Zophia Bootcamp - Final Project

Gracia F. Ramiro

March 2022

Resumen

El proyecto consiste en armar un flujo de extracción, transformación y carga de datos utilizando los programas Airbyte para el exportación de datos hacia Google Cloud Storage (GCS), capas de staging y producción en BigQuery (BQ). Se utilizó un cluster de Dataproc trabajando con Notebooks de Jupyter web con lenguaje de Spark y Python.

Introducción y descripción del proyecto

Como es sabido, un flujo ETL es un sistema que tiene la capacidad de leer diferentes formatos de archivo y tipos de datos para transportarlos de un entorno a otro.
En nuestro caso particular tomamos datos de Cockroach, sistema comercial distribuido de administración de bases de datos SQL, por medio de Airbyte, nos permite crear réplicas incrementales que llevamos a GCS para el almacenamiento crudo de datos. Luego de un cluster creado por medio de Dataproc, un servicio que permite gestionar los cluster de Spark y Hadoop en Google Cloud, cargamos los datos con una ligera limpieza a una capa staging en BQ para facilitar la extracción de datos, se utiliza los recursos y la facilidad de acceso que nos brinda el cluster para crear las tablas listas para el consumo de análisis de negocios.
Los datos con los que trabajaremos corresponden a productos, precios, numero de ventas, etc. Se podrá ver los datos con mas detalle por medio de la documentación provista por Zophia data academy.
Los lenguajes utilizados para los procesos que involucran el cluster de Dataproc son python con librerías como Pandas y Spark, un motor ultrarrápido para el almacenamiento, procesamiento y análisis de grandes volúmenes de datos.
Además de los datos estáticos, desde BQ se encuentra unos datos incrementales diarios, por medio de JOBS que se ejecutaran manualmente, se desea trasladar dichos datos incrementales a la capa de staging correspondiente y procesarlos como es debido para agregarlos a las tablas que serán utilizadas para el análisis.

Plan de trabajo

Como se nombra al principio, lo primero es realizar una replica de datos desde Airbyte hacia GCS, esto no presenta más dificultad que configurar una fuente y un destino en Airbyte.

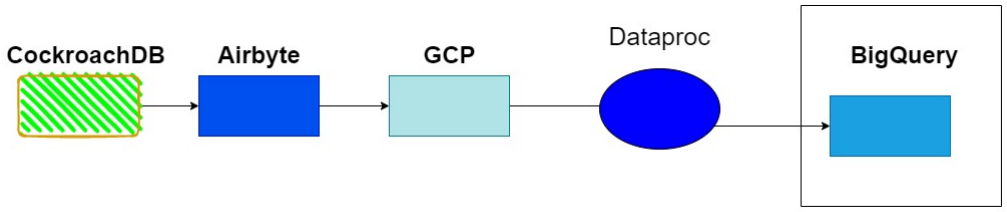

Figure 1: Replica de datos desde CockroachDB hacia GCS

Se creó un Notebook, que posee la lógica para el traslado de datos desde GCS hacía la capa staging de BQ llamado "create_stg_tables.ipynb".

Para el acceso a los datos se utilizó una lógica similar a la siguiente:

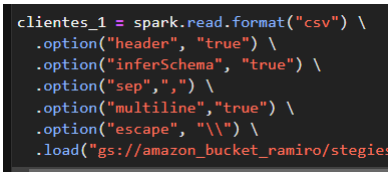

Figure 2: Adquisición de datos a dataframe de spark

Luego una ligera limpieza donde eliminamos columnas creadas por Ab, renombramos y "casteamos" algunas columnas según corresponda. La lógica se ve aproximadamente de la siguiente manera:

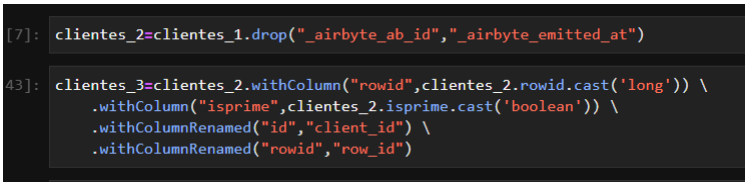

Figure 3: Limpieza de datos modificados por Ab

Luego se realiza un "write" sobre una capa staging de bigquery para almacenar los datos en esta con los propósitos nombrados en la sección anterior.

Figure 4: Write en la capa staging

Este proceso se repitió para todos los conjuntos de datos creando las siguientes capas de staging, STAGING CLIENTS, STAGING COMPRAS, STAGING EXTERNAL PRODUCTS, STAGING PRODUCTS y STAGING TASAS CAMBIO ANUAL.

Una vez que se posee las tablas de stg, se puede proceder a realizar las tablas pedidas, empezaremos por la tabla STAGING PRODUCTS, que pasa por un proceso de limpieza para posteriormente crear las tablas products_standard_price, products_avg_price, products_price_range, product_rate_avg .Estos pasos fueron resueltos con anterioridad en python, por lo que la lógica se adapto para notebook. Esto implica que los datos deben ser convertidos en dataFrame de pandas lo cual no es lo mas eficiente, pero si lo más practico teniendo en cuenta los tiempos de entrega del proyecto.
Para la tabla products_avg_price, se utilizó una linea de código de spark que facilita la obtención de la tabla solicitada.

Para la extracción de los datos en staging se utilizó:

```
table = "becade_rgarciaf.stg_products"
products = spark.read \
  .format("bigquery") \
  .option("table", table) \
  .load()

products.printSchema()
```

```
root
 |-- app_sale_price: string (nullable = true)
 |-- app_sale_price_currency: string (nullable = true)
 |-- country: string (nullable = true)
 |-- evaluate_rate: string (nullable = true)
 |-- isbestseller: boolean (nullable = true)
 |-- isprime: boolean (nullable = true)
 |-- original_price: string (nullable = true)
 |-- product_detail_url: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- product_main_image_url: string (nullable = true)
 |-- product_title: string (nullable = true)
```

Figure 5: Lógica de adquisición de datos desde BigQuery al entorno de Jupyter Lab

Luego se guarda los datos y se pasa por los siguientes procesos, normalización de los precios(app_price_normlizer), limpieza de los símbolos de moneda (clean_currency). Luego de manera similar a se extrae los datos de tasas de cambio, de donde se toman los valores actuales de las tasas de cambio, con esto generamos los precios de los productos en dolares por medio de una función (generate_app_price_us) que toma de variables los datos de los productos y de las tasas de cambio. Para finalizar convertimos los datos de nuevo en dataFrame de spark y escribirlos como una nueva tabla "products_standard_price" que era parte de lo pedido.
Para la tabla products_price_range se paso los datos de la tabla products_standard_price por una función (find_max_and_min) encargada de devolver un arreglo que posee los precios maximos, minimos y los países correspondientes a esos precios, esta función combinada junto con otros pasos utilizando funciones de spark obtiene los resultados deseados.
Para la tabla product_rate_avg de la tabla standar price se filtro por medio de una función (filter_evaluate_range) los caracteres no deseados dejando solo la evaluación, luego por medio de funciones de spark se obtuvo el resultado deseado. Con la capa staging de external_products se procedió de manera similar.

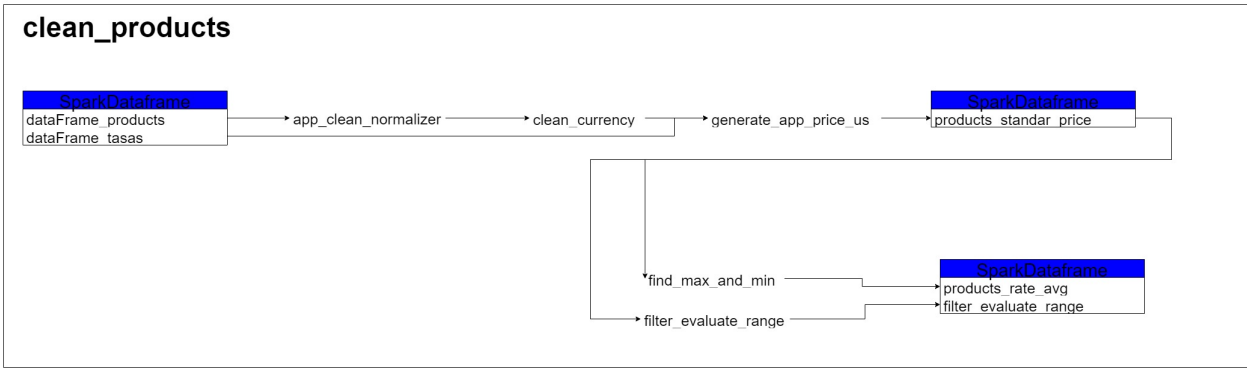

Figure 6: Proceso de limpieza de datos de productos

Otro staging que debemos trabajar, STAGING COMPRAS, para el cual se pidió que realicemos 3 tablas, compras_anuales, compras_mensuales y una tabla de hechos donde obtengamos determinada información detallada en el enunciado del proyecto.
De manera similar al paso anterior, obtenemos los datos de BQ y los cargamos en dataFrame de spark, luego por medio de funciones GroupBy y aggregate obtenemos las tablas deseadas, esto junto con determinados joins. Los pasos realizados se encuentran en el archivo createTables_from_compras.ipynob.
Para la tabla de hechos, utilizamos se "codeo" otro archivo llamado facts_tables.ipynb.

Adicionalmente para cubrir la agregación y manipulación de datos que se generan diariamente utilizamos dos archivos, create_stg_amazon_updates encargada de agregar datos de compra a STAGING COMPRAS y

create_amazon_daily_tables encargada de realizar las tablas de compras_mensuales y compras_anuales pedidas. La única diferencia es que una a la hora de escribir datos ambas realizan append. Para ello, estas traen los datos adicionales del staging, compras mensuales y compras anuales para no tener que realizar el proceso de transformación y limpieza de nuevo a los mismos datos, sino solo a los adicionales.

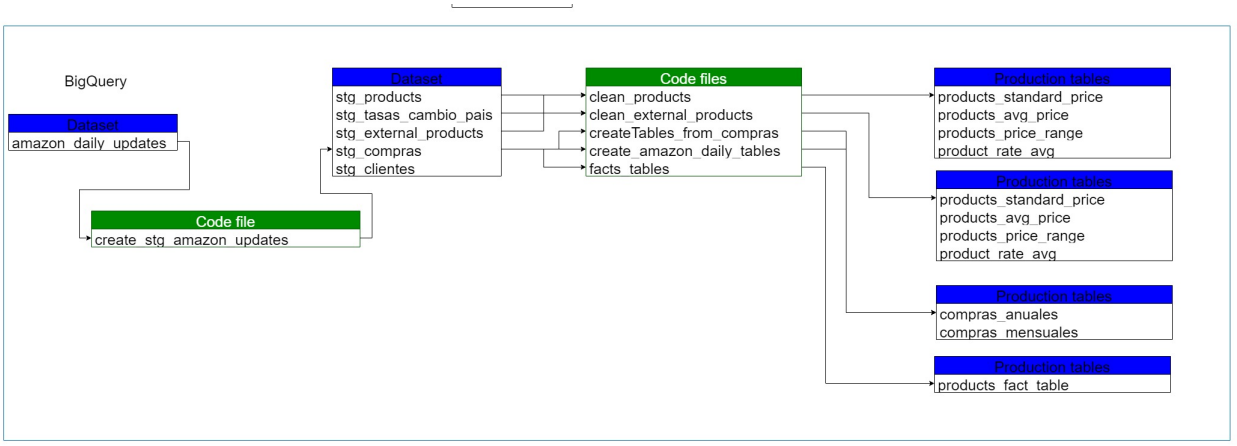

Figure 7: Proceso de creación de tablas de producción

Conclusión

Los datos poseen ahora una estandarización, las tablas fueron creadas según las especificación y en particular la tabla de hechos a partir de los datos de compras permite visualizar toda la información requerida.

Observaciones

El mal uso de la función count produce que a la hora de contar países se contara el mismo país varías veces, ya que muchos productos figuran mas de una vez en un mismo país. Esto se puede solucionar utilizando la función countDistinct en lugar de count o cambiando la lógica ligeramente para obviar los países repetidos. La lógica de python para la estandarización de los datos junto con su limpieza es compleja y dificil de leer, se recomienda trasladar toda esa lógica al lenguaje de Spark lo que además ahorrar un paso inecesario que involucra convertir los dataFrame de spark a dataFrame de pandas.

Resultados

54	false	B07SJ29VB4	46.4	£	false	4.5 out of 5 stars	GB	59.49
55	false	B07Y27R9C6	34.99	£	false	4.5 out of 5 stars	GB	44.86
56	false	B08158MN19	82.99	£	false	4.5 out of 5 stars	GB	106.4
57	false	B085S9XSWG	7.69	£	false	4.5 out of 5 stars	GB	9.86
58	false	B08G8R51CD	42.29	£	false	4.5 out of 5 stars	GB	54.22
59	false	1680227750	20.12	$	false	4.5 out of 5 stars	AU	13.85
60	false	B000O7CZKC	15.32	$	false	4.5 out of 5 stars	AU	10.54
61	false	B000O7CZKC	15.32	$	false	4.5 out of 5 stars	AU	10.54
62	false	B0028JPJ2A	53.52	$	false	4.5 out of 5 stars	AU	36.83

Figure 8: Productos estandarizados con el precio en dolares

1205	B00NH12R1O	8.78	DE	8.44	FR
1206	B00Q4TK2W2	27.86	DE	27.41	FR
1207	B00DUGZDJ4	17.13	DE	14.53	FR
1208	B07DRR5HHC	63.12	DE	56.88	FR
1209	B07WC1QWZH	47.54	ES	42.82	FR
1210	B07ZTS47LG	10.61	ES	8.7	FR
1211	B07KRBPPPF	14.81	ES	12.74	FR
1212	B07VXVDYD4	54.6	FR	54.6	FR

Figure 9: Rango de precios en distintos países

2851	B076WRP12Q	17.126667	3
2852	B076ZSR6BB	36.986667	3
2853	B0842DHX4Z	48.13	3
2854	B06X3W3YQD	23.13	3
2855	B07SYL958P	18.25	3
2856	B0798DTVL8	58.81	3
2857	B01LWP8AL2	28.356667	3
2858	B01B8R6PF2	20.216667	3

Figure 10: Precio promedio en distintos países

Fila	product_id	avg_evaluation_rate	country_count
3208	B06X6H2WF7	4.57	4
3209	B07NXGG55W	4.57	4
3210	B07PCXZ4CP	4.57	4
3211	B01M0A4B9M	4.57	4
3212	B01N17KOOO	4.23	4
3213	B07DTCFJTW	4.35	4
3214	B07Z9CRYT4	4.35	4
3215	B00XI87KV8	4.35	4
3216	B071G5KNXK	4.44	5
3217	B01GPXWNP0	4.44	5

Figure 11: Promedio de puntación en distintos países

Fila	year	venta_total	total_compras	avg_venta_mensual
1	2010	189442	177276	15786
2	2011	190838	178561	15903
3	2012	193062	180462	16088
4	2013	191325	178695	15943
5	2014	189880	177407	15823
6	2015	191521	178865	15960
7	2016	191064	178483	15922
8	2017	192075	179649	16006
9	2018	189889	177502	15824
10	2019	190901	178539	15908
11	2020	193183	180368	16098
12	2022	379	351	189

Figure 12: Compras anuales

25	3	2011	11932	11161	11409
26	3	2012	13056	12237	11932
27	3	2013	12607	11764	13056
28	3	2014	11987	11207	12607
29	3	2015	12011	11263	11987
30	3	2016	11498	10757	12011
31	3	2017	12128	11402	11498

Figure 13: Compras mensuales

Fila	year	product_id	cantidad_de_unidades	numero_de_clientes	cantidad_de_ventas	evaluate_rate
1	2010	B00N69D6AS	1927	100	1800	4.4 out of 5 stars
2	2018	B00N69D6AS	1955	100	1809	4.4 out of 5 stars
3	2014	B00N69D6AS	1926	100	1812	4.4 out of 5 stars
4	2020	B00N69D6AS	1947	100	1824	4.4 out of 5 stars
5	2019	B00N69D6AS	1939	100	1828	4.4 out of 5 stars
6	2013	B00N69D6AS	1852	100	1723	4.4 out of 5 stars
7	2017	B00N69D6AS	1858	100	1731	4.4 out of 5 stars
8	2011	B00N69D6AS	1848	100	1751	4.4 out of 5 stars
9	2012	B00N69D6AS	1881	100	1761	4.4 out of 5 stars
10	2016	B00N69D6AS	1893	100	1773	4.4 out of 5 stars
11	2015	B00N69D6AS	1913	100	1784	4.4 out of 5 stars

Figure 14: Tabla de hechos