Proyecto de ETL - 3da Parte

Kevin Santiago Artunduaga Vivas

Universidad Autónoma de Occidente

Cali, Colombia

Para la entrega final del proyecto se especificó acomodar el modelo dimensional, en donde se enviarán los conjuntos de datos con herramientas de extracción, transformación y carga (ETL). A su vez envió datos en tiempo real como Apache Airflow y Apache Kafka. Para ello se seleccionó el conjunto de datos de los trabajos para ofrecer información de la calificación del trabajo y el título del trabajo. Ahora bien para el procesado de todas las dags se arrancó como se inició el proyecto con la base de datos grande y ya luego se fue arreglando el modelado dimensional.

Airflow DAG

El proceso de las DAG´s arrancó con el extract dataset y el extract API, para el extract dataset como se había hecho anteriormente primero se procesa este a un csv de pandas en donde fuera realizable su manipulación, para ya posteriormente montar todas las columnas a la base de datos, para la API se hizo uso de un csv que se realizó en la segunda entrega, este csv fue creado a partir de peticiones a la API, estas API por motivos de límite de consultas tenia un limite, asi que decidi incrustar todas esas peticiones en un csv. Este se extrajo y se subió en la base de datos en una tabla llamada ‘datajobs_salaries’.

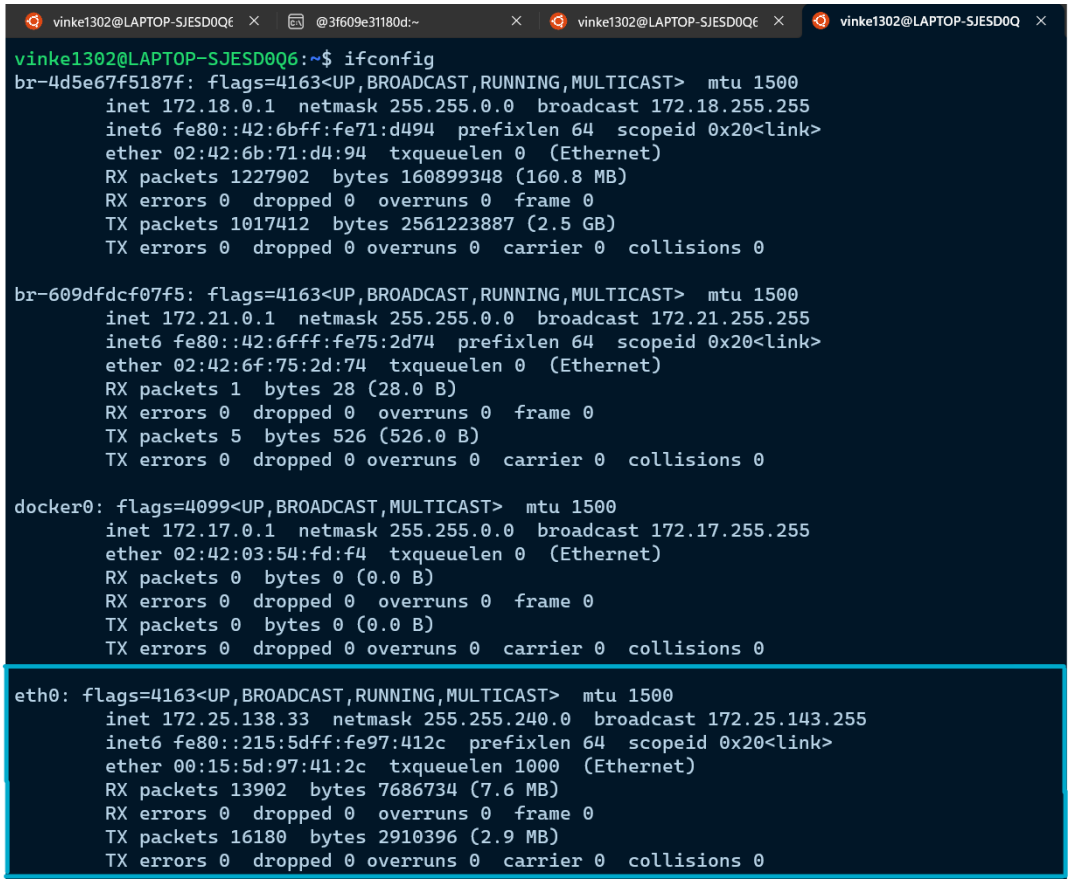
Luego para el proceso de transform dataset y transform API, en el dataset se fueron realizando todas las transformaciones que ya se habían realizado en la primera entrega, en donde se hicieron cada una de las agrupaciones de los trabajos a partir de grupos en donde su zona coincidiera, también se hizo la eliminación de columnas que no iban a ser usadas debido a la cantidad de columnas exageradas con las que se contaba, cabe recalcar que para las transformaciones de la API solo fue eliminada una columna la de publisher link ya que esta no era necesaria.

Ahora bien al pasar al proceso de merge como no se realizó ninguna combinación de tablas aproveche para hacer el modelado dimensional, así que se dejaron 6 dimensiones creadas y la tabla de hechos llamada ‘datajobs_glassdoor’, para las dimensiones la primera dimensión que se creó fue la de la API, llamada datajobs_salaries ésta incluye toda la información de los salarios el mínimo, el promedio, el máximo, el periodo de pago y el tipo de moneda. Luego a partir de las tablas con las cuales venía el dataset principal se crearon 2 dimensiones estas fueron ‘datajobs_benefitshighlights’ y ‘datajobs_salarystats’, la primera tiene que ver con todos los comentarios destacados a partir del trabajo cosas muy de texto, y ya para la de salary stats se manejaban todas las estadísticas en donde estaban los percentiles de cada uno de los títulos de trabajo.

Posteriormente están las 2 dimensiones que se sacaron por medio de la base de datos principal primero está la de ‘datajobs_company’ y ‘datajobs_job’, en la de company se metió toda la información referenciada a la compañía o industria del trabajo, tales como fecha en la que se fundo, la zona en la que se encuentra, el tipo de industria, los ingresos de esta, el tamaño de empleados, el tipo de la compañía, su descripción, el link de la página y el nombre de la empresa.

Al mismo tiempo la de ‘datajobs_job’ esta toma toda la información relevante del trabajo como el título, la fecha en la que se posteo el trabajo, su descripción, el recurso de donde se saco el trabajo, la localización, su latitud, su longitud el país del trabajo, su rating, la agrupación del trabajo y la industria a la que pertenece.

En el proceso de load no se hace nada importante simplemente culmine el proceso hasta ahí y ya con eso pude realizar la conexión de la base de datos a power bi, para ello tuve que tomar la dirección de la IP de la máquina de wsl la eth0, como se puede apreciar al hacer uso del comando de ifconfig, la que puse para la conexión fue la 172.17.0.1. Mediante ella se pudo efectuar la conexión.

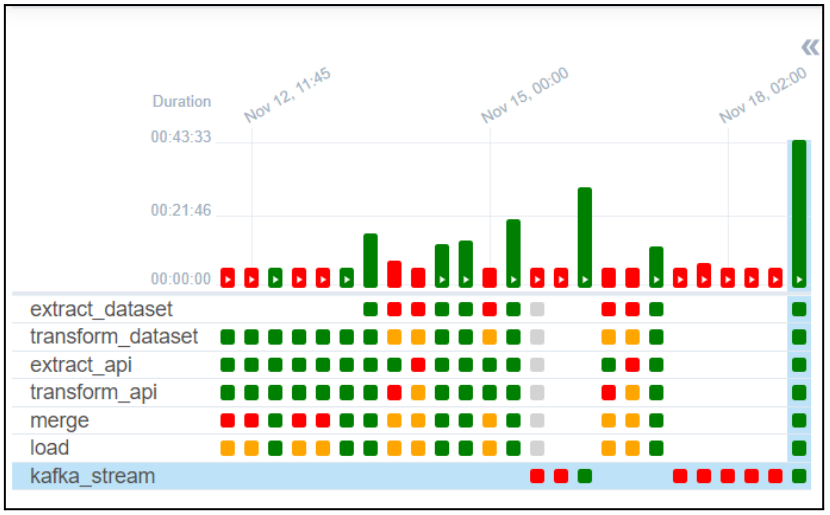


Ya para finalizar hice el proceso de kafka stream, en este se hace inicio del kafka producer para hacer envío de los mensajes, y mandar las características al consumer, cabe recalcar que no hice envío de todo el dataset sino que quise enfocarme únicamente en la calificación de los trabajos, para ello hice envío las columnas de job_discoverdate, header_jobtitle y rating_starrating, todas estas

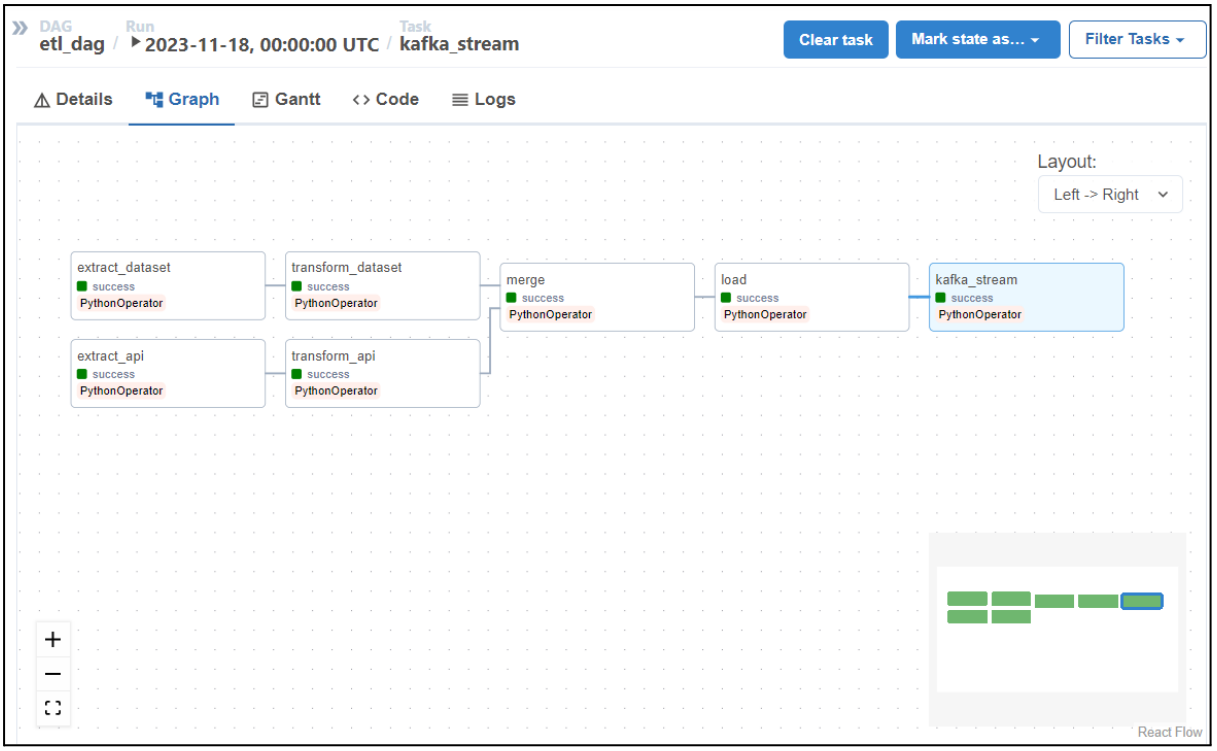
columnas hacer parte de la dimensión de ‘datajobs_job’. De esta manera quedó distribuido el código del producer para entenderlo de una mejor manera.

```
def query():
    conn = conn_postgresql()
    df = pd.read_sql_query("SELECT header_jobtitle,
job_discoverdate, rating_starrating FROM datajobs_job", conn)
    print(df)
    return df

def kafka_producer():
    producer = KafkaProducer(
        bootstrap_servers=['localhost:9092'],
        value_serializer=lambda v: json.dumps(v).encode('utf-8')
    )
    df = query()
    for index, row in df.iterrows():
        message = {
            "header_jobtitle": row["header_jobtitle"],
            "job_discoverdate": str(row["job_discoverdate"]),
            "rating_starrating": row["rating_starrating"]
        }
        producer.send('project', value=message)
        time.sleep(0.01)

    producer.close()
```

Para visualizar que cada una de las dags se efectuó correctamente podemos ver que en el último proceso todas se encuentran como success.



Y la pipeline de los dags.



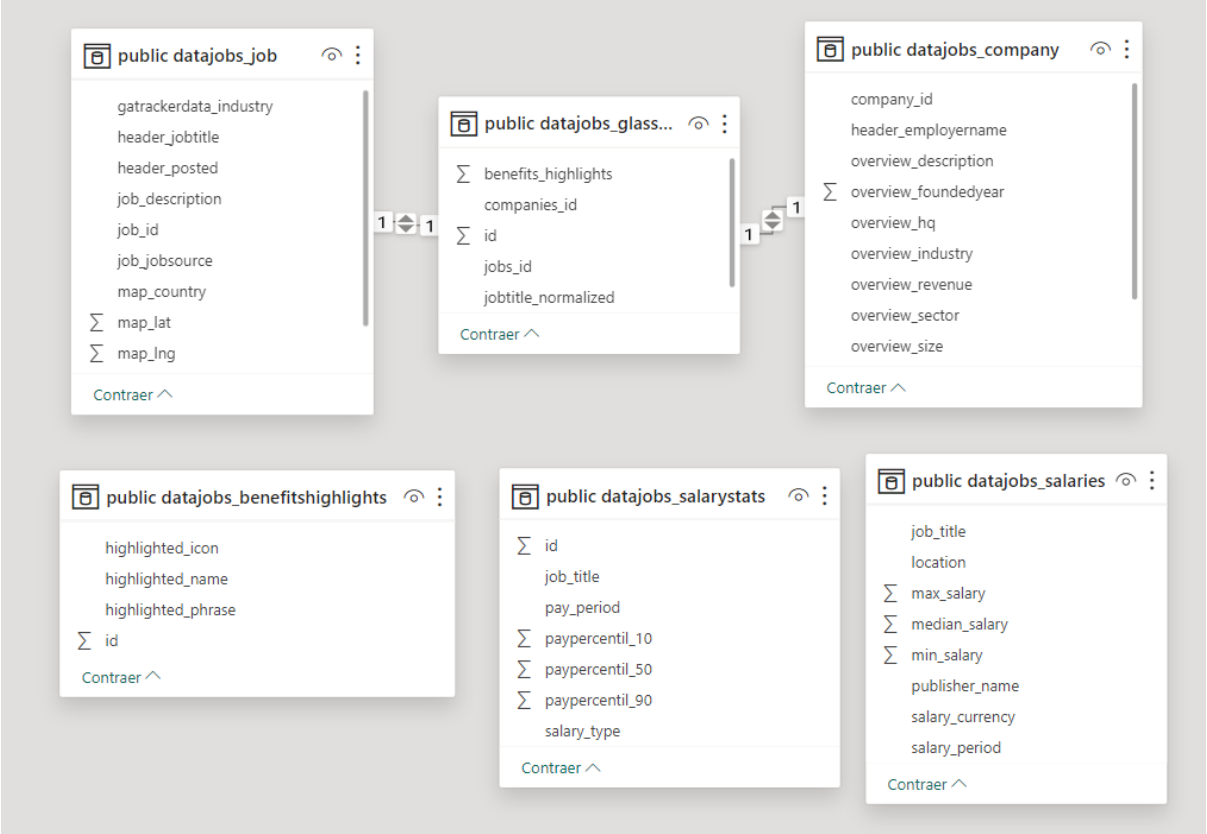
Kafka Stream

En cuanto al proceso de streaming como ya mencione anteriormente el producer hace parte como una de las dags pero el consumer si se inicia por a parte, para el hago uso de la API de power BI en donde se hacen envío de las columnas que mencione que iba hacer uso, asi que para comenzar se inicia el consumer para que comience a escuchar al producer, el producer en cuanto inicia el proceso de envío de mensajes comienza todo el proceso de stream, cabe recalcar que para que el funcionamiento de streaming se efectúe de manera correcta hay que hacer la creación del topic llamado project, este topic es que hace uso kafka para realizar la conexión.

En cuanto al envío de mensajes al poner a kafka a escuchar el topic se envían cada una de las columnas que hacer parte del dataset y el consumer hago que printee la fecha porque para la fecha tuve que hacer unos cambios debido a que en power bi sol se acepta un formato específico para este, asi que gracias al consumer efectuó este cambio y se hace posible el envío de esta columna transformada. Aquí muestro como al iniciar el producer se comienzan a enviar las fechas ya con el formato especificado para power BI.

```
vinke1302@LAPTOP-SJESD0Q6 | @3f609e31180d:~ | vinke1302@LAPTOP-SJESD0Q
(venv) vinke1302@LAPTOP-SJESD0Q6:~/Apache/dags$ python3 kafka_consumer.py
0    2019-10-24T04:52:23.000000Z
Name: job_discoverdate, dtype: object
0    2019-10-21T08:16:53.000000Z
Name: job_discoverdate, dtype: object
0    2019-11-01T02:33:20.000000Z
Name: job_discoverdate, dtype: object
0    2019-10-19T11:57:10.000000Z
Name: job_discoverdate, dtype: object
0    2019-11-03T20:27:38.000000Z
Name: job_discoverdate, dtype: object
0    2019-10-23T18:48:28.000000Z
Name: job_discoverdate, dtype: object
0    2019-11-06T17:35:33.000000Z
```

Dimensional model



Como se puede apreciar de esta manera quedó el modelado dimensional, con sus 5 dimensiones y la tabla principal la tabla de hechos, a pesar de que no se vean las conexiones con las otras tablas, benefits se conecta mediante el id, y para salary stats y salaries se efectúa mediante el título del trabajo, esto principalmente porque la de salary stats aunque tuviera un id que supuestamente hacía la conexión a la tabla principal en el segundo corte se pudo ver que no se podía hacer la conexión con el id, esta tabla contaba con cosas raras como el número exagerado de filas y la

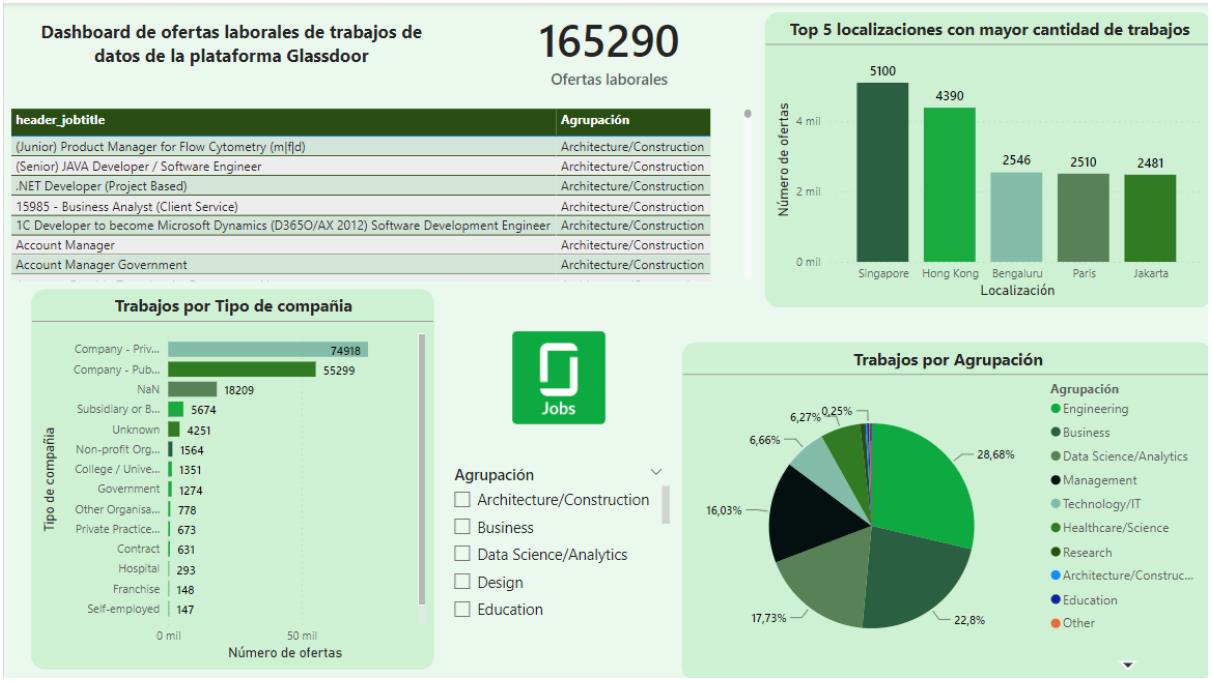
duplicación de id, por ende se tomó la decisión de hacer la conexión gracias al título del trabajo.

Ahora bien la razón de tener tantas tablas como dimensiones es porque disponía de una cantidad bastante grande de columnas para ello tuve que pensar en su división de manera que tampoco quedarán muy saturadas y las dimensiones tuvieran sentido entre si, asi que los beneficios,informacion del trabajo, información de la compañía, información del salario en cuanto a estadísticas y la información de los salarios gracias a las consultas a la API fueron necesarias para su desarrollo.

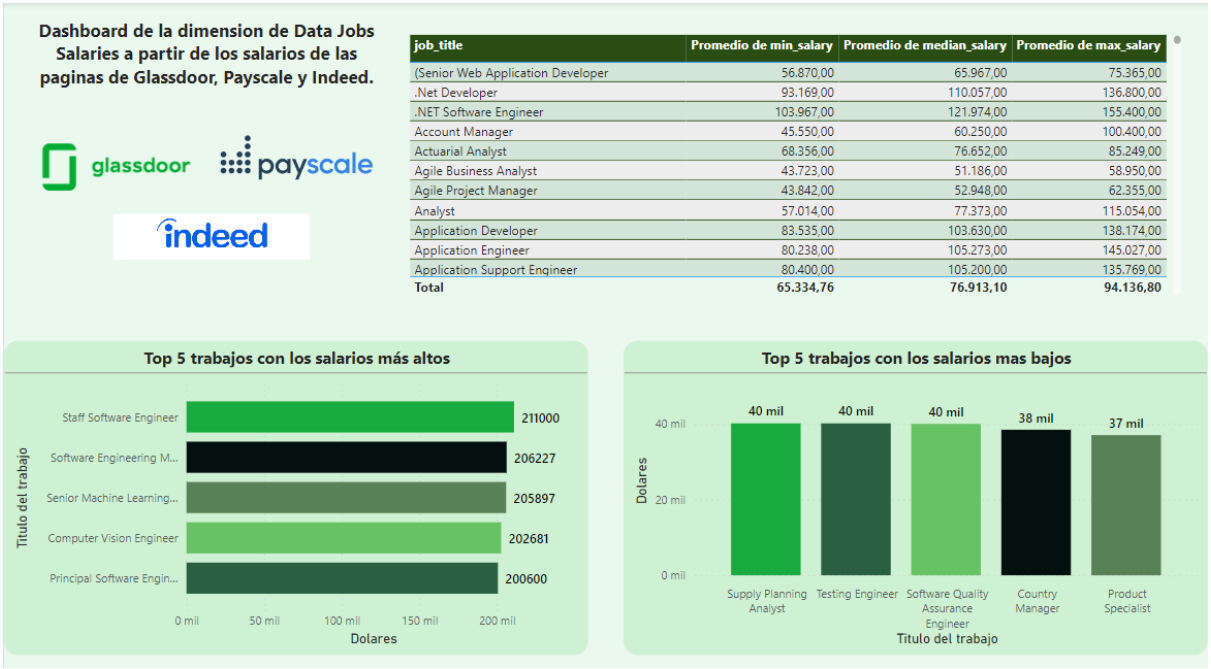
Dashboard

Anteriormente explique cómo se efectuó la conexión con power BI para eso tuve que hacer uso de docker en donde los contenedores que use fue postgresql y la interfaz de pgAdmin para la facilidad de uso de la base de datos. Entonces por medio de la dirección de eth0 de la máquina de wsl2 la IP 172.25.138.33 se realizaron los dashboard que disponía anteriormente, sobre las agrupaciones del trabajo y el de la API.

Dashboard principal



Dashboard de la API



En caso de que se quiera saber información del uso de cada una de las gráfica e información específica de las interpretación se encuentra en las anteriores entregas del proyecto.

EDA'S

En cuanto a la información de los EDA 's se puede mirar en los notebook de Jupyter ahí está específicamente el proceso que se manejo en el análisis exploratorio de datos, y las conclusiones que se sacaron a partir de estos.

Kafka consumer

Para el kafka consumer el propósito de este fue hacer envío de los datos a la API de power BI y enviarlos por medio del proceso de streaming, inicialmente a este se le asigna el topic al cual consumirá, para luego ya hacer un consumer.poll y un consumer.seek_to_end mediante estas dos funciones se hará la petición de los nuevos mensajes y busca el topic de kafka para comenzar a consumir los mensajes desde los más recientes.

Luego en cada uno de los mensajes de transforma a un dataframe para efectuar la transformación del formato de la fecha y ya luego enviar estos en formato de bytes, esto es específicamente para la API de power BI en donde para cada uno de los datos y tener la capacidad de leerlos se maneja gracias a este formato y se hace el post a la API. Aquí hago muestra del código del consumer y entenderlo de una mejor manera.

```
def kafka_consumer():
    consumer = KafkaConsumer(
```

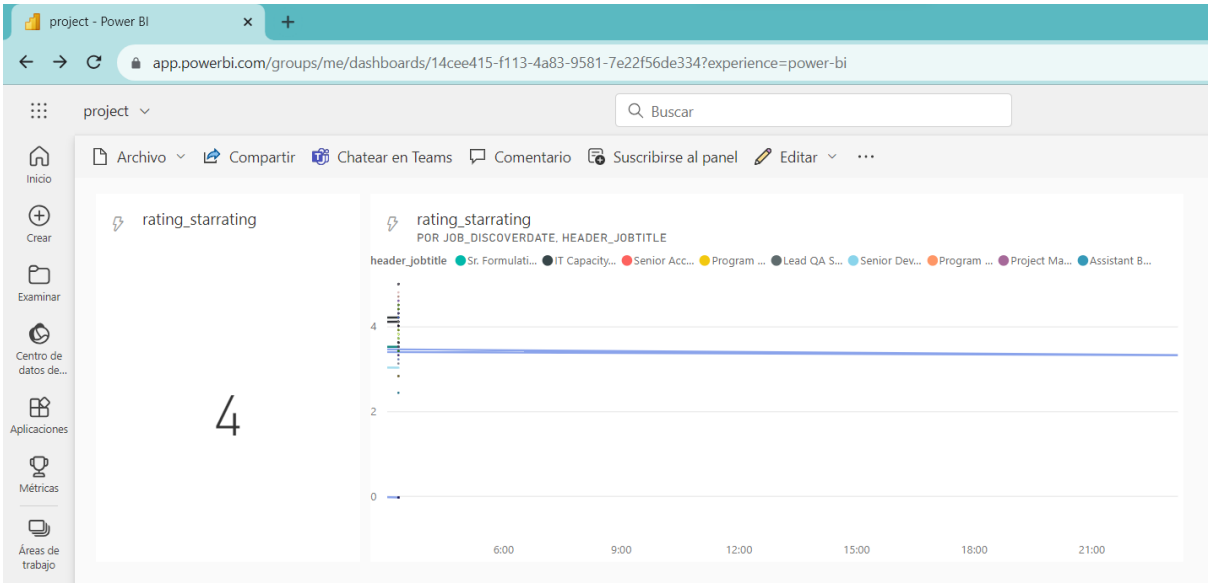
```
        'project',
        auto_offset_reset='earliest',
        enable_auto_commit=True,
        group_id='my-group',
        bootstrap_servers=['localhost:9092'],
        value_deserializer=lambda m: loads(m.decode('utf-8'))
    )
    consumer.poll()
    consumer.seek_to_end()

    for message in consumer:
        s = message.value
        df = pd.DataFrame.from_dict([s])
        df['job_discoverdate'] =
pd.to_datetime(df['job_discoverdate'])
        df['job_discoverdate'] =
df['job_discoverdate'].apply(lambda x:
x.strftime("%Y-%m-%dT%H:%M:%S.%fZ"))
        print(df["job_discoverdate"])

        data = bytes(df.to_json(orient='records'), 'utf-8')
        req = requests.post(API_ENDPOINT, data)
```

Real time dashboard

Ahora bien para el real time dashboard especifique hacer uso de únicamente 3 columnas, para poder mirar la evaluación de cada uno de los trabajos y como se encontraban clasificados para ello a la api le pase el formato de las columnas de esta manera.

```
[
 {
  "header_jobtitle" :"AAAAA555555",
  "job_discoverdate" :"2023-11-18T04:56:41.763Z",
  "rating_starrating" :98.6
 }
]
```

Para posteriormente hacer la creación de una tarjeta en donde se especifica el rating del trabajo, y una gráfico de líneas en donde al pasar el tiempo se usa como eje x la fecha y el star rating y el trabajo como los valores que se expresan en ella.



De esta manera es posible ver cómo es calificado cada uno de los trabajos y poder sacar deducciones de cuál sería la mejor opción de elección si es que todavía no se ha definido por alguno en especial.