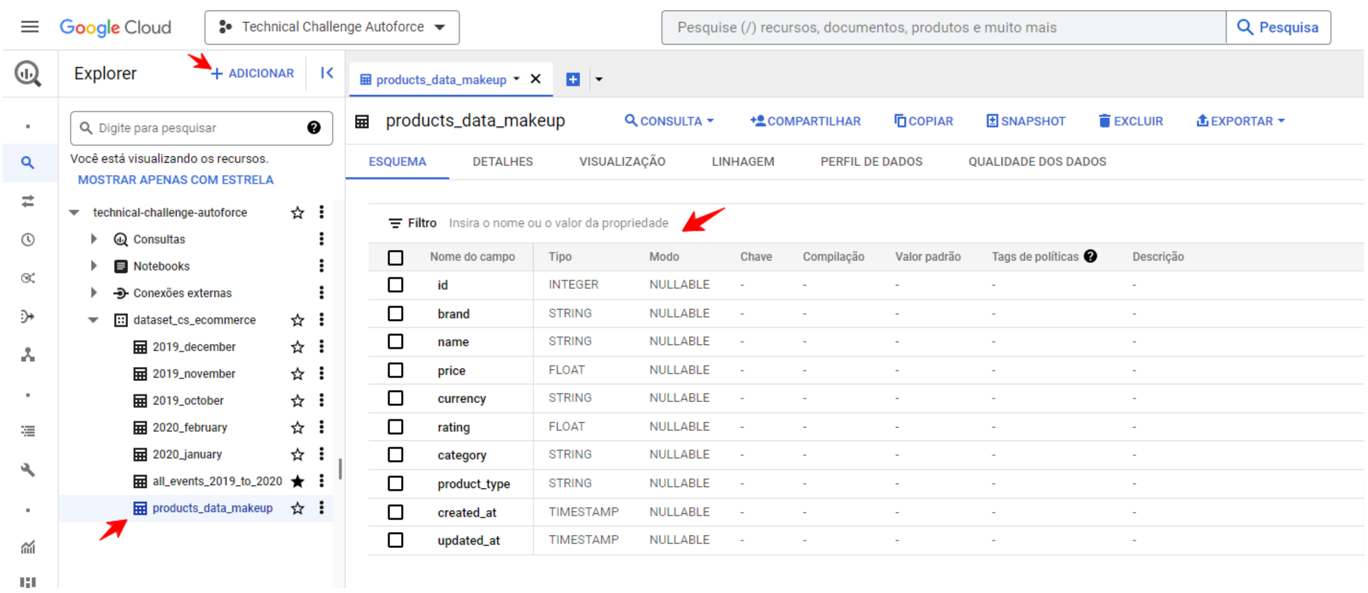
- A HTTP request was sent to the URL created by **Cloud Functions** in order to execute the function and generate the CSV file.

5. **CSV file access:**

- After executing the function, the CSV file was accessible in the designated **Cloud Storage** bucket.

6. **Data Import into BigQuery**

- Imported new makeup CSV file from Google Cloud Storage to BigQuery.



Nome do campo	Tipo	Modo	Chave	Compilação	Valor padrão	Tags de políticas	Descrição
id	INTEGER	NULLABLE	-	-	-	-	-
brand	STRING	NULLABLE	-	-	-	-	-
name	STRING	NULLABLE	-	-	-	-	-
price	FLOAT	NULLABLE	-	-	-	-	-
currency	STRING	NULLABLE	-	-	-	-	-
rating	FLOAT	NULLABLE	-	-	-	-	-
category	STRING	NULLABLE	-	-	-	-	-
product_type	STRING	NULLABLE	-	-	-	-	-
created_at	TIMESTAMP	NULLABLE	-	-	-	-	-
updated_at	TIMESTAMP	NULLABLE	-	-	-	-	-

7. **SQL Queries and Analysis on Colab**:
   - Conducted new analysis on product categories as requested on **Google Colab** for in-depth insights.

Benefits of Using Google Cloud Tools

- Robust solution for handling structured and unstructured data.
- Efficient data processing and analysis capabilities.
- Effective visualization and BI insight creation with Looker Studio.