

Prueba Tecnica - MVM Global

DESAFIO # 1: Construya un script que genere de forma automática los datos de: departamentos, puestos de trabajo, y empleados.

```
In [1]: import pandas as pd
import numpy as np
from datetime import datetime

# Departamentos Data
departamentos_data = {
    "depto_id": [1, 2, 3, 4, 5],
    "depto_nombre": ["IT", "Recursos Humanos", "Finanzas", "Marketing", "Producción"]
}

# Puestos de Trabajo Data
puestos_data = {
    "puesto_id": [101, 102, 103, 104, 105],
    "puesto_nombre": ["Analista", "Gerente", "Director", "Asistente", "Técnico"],
    "depto_id": [1, 2, 2, 3, 4] # Ejemplo de asignación de puestos a departamentos
}

# Empleados Data
np.random.seed(0) # Para reproducibilidad

n_empleados = 100 # Cantidad de empleados a generar
nombres = ["Nombre" + str(i) for i in range(1, n_empleados + 1)]
apellidos = ["Apellido" + str(i) for i in range(1, n_empleados + 1)]

empleados_data = {
    "empleado_id": np.arange(1, n_empleados + 1),
    "nombre": nombres,
    "apellido": apellidos,
    "depto_id": np.random.choice([1, 2, 3, 4, 5], n_empleados),
    "puesto_id": np.random.choice([101, 102, 103, 104, 105], n_empleados),
    "fecha_contratacion": [datetime(2020, np.random.randint(1, 13), np.random.randi
```

	depto_id	depto_nombre
0	1	IT
1	2	Recursos Humanos
2	3	Finanzas
3	4	Marketing
4	5	Producción

	puesto_id	puesto_nombre	depto_id
0	101	Analista	1
1	102	Gerente	2
2	103	Director	2
3	104	Asistente	3
4	105	Técnico	4

	empleado_id	nombre	apellido	depto_id	puesto_id	fecha_contratacion
0	1	Nombre1	Apellido1	5	103	2020-10-24
1	2	Nombre2	Apellido2	1	104	2020-06-20
2	3	Nombre3	Apellido3	4	103	2020-05-22
3	4	Nombre4	Apellido4	4	102	2020-04-04
4	5	Nombre5	Apellido5	4	103	2020-08-10

```
In [10]: # Guardar en CSV
departamentos_df.to_csv('departamentos.csv', index=False)
puestos_df.to_csv('puestos.csv', index=False)
empleados_df.to_csv('empleados.csv', index=False)
```

¿Por qué elegir CSV?

Compatibilidad universal: Prácticamente cualquier sistema puede leer y escribir archivos CSV.
 Facilidad de uso: Los archivos CSV son fáciles de entender y manipular, incluso con editores de texto simples.
 Interoperabilidad: Ideal para intercambiar datos entre diferentes aplicaciones.

```
In [13]: import gspread
from google.oauth2.service_account import Credentials as ServiceAccountCredentials
from google.cloud import storage
import json

base_auto = {
    "type": "service_account",
    "project_id": "planar-abbey-417215",
    "private_key_id": "500d97444da1948148a30f480a55e4a198f79173",
    "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCCKgw",
    "client_email": "johnbyron93@planar-abbey-417215.iam.gserviceaccount.com",
    "client_id": "113262839633462296164",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
    "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/johnby",
    "universe_domain": "googleapis.com"
}

# Para gspread
gspread_credentials = ServiceAccountCredentials.from_service_account_info(base_auto)
gc = gspread.authorize(gspread_credentials)
```

```
# Para google.cloud.storage
storage_credentials = ServiceAccountCredentials.from_service_account_info(base_auto
storage_client = storage.Client(credentials=storage_credentials)
```

```
In [ ]: from google.cloud import storage
from google.oauth2 import service_account
import json

# Carga tus credenciales de cuenta de servicio desde un diccionario o archivo JSON
credenciales_dict = {
    "type": "service_account",
    "project_id": "planar-abbey-417215",
    "private_key_id": "500d97444da1948148a30f480a55e4a198f79173",
    "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAKgw",
    "client_email": "johnbyron93@planar-abbey-417215.iam.gserviceaccount.com",
    "client_id": "113262839633462296164",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
    "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/johnby",
    "universe_domain": "googleapis.com"
}

credentials = service_account.Credentials.from_service_account_info(credenciales_dict)

# Crea el cliente de Google Cloud Storage con tus credenciales
storage_client = storage.Client(credentials=credentials, project=credenciales_dict["project_id"])

def subir_archivo_a_gcs(bucket_name, source_file_path, destination_blob_name):
    """Sube un archivo al bucket de Google Cloud Storage especificado."""
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_path)

    print(f"Archivo {source_file_path} subido a {destination_blob_name} en el bucket {bucket_name}")

# Parámetros para la función
bucket_name = 'bucket-1_entrevista'
archivo_csv = 'empleados.csv'
destination_blob_name = '2024/03/empleados.csv'

# Llama a la función para subir el archivo
subir_archivo_a_gcs(bucket_name, archivo_csv, destination_blob_name)
```

El código anterior hace uso de la generación de las credenciales necesarias para acceder a los servicios de Google Cloud. En el presente desafío se hace uso de los mismos para enviar los archivos CSV resultantes a la nube de Google mediante la creación de un bucket que los contiene y guardará para posteriores análisis y conversiones. También las credenciales nos servirán para los desafíos futuros.

DESAFIO 3: Implemente un proceso batch para migrar los datos a una base de datos SQL/NoSQL, o si lo desea, a un Datawarehouse o bucket analítico de un Datalake.

```
In [ ]: pip install google-cloud-bigquery
```

```
In [33]: from google.cloud import bigquery
import pandas as pd
import os

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

# Función para cargar un archivo CSV a BigQuery
def cargar_csv_a_bigquery(nombre_archivo_csv, nombre_tabla_completo):
    dataset_id, table_id = nombre_tabla_completo.split('.')[0:-2:]
    dataset_ref = client.dataset(dataset_id)
    table_ref = dataset_ref.table(table_id)

    job_config = bigquery.LoadJobConfig()
    job_config.source_format = bigquery.SourceFormat.CSV
    job_config.skip_leading_rows = 1 # Salta la cabecera del archivo CSV
    job_config.autodetect = True # Detecta el esquema automáticamente

    with open(nombre_archivo_csv, "rb") as source_file:
        job = client.load_table_from_file(source_file, table_ref, job_config=job_co

    job.result() # Espera a que la carga se complete
    print(f"Tabla {nombre_tabla_completo} cargada con éxito.")

# Función para guardar DataFrame como CSV y cargarlo a BigQuery
def dataframe_a_bigquery(df, nombre_tabla_completo):
    # Guarda el DataFrame como un archivo CSV temporal
    nombre_archivo_csv = "temp.csv"
    df.to_csv(nombre_archivo_csv, index=False)

    # Carga el archivo CSV a BigQuery
    cargar_csv_a_bigquery(nombre_archivo_csv, nombre_tabla_completo)

    # Elimina el archivo CSV temporal
    os.remove(nombre_archivo_csv)

# Nombres completos de las tablas en BigQuery
nombre_tabla_departamentos = 'My First Project.entrevista_datos.tabla_departamentos'
nombre_tabla_puestos = 'My First Project.entrevista_datos.tabla_puestos'
nombre_tabla_empleados = 'My First Project.entrevista_datos.tabla_empleados'

# Cargar DataFrames a BigQuery
dataframe_a_bigquery(departamentos_df, nombre_tabla_departamentos)
dataframe_a_bigquery(puestos_df, nombre_tabla_puestos)
dataframe_a_bigquery(empleados_df, nombre_tabla_empleados)

print("Carga completada.")
```

Tabla My First Project.entrevista_datos.tabla_departamentos cargada con éxito.
 Tabla My First Project.entrevista_datos.tabla_puestos cargada con éxito.
 Tabla My First Project.entrevista_datos.tabla_empleados cargada con éxito.
 Carga completada.

El código anterior hace uso de BigQuery, perteneciente a GCS, para poder convertir los archivos CSV en tablas en dicha nube, para su posterior y permanente manipulación.

DESAFIO 4: Dependiendo si escoge una base de datos SQL/NoSQL, un Datawarehouse, o un Datalake, entonces desarrolle una view/query/report a partir del modelo de datos.

Este desafío se encuentra realizado en el PDF adjunto a los archivos enviados para la presente prueba, junto con sus archivos soporte SQL.

DESAFIO 5: Desarrolle una API REST para consultar la view/query/report. Para el desarrollo de la API considere algún framework de Python, C#/.Net.

```
In [ ]: pip install flask-ngrok
```

```
In [48]: from flask import Flask, jsonify
from flask_ngrok import run_with_ngrok
from google.cloud import bigquery
from google.oauth2 import service_account

app = Flask(__name__)
run_with_ngrok(app) # Inicia ngrok cuando la app esté en ejecución

# Cliente de BigQuery configurado con tus credenciales
client = bigquery.Client(credentials=credentials, project=credentials.project_id)

@app.route('/')
def index():
    return 'El servidor está funcionando correctamente. añadele /report a la url pr

@app.route('/report', methods=['GET'])
def get_report():
    query = """
        SELECT * FROM `planar-abbey-417215.entrevista_datos.vista_empleados_detalle
        """
    try:
        query_job = client.query(query)
        results = query_job.result() # Espera a que los resultados estén disponibl

        # Convertir los resultados a un formato JSON serializable
        rows = [dict(row) for row in results]
        print(rows) # Debugging: Imprime los resultados crudos
        return jsonify(rows)
    except Exception as e:
        print(e) # Debugging: Imprime el error si ocurre alguno
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run()
```

```
* Serving Flask app '__main__' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
```

```

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Mar/2024 00:39:33] "GET / HTTP/1.1" 200 -
Exception in thread Thread-31:
Traceback (most recent call last):
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connection.py", line 174, in _new_conn
    conn = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\util\connection.py", line 95, in create_connection
    raise err
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\util\connection.py", line 85, in create_connection
    sock.connect(sa)
ConnectionRefusedError: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connectionpool.py", line 703, in urlopen
    httplib_response = self._make_request(
                       ^^^^^^^^^^^^^^^^^
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connectionpool.py", line 398, in _make_request
    conn.request(method, url, **httplib_request_kw)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connection.py", line 239, in request
    super(HTTPConnection, self).request(method, url, body=body, headers=headers)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 1283, in request
    self._send_request(method, url, body, headers, encode_chunked)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 1329, in _send_request
    self.endheaders(body, encode_chunked=encode_chunked)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 1278, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 1038, in _send_output
    self.send(msg)
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\http\client.py", line 976, in send
    self.connect()
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connection.py", line 205, in connect
    conn = self._new_conn()
           ^^^^^^^^^^^^^^^^^
  File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connection.py", line 186, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x000001ABE150EED0>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

```
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\adapters.py", line 440, in send
    resp = conn.urlopen(
            ^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\connectionpool.py", line 787, in urlopen
    retries = retries.increment(
            ^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\urllib3\util\retry.py", line 592, in increment
    raise MaxRetryError(_pool, url, error or ResponseError(cause))
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost', port=4040): Max retries exceeded with url: /api/tunnels (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001ABE150EED0>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión'))
```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

```
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\threading.py", line 1038, in _bootstrap_inner
    self.run()
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\threading.py", line 1394, in run
    self.function(*self.args, **self.kwargs)
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask_ngrok.py", line 70, in start_ngrok
    ngrok_address = _run_ngrok()
                    ^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\flask_ngrok.py", line 35, in _run_ngrok
    tunnel_url = requests.get(localhost_url).text # Get the tunnel information
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\api.py", line 75, in get
    return request('get', url, params=params, **kwargs)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\api.py", line 61, in request
    return session.request(method=method, url=url, **kwargs)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\sessions.py", line 529, in request
    resp = self.send(prepared_request, **send_kwargs)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\sessions.py", line 645, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Users\Byron\AppData\Local\Programs\Python\Python311\Lib\site-packages\requests\adapters.py", line 519, in send
```



```
raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=4040): Max retries exceeded with url: /api/tunnels (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001ABE150EED0>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión'))
127.0.0.1 - - [15/Mar/2024 00:40:47] "GET /report HTTP/1.1" 200 -
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Emp
leado': 'Nombre96', 'Apellido_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recur
sos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido
_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puest
o': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido_Empleado': 'Apellido96',
'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Emp
leado': 'Nombre96', 'Apellido_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recur
sos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido
_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puest
o': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido_Empleado': 'Apellido96',
'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Emp
leado': 'Nombre96', 'Apellido_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recur
sos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido
_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puest
o': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido_Empleado': 'Apellido96',
'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Emp
leado': 'Nombre96', 'Apellido_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recur
sos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido
_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puest
o': 'Técnico'}, {'Nombre_Empleado': 'Nombre96', 'Apellido_Empleado': 'Apellido96',
'Nombre_Departamento': 'Recursos Humanos', 'Nombre_Puesto': 'Técnico'}, {'Nombre_Emp
leado': 'Nombre96', 'Apellido_Empleado': 'Apellido96', 'Nombre_Departamento': 'Recur
sos Humanos', 'Nombre_Puesto': 'Técnico'}]
```

El código anterior hace uso de Flask, librería creada para desplegar código en servicios HTTP y así generar solicitudes get y post a la misma, como se puede ver en el resultado, el algoritmo fue exitoso en generar y desplegar la información para poder ser consultada a través de una view o vista en formato objeto JSON.

DESAFIO # 6: Mejore la implementación de la API realizando un despliegue que use contenedores (valide las distintas opciones que le brinda su nube). Considere una prueba de consumo a la API implementando o activando algún front de acceso para ejecutar la invocación a la view/query/report.

In []: Para este desafío fue creado un documento PDF, el cual recoge toda la información p