School of Tech

Febrero 2023

Práctica 2.

David Orihuela Vélez, david.orihuela@accenture.com
Andrés Sánchez García, andres.d.sanchez@accenture.com
Rodrigo Rojas Gutiérrez, rodrigo.rojas@accenture.com

ÍNDICE

1. Objetivos de la solución.

Como aproximación a un proyecto real, se plantea la necesidad por parte de un cliente de tener una arquitectura big data. Para esta arquitectura, el cliente plantea una serie de requisitos que deben de ser cumplidos por parte del diseño:

A. Data lake para el almacenamiento de todos los datos que se van a explotar

B. Generación de un modelo final de datos en el data lake que sea totalmente accesible a partir de consultas SQL

Además, el cliente presenta una serie de requerimientos que el modelo final debe cumplir, con el objetivo de permitir realizar una cierta analítica sobre los datos. Estos requerimientos son tanto una serie de KPI's calculados dentro del modelo, como el enriquecimiento de las tablas finales con ciertos campos para facilitar tareas de explotación posteriores. Estas métricas son:

1) Número de usuarios por país que pertenecen a cada signo del horóscopo

2) Signo del horóscopo con más usuarios a nivel mundial

3) Grupo más escuchado por el signo del horóscopo con más usuarios

El data set inicial proporcionado cuenta con dos ficheros, de los cuales no se recibió mas información a priori, por lo que una de las tareas iniciales será explorar los datos dados para poder completar el modelo correctamente. Además, como otras necesidades del cliente, se encuentra la de generar valor a partir de completar y enriquecer las tablas iniciales del data set original, incluyendo en el modelo final:

4) Tabla maestra con los horóscopos

5) Tabla maestra de países y un campo con el continente al que pertenecen

6) Facilitar la agrupación de usuarios por su horóscopo, calculado a partir de la fecha de registro del usuario

7) Proporcionar mediante el modelo el continente al que pertenece cada usuario

El objetivo del cliente con esta arquitectura es poder realizar analítica sobre ella para tomar ciertas decisiones de negocio. Para ello les interesa conocer en que continentes es más útil hacer campañas de publicidad.

2. Arquitectura de la solución

El diseño de la arquitectura propuesta en la solución tiene como objetivo satisfacer las necesidades principales comentadas en el apartado 1), aportando una arquitectura con un almacenamiento en data lake, con una integración completa con una herramienta que permita lanzar consultas SQL sobre el lago.

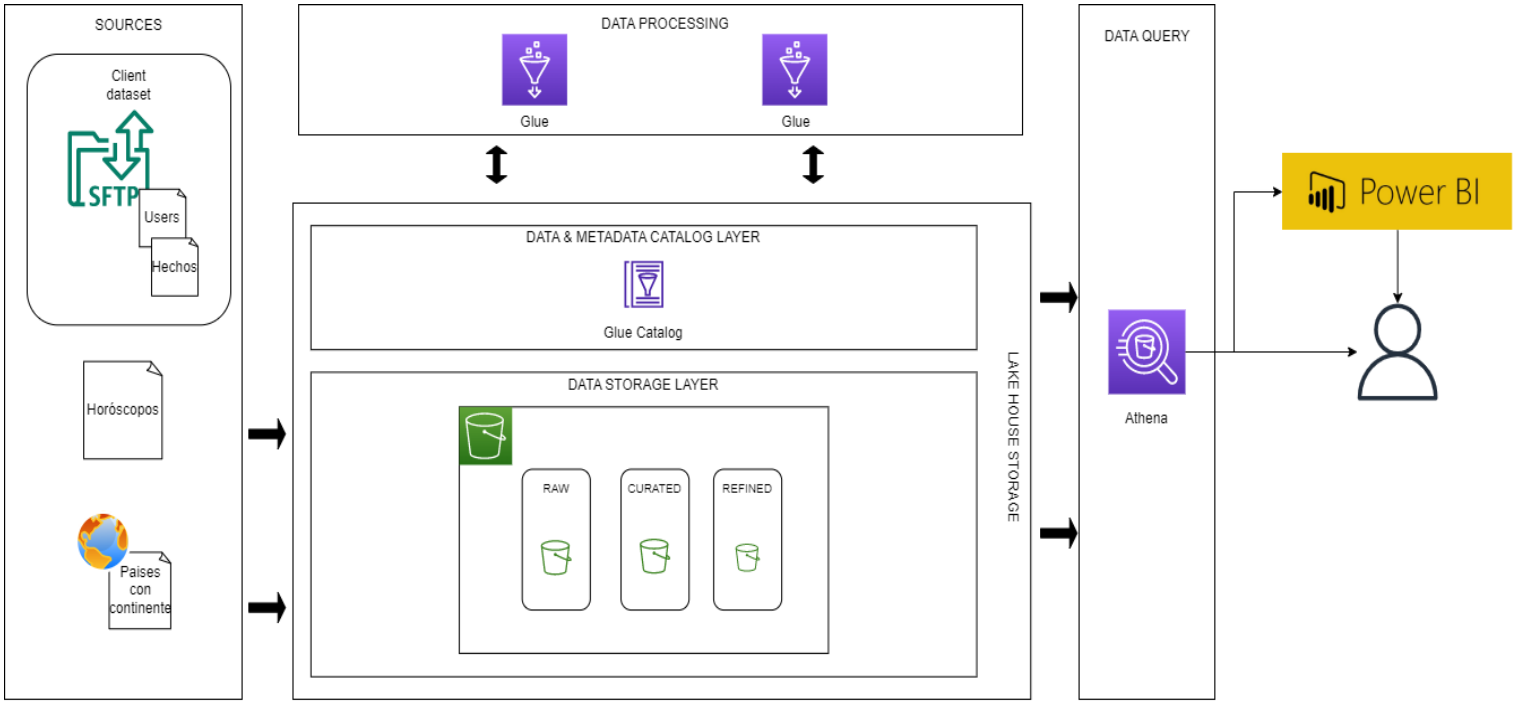
A lo largo de este apartado se desarrolla la arquitectura propuesta, presentando tantos los diagramas, diagrama de capas y el diagrama del ciclo de vida, como las herramientas y su integración dentro de la arquitectura.

2.1. Diseño de la solución

Para explicar correctamente la solución propuesta, se presentará primero el diagrama de capas que la componen, para después hablar del ciclo de vida de los datos dentro de ella, y las herramientas con la que se va a realizar la implementación.

2.1.1. Diagrama por capas de la arquitectura

Para la arquitectura, se ha seguido un diseño dividido en capas. Cada una de las capas representa una parte de la arquitectura que se encarga de realizar alguna de las tareas de las que necesita el ciclo de vida del dato, dentro de la arquitectura, para completarse y llegar hasta el cliente final.



En la imagen se muestran todas las capas que componen la arquitectura, siendo 5.

2.1.1.2. Data sources layer

Esta capa está compuesta por las fuentes originales de datos. Aquí tendremos tanto el data set original que ha aportado el cliente, como los data set incluidos desde otras fuentes para poder completar el modelo.

2.1.1.3. Data storage layer

Data storage layer, es la capa compuesta por el almacenamiento de los datos de la arquitectura, en todas las fases de este. Esta capa está dividida en 3 subetapas, donde se podrán encontrar datos a distintos niveles de madurez:

- **Raw stage**, datos en crudo, traídos directamente desde la capa de fuentes de origen.

- **Curated stage**, datos que han recibido un tratamiento para corregir formatos, nulos, duplicidades y otros aspectos.

- **Refined stage**, datos que se encuentran en un estado final y que son completamente explotables. Para llegar a este paso, es necesario enriquecer y generar valor explotable a partir de los datos que se encuentran en el curated stage.

Toda esta capa utiliza s3 para su implementación, y cada uno de los stages está representado por un bucket distinto. Esta capa, sumada a la siguiente, conforma todo el data lake completo, sobre el que se hará la posterior explotación de los datos.

2.1.1.4. Metadata catalog layer

Para poder integrar una herramienta SQL al almacenamiento de s3, es necesario otra capa que gestione todo el catalogo de metadatos y datos. Es decir, una capa por encima del almacenamiento que se encargue de traducir el almacenamiento bruto de s3 a una herramienta que permite el acceso a los datos mediante consultas SQL, como por ejemplo Athena.

Esta capa esta implementada dentro de la arquitectura usando Glue catalog, encargándose los mismos crawlers de Glue de llevar a cabo la sincronización del almacenamiento de s3 y el catalogo de metadatos de Glue.

2.1.1.5. Data processing layer

La arquitectura necesita de una capa que sea la encargada de procesar los datos para hacerlos madurar desde la etapa raw, hasta la etapa de explotación en el refined stage. Por lo tanto, esta capa necesitará acceder continuamente al almacenamiento de los buckets de s3 para ir recogiendo los datos, transformarlos y enviarlos al stage siguiente.

Ese proceso de extraer, transformar y cargar representa un proceso de ETL, para los cuales es necesario alguna herramienta que nos permita realizar computo. Aquí se ha escogido Glue como herramienta para procesar los datos e implementar los procesos de etl.

El objetivo de utilizar Glue, es ahorrar tiempo a la hora de aprovisionar la infraestructura para realizar el procesamiento de los datos. Al no necesitar de una carga computacional muy alta, Glue puede ser la herramienta perfecta para los procesos de etl que requiere la arquitectura.

2.1.1.6. Data query layer

Es necesario que el cliente pueda acceder mediante consultas SQL al modelo de datos almacenado en la capa de almacenamiento. Para ello, es necesario incorporar en la arquitectura una capa que utilice alguna de las herramientas de consulta SQL disponibles.

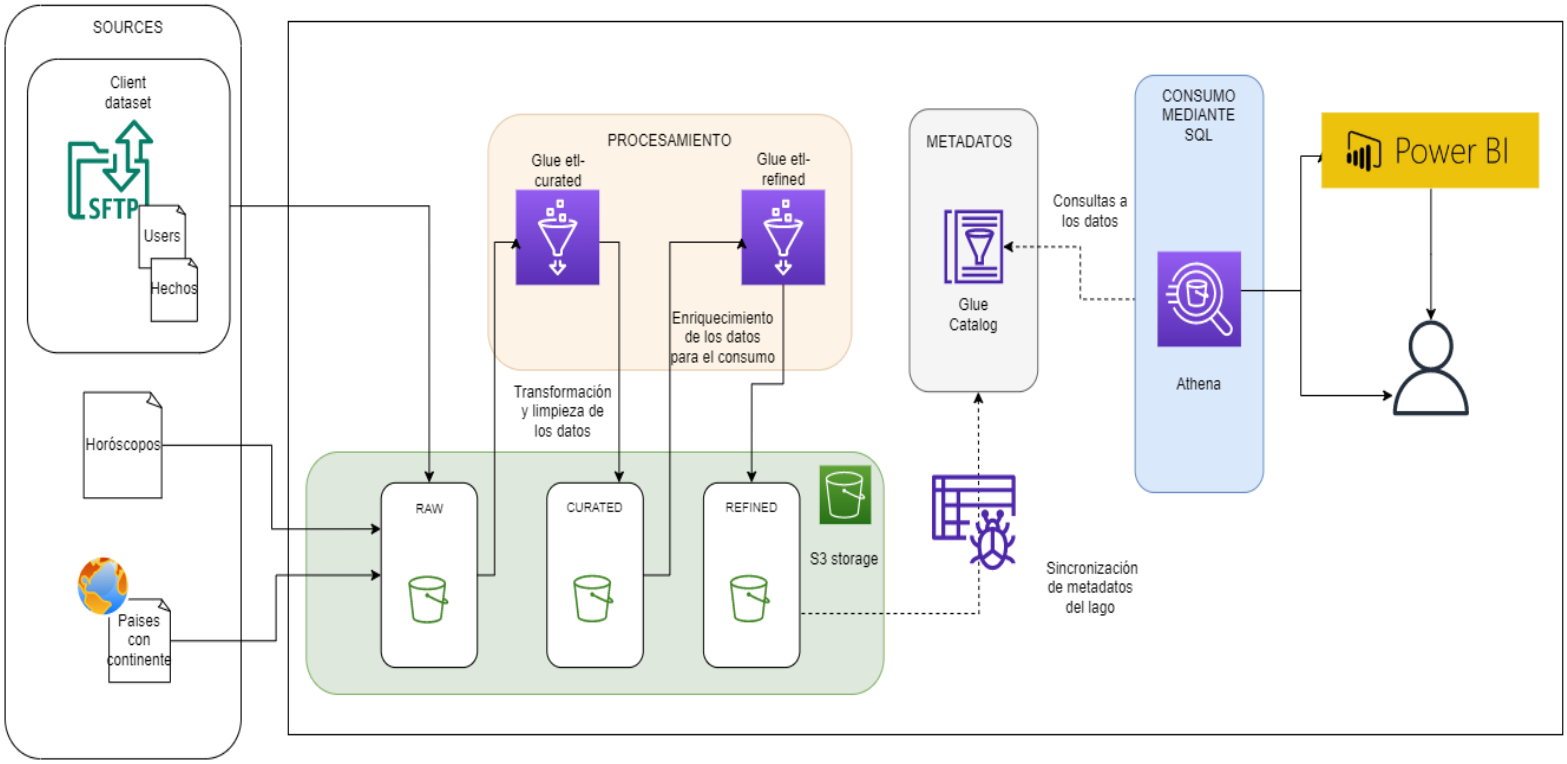
Aquí hay varias opciones, como incluir algún sistema OLAP o un sistema de base de datos tradicional, que permita realizar analítica sobre los datos del data lake, como por ejemplo un redshift o un rds. Sin embargo, debido a que el cliente no necesita una analítica muy extensa, la mejor opción es integrar una herramienta serverless como lo es Athena.

Athena, es un servicio totalmente serverless de consultas interactivas ofrecido por amazon. Athena se integra perfectamente con s3, permitiendo lanzar consultas sobre los datos que se encuentran en los buckets, gracias a la capa de metadatos de Glue Catalog, tal y como veremos en el siguiente punto, del flujo y ciclo de vida de la arquitectura.

Mediante todas las capas descritas, la arquitectura queda cubierta con todas las necesidades que se planteaban por parte del cliente en el inicio. El siguiente punto, consiste en entender como se integran las distintas herramientas y capas entre sí, y ver el ciclo de vida normal del dato dentro de la arquitectura.

2.1.2. Diagrama de flujo y ciclo de vida de la arquitectura

Una vez entendida la arquitectura, el siguiente paso es entender como es el flujo y como se conectan entre sí las distintas capas. El siguiente diagrama muestra todo el flujo y ciclo de vida de los datos, desde su estancia en las fuentes de origen, hasta su llegada al usuario final.



2.1.2.1. Ingestión y almacenamiento.

Para la parte de ingestión al haber sido los data set proporcionados directamente, se ha optado por tomar como punto de inicio una situación en la que los ficheros ya se encuentran dentro del raw stage del data lake.

Una vez aquí, los datos pasarán a formar parte del almacenamiento en nube de la arquitectura, y comenzarán a estar disponibles para que la etapa de procesamiento se conecte con s3 y ejecute los procesos de etl necesarios, para ir pasando etapa por etapa.

Dentro de las etapas del almacenamiento, se partirá de la etapa raw, un proceso de etl llevará los datos transformados hasta la capa curated y finalmente, otro proceso etl, llevará los datos finales a la capa refined, capa de explotación.

2.1.2.1. Procesamiento.

Tras haber llegado los datos a la capa de almacenamiento raw, el siguiente paso es realizar el primer procesado de ellos. Como ya se ha explicado en el punto 2.1.1.5., toda la parte de procesamiento se llevará a cabo mediante GlueJobs, utilizando pySpark.

Como muestra el diagrama el procesamiento se divide en dos etapas, dos procesos de etl:

1. Primera etapa de procesamiento encargada de comprobar formatos, añadir columnas, eliminar registros no válidos, y cual operación referente a valores de filas y columnas. Los datos tras esta etapa se encuentran en el curated stage

2. En la segunda etapa, se da prioridad a preparar el modelo de datos para su explotación por parte del cliente enriqueciéndolo, generando los cruces entre tablas necesarios para ello. Los datos en esta etapa se encuentran en el refined stage listos para la explotación.

El flujo es simple, se lanzará la primera etl, se recogen los datos del bucket de raw stage, se transforman y se dejaran en el bucket del curated stage, y una vez finalizada la primera etl es el momento de lanzar el segundo GlueJob. Este transformará de nuevo los datos, y dejará el resultado en el bucket del refined stage.

Una vez se han realizado las dos ejecuciones de los GlueJobs, ahora los datos están disponibles para su explotación, pero antes es necesario que la capa de metadatos este completamente sincronizada, para que Athena sea capaz de ver mediante Glue Catalog la ultima versión del modelo de datos del refined stage.

2.1.2.1. Sincronización de metadatos.

Para que Athena sea capaz de tener la ultima versión de los datos que existe en el bucket de refined stage, es necesario que los metadatos de Glue Catalog estén completamente actualizados.

Para ello, entran en juego varios elementos que existen dentro de Glue Catalog. En primer lugar, esta la base de datos. La base de datos esta enlazada con un almacenamiento y es donde se van a generar y gestionar todos los metadatos de aquellas fuentes de almacenamientos que seleccionemos para dicha base de datos. En este caso, seleccionamos el bucket de refined stage. Unido a la base de datos, es necesario crear un crawler. El crawler es el encargado de recopilar toda la información sobre los ficheros y los datos que están almacenamientos en el almacenamiento asociado a la base de datos, y actualizarlos dentro del Glue Catalog, por lo que su ejecución es fundamental para la sincronización de este.

El flujo que seguiría toda la arquitectura consiste en una vez almacenados los datos en la capa refined, se debería ejecutar el crawler asociado a dicho bucket, lo que provoca la

actualización de los datos en la base de datos del Glue Catalog. Mediante este flujo, Athena ya es capaz de consultar la última versión de estos y solo queda explotarlos.

2.1.2.1. Consumo SQL y de herramientas externas.

La ejecución del crawler resulta en la actualización del Glue Catalog y, por ende, Athena es capaz de ver lo último que existe dentro del data lake. Ahora, Athena es capaz de realizar consultas SQL bajo demanda del cliente, de forma que le permite realizar una explotación completa del modelo final de datos.

Como se definió en el apartado 1, además de facilitar una forma de explotar los datos con SQL, el cliente necesita tener unos KPI precalculados y poder visualizarlos dentro de un powerBI. Para satisfacer esta necesidad, dentro de Athena se han definido una serie de vistas, que permiten consultar las KPI requeridas de forma automática, además de permitir integrar un powerBI, para generar cuadros de mando utilizando las consultas de esas vistas, como se verá a lo largo de los apartados 3.4 y 3.5.

3. Desarrollo de la solución

Una vez explicada el diseño para la arquitectura que se ha propuesto como solución, el siguiente paso es explicar como se ha llevado a cabo el desarrollo de esta. Para ello se explica primero como ha sido el proceso y la metodología de desarrollo para después pasar a explicar paso por paso y de forma ordenada cuales han sido las implementaciones que se han ido realizando para cada uno de los elementos que componen la arquitectura.

3.1. método de desarrollo utilizado

El método de desarrollo que hemos utilizado se ha basado en un método simple, utilizando todas las herramientas que nos proporciona la consola de aws para llevar a cabo tanto los aprovisionamientos de infraestructura, como el desarrollo de las partes que necesitaban código, principalmente los procesos etl y la parte de análisis exploratorio de los datos, partes que se han desarrollado en GlueStudio mediante GlueJobs.

Además, se han ido utilizando los propios buckets s3 creados para el almacenamiento de los datos, como repositorio para tener almacenados los GlueJobs y algunos ficheros de logs, con el objetivo de tener todo sincronizado para todos los miembros del equipo.

Como otra forma de mantener a todo el equipo conectado, se ha utilizado también un repositorio de GitHub, pero debido a la falta de desarrollo local, por el uso de GlueStudio, este repositorio se ha utilizado más como una forma de almacenamiento alternativa de los ficheros del proyecto que como una herramienta de desarrollo ágil.

Es conveniente destacar que todos los recursos de la arquitectura han sido implementados sobre la región de London, eu-west-2, en la cuenta de aws.

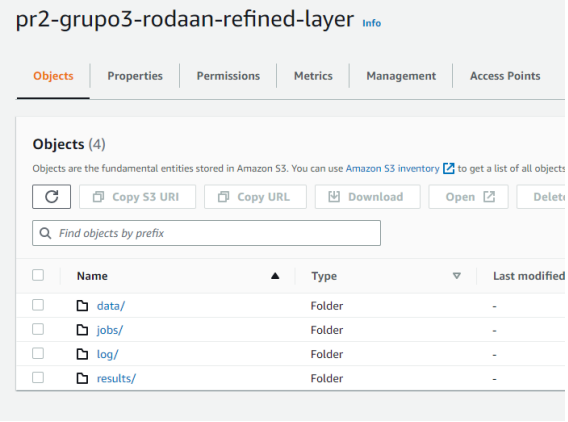
3.2. implementación del data lake

El primer paso de la implementación de la arquitectura es el aprovisionamiento de la infraestructura del data lake. Como se ha comentado en el apartado Diseño de la solución, el almacenamiento está basado en 3 capas de buckets, raw, curated y refined, cada capa representada por un bucket diferente. El primer paso consiste en crear los 3 buckets de cada uno de los stages, generados con los siguientes nombres:

- pr2-grupo3-rodaan-raw-layer
- pr2-grupo3-rodaan-curated-layer
- pr2-grupo3-rodaan-refined-layer

Dentro de los buckets raw y curated se pueden encontrar dos directorios, el **directorio data**, donde se encuentras los datos del modelo, y el **directorio log**, donde los procesos etl van dejando ficheros que contienen información sobre cada ejecución.

Dentro del bucket de refined además de los directorios de log y data, también podemos encontrar un directorio de Jobs, que contiene todos los GlueJobs desarrollados para la arquitectura y otro directorio results, enlazado con Athena para que se guarden resultados de consultas en él, en caso de necesitarlo.



El almacenamiento está implementado con los buckets, pero también se necesita levantar la parte de GlueCatalog, que como se explicó en el apartado Sincronización de metadatos., es totalmente necesario para poder conectar Athena con el almacenamiento de s3. También es necesario levantar la base de datos enlazada al directorio data de la capa refined y el crawler para actualizar los metadatos de dicha base de datos. Esta infraestructura es levantada a partir de la consola de Glue en aws, levantando:

- La **base de datos** pr2-grupo3-rodaan-refined-db pr2-grupo3-rodaan-refined-layer

- El **crawler** pr2-grupo3-rodaan-refined-crawler, enlazado a la base de datos anterior y ha dicho directorio

Hechos estos pasos, toda la infraestructura necesaria para la capa de almacenamiento ya está preparada.

3.3. Análisis exploratorio de los dataset

Para poder desarrollar los procesos de etl, primero es necesario conocer como están los dataset y entender cuales son los procesos que van a necesitar implementar los procesos de la solución. Para ello, se han hecho dos análisis exploratorios de los datos, utilizando en ambos casos, igual que para las etl, GlueJobs, en formato de notebook. Estos análisis son:

- analysis-raw-dataset, que se encarga de analizar los dataset originales, tal y como llegan de las fuentes, para conocer las operaciones a realizar sobre ellos en los procesos etl.

- tranform-data-to-curated, que se encarga de analizar los dataset que se generan tras la ejecución del primer proceso de etl, los dataset en la capa curated, para explorar que acciones son necesarias de ejecutar en la segunda fase de la etl, para cumplir por los requisitos impuestos por el cliente, definidos en el apartado 1.

Dentro de dichos GlueJobs se puede ver toda la información encontrada en los análisis y los problemas a solucionar en los procesos de etl.

3.3.1. Decisiones sobre el dataset original

Explorando los dataset dados, dentro del dataset de usuarios se pueden encontrar varios caso que hay que tratar como especiales y tomar decisiones sobre ellos:

1. Existen usuarios en la dimensión de usuarios que no tienen fecha de registro y que además tienen todos los valores a NULL menos el user id.

```
+-----------+------+----+-------+----------+------------+
|        #id|gender| age|country|registered|country_name|
+-----------+------+----+-------+----------+------------+
|user_000030|  null|null|   null|      null|        null|
|user_000129|  null|null|   null|      null|        null|
|user_000174|  null|null|   null|      null|        null|
|user_000353|  null|null|   null|      null|        null|
|user_000588|  null|null|   null|      null|        null|
|user_000734|  null|null|   null|      null|        null|
|user_000806|  null|null|   null|      null|        null|
|user_000976|  null|null|   null|      null|        null|
+-----------+------+----+-------+----------+------------+
```

Para estos usuarios al no aportar ningún valor útil se ha decidido excluirlos del dataset final y registrarlos, con el objetivo de poder informar al cliente sobre la existencia de ellos y sobre cómo tratarlos. Al excluirlos de la dimensión de usuarios, es conveniente eliminar sus hechos de la tabla de hechos

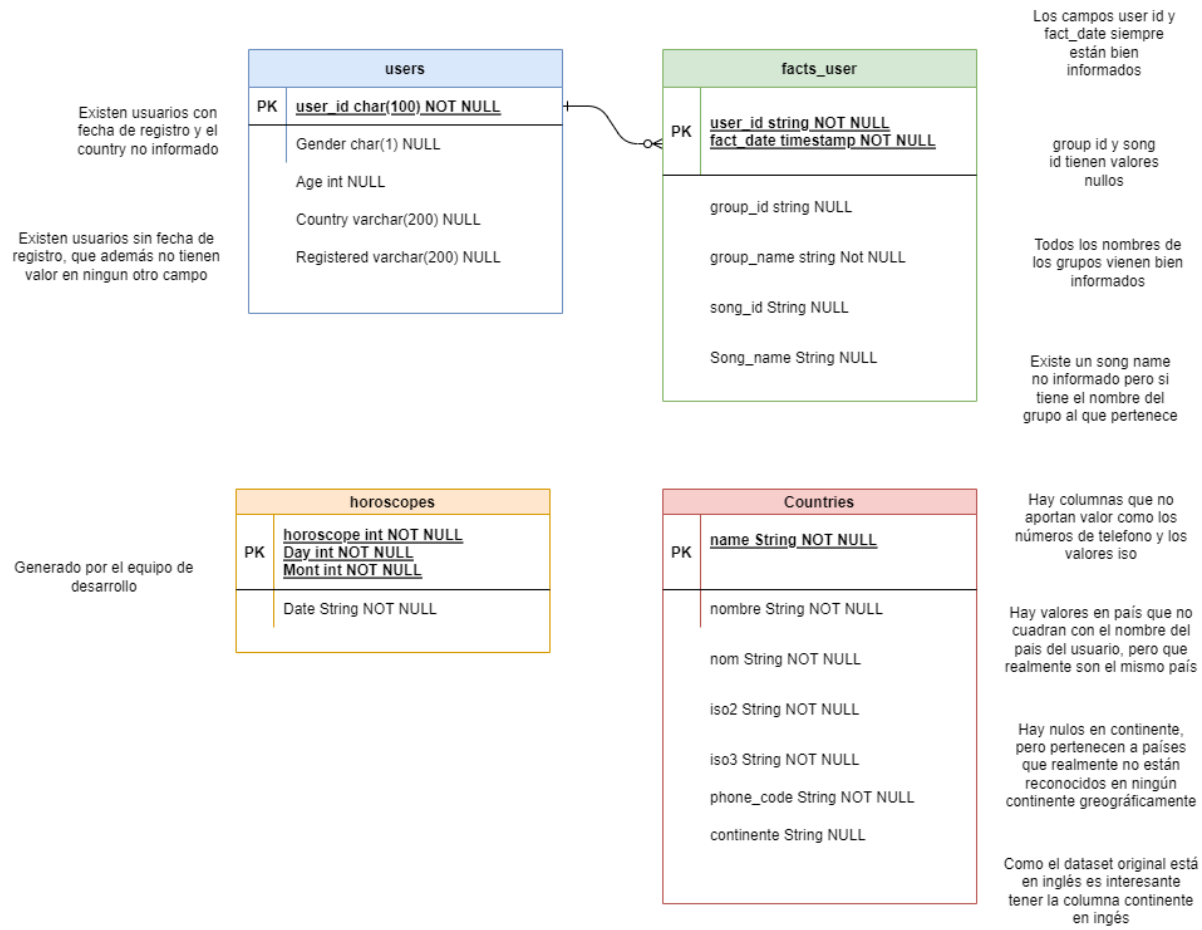
2. Hay otros usuarios que tienen fecha de registros y otros campos informados pero el campo de país viene vacío.

#id	gender	age	country	registered	country_name
user_000004	f	null	null	Apr 26, 2006	null
user_000014	null	null	null	Jan 27, 2006	null
user_000049	null	null	null	Jan 11, 2006	null
user_000061	null	null	null	Apr 29, 2007	null
user_000068	null	null	null	Oct 16, 2009	null
user_000077	null	null	null	Apr 24, 2006	null
user_000080	null	null	null	Aug 28, 2006	null
user_000086	f	27	null	Sep 21, 2007	null
user_000095	f	4	null	Jan 27, 2006	null
user_000126	f	null	null	Aug 3, 2006	null
user_000133	null	null	null	Jul 6, 2005	null
user_000146	null	null	null	Apr 4, 2007	null
user_000153	m	33	null	Jan 24, 2005	null
user_000171	null	null	null	Oct 29, 2002	null
user_000183	m	null	null	Dec 29, 2006	null
user_000200	null	null	null	Oct 20, 2006	null
user_000216	f	null	null	Jan 21, 2007	null
user_000220	f	null	null	Jul 27, 2006	null
user_000229	null	null	null	Jan 5, 2005	null
user_000231	null	null	null	Aug 9, 2006	null

Al tener informados campos importantes como el user id, la fecha de registros e incluso otros como el genero y la edad, se tomarán como registros útiles. Durante el cálculo de las KPI solicitadas, saldrá información sobre ellos que permitirán al cliente tomar decisiones acerca de ellos.

3.3.2. Modelo de datos inicial

Tras el análisis exploratorio del dataset proporcionado inicialmente se puede ver que se nos ha proporcionado unos datos sobre unos usuarios de una cierta aplicación, y los hechos que han realizado esos usuarios dentro de la aplicación, concretamente reproducciones de canciones de grupos de música. Modelo al que hemos incluido tanto una tabla maestra de horóscopos y otra de países con sus continentes. De esta manera el dataset original queda definido con este diagrama:



3.4. Implementación de las ETL de Glue

Para los procesos de ETL que mueven los datos entre capas y hacen que los datos del modelo alcancen la calidad necesaria para cada una de ellas, se ha optado por utilizar una solución serverless, que nos permita ejecutar y desarrollar el código sin necesidad de aprovisionar infraestructura. Es por ello por lo que Glue resulta como la mejor opción.

Al necesitar 2 movimientos entre etapas del data lake, movimiento de raw a curated y movimiento de curated a refined, el proceso se ha dividido en 2 ETL encargadas de cada uno de los pasos.

El desarrollo del código para ellas se ha hecho directamente sobre Glue utilizando notebooks y generando GlueJobs a partir de ellos. Estos dos GlueJobs son:

- etl-curated-data, para mover los datos a la capa curated

- etl-refined-data, para mover los datos a la capa refined

Las dos etl se pueden encontrar dentro del repositorio, en la carpeta de GlueJobs/etl.

3.4.1. Sistema de logs

Como punto para tener en cuenta, para poder realizar trazabilidad en las ejecuciones de cada Job, y conocer los puntos de error dentro de la ejecución, se ha implementado en ellas un sistema de trazas de log, que genera dentro de un directorio de log 1 fichero de texto por cada ejecución del Job, utilizando de nombre la Timestamp de la hora a la que se lanzó el Job.

Como trazas de log, se incluyen una serie de variables que contienen información sobre la operación que esta realizando el Job en cada paso de la ETL. Estas variables se incluyen al principio del script. Estas trazas tienen la siguiente forma:

1. Timestamp de escritura de la traza

2. Alerta de que la traza es de información o de error

3. Mensaje de la traza, en caso de ser un error se captura la excepción y se escribe en la línea.

```
(2023-03-28T15:41:11+0000) LOG INF: Reading countries data file to dataframe
(2023-03-28T15:41:34+0000) LOG INF: Changing column names
(2023-03-28T15:41:34+0000) LOG INF: Creating column continent in english
(2023-03-28T15:41:34+0000) LOG INF: Removing congo duplicates from dataframe
(2023-03-28T15:41:34+0000) LOG INF: Updating continent in countries that don't have continent
```

Estos ficheros de log se guardan en el bucket de destino que tiene el proceso que se esta ejecutando, la ETL de curated lo guarda en el directorio log del bucket curated (log/) y la ETL de refined lo guarda en el directorio log del bucket refined (log/https://s3.console.aws.amazon.com/s3/buckets/pr2-grupo3-rodaan-curated-layer?region=eu-west-2&prefix=log/&showversions=false).

3.4.2. Funciones útiles

Para el desarrollo de los procesos de etl, se van a necesitar varias funcionalidades que permitan al proceso interactuar con los buckets de s3, leyendo y escribiendo ficheros en ellos, librerías de variables globales con rutas y trazas de log y otro tipo de útiles.

La idea era elaborar unas librerías Python, almacenarlas en un paquete de útiles e importarlas desde los GlueJobs. Aunque se han encontrado formas de hacerlo, al necesitar un tiempo de implementación extra para realizarlo, se ha decidido por optar a incluir dentro del script de cada etl las propias funciones y variables que van a ser necesarias, ahorrando tiempo en el desarrollo. Estas funciones son:

1. **get_timestamp**(), sirve para coger la Timestamp actual, utilizada tanto para el nombre del fichero de log como para las trazas de log.

2. **vaciar_carpeta_s3** (bucket, carpeta), borra todo el contenido de un directorio de un bucket de s3. Esta función se utiliza a la hora de escribir en un directorio, y permite borrar primero el contenido de este, en caso de que no sea la primera ejecución de la etl, evitando que haya ficheros de datos repetidos en el almacenamiento.

3. **write_log_in_s3**(path, texto, Timestamp), permite escribir el texto pasado por parámetro dentro de un fichero con ruta path. Utilizado para escribir las trazas en los ficheros de log.

4. **write_df_to_s3**(log_timestamp, directory_path, s3_path, glueContext, spark, catalog_database, catalog_table, df_to_write, partition_keys=[]), permite escribir un dataframe de spark dentro de un directorio de un bucket pasado por parámetro un dataframe de spark. Siempre utiliza el formato parquet para la escritura y además se puede especificar la partition key en caso de necesitarla.

5. **read_s3_csv_to_df**(log_timestamp, spark, s3_path, format = 'csv', header = 'true', delimiter = '\t'), permite leer de un directorio cualquier tipo de fichero, por defecto intenta leer el formato csv, y con el delimitador '\t', pero se puede especificar el que se necesite. En caso de especificar otro formato que no es csv, como parquet, entonces la opción del delimitador no es tomada en cuenta.

3.4.3. ETL curated

Puedes encontrar esta etl en la ruta etl-curated-data o en el mismo repositorio en la ruta GlueJobs/etl. El flujo es el siguiente:

1. La etl comienza con la creación de la sesión de spark y la definición de las variables que contienen las trazas de logs.

2. después se definen las funciones útiles que se van a utilizar

3. Se definen funciones de limpieza para cada fichero del dataset. Dentro de estas funciones se realizan todas las operaciones necesarias para limpiar los datos, definidas a partir del análisis exploratorio. Por cada operación de transformación, se va escribiendo una traza dentro del fichero de log.

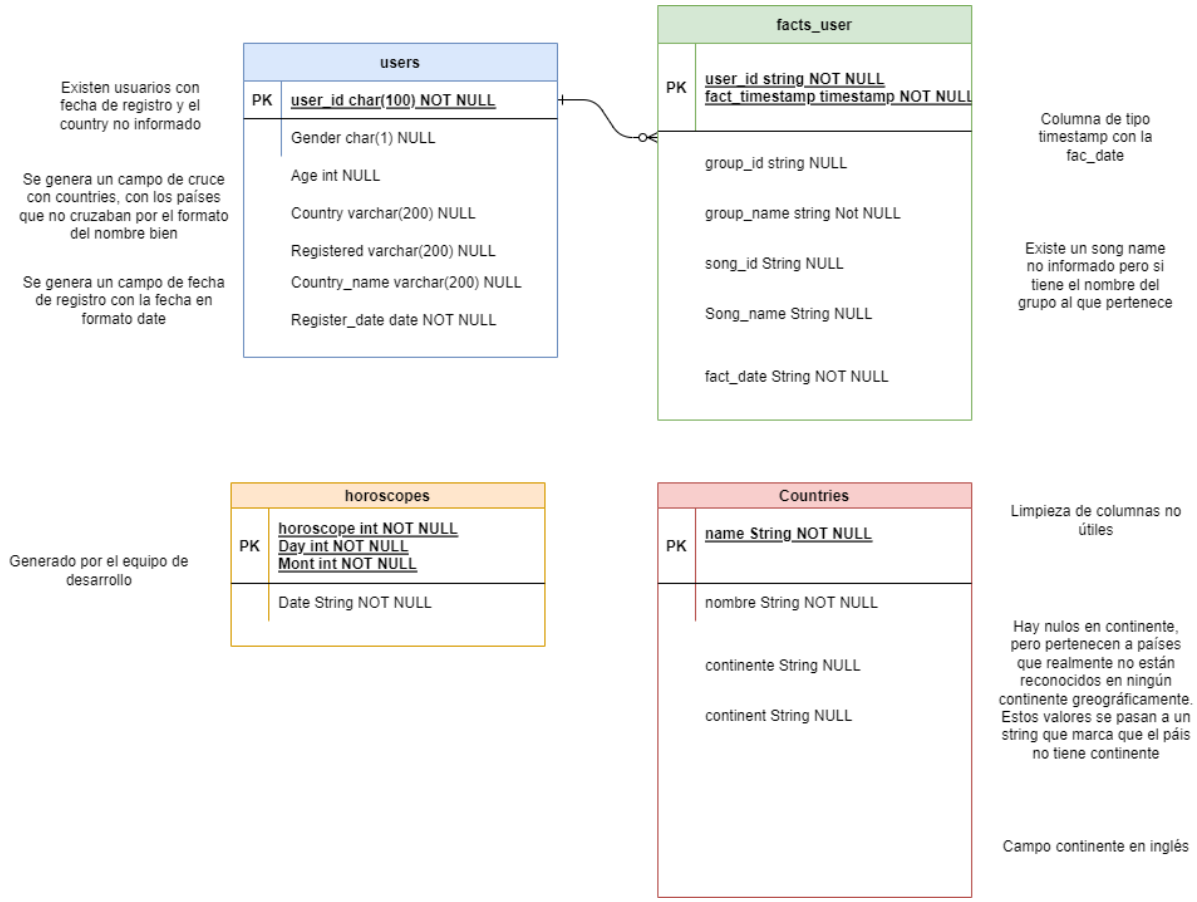
4. En ultima instancia se definen los pasos a ejecutar por parte del script para cada uno de los ficheros del dataset, dentro del cuadro "MAIN".

Como añadido, en la limpieza de los dataset de usuarios y hechos es necesario incluir 2 funciones más:

1. **udf_date_convert**(date), que es una función que recibe un string fecha en el formato que tiene dentro del fichero original de usuarios y la transforma a un campo de tipo date, mas utilizable. Esta función se utiliza para generar una UDF de spark y utilizar sobre el valor fecha de registro de todas las filas de usuarios.

2. **generate_list_users_to_clean**(df_users_without_values), se utiliza para generar un objeto de tipo lista a partir de un dataframes. Tiene el objetivo de generar una lista de usuarios, para poder realizar una limpieza de los hechos de ellos, dentro del fichero de hechos de usuarios. Esto se realiza debido a como se encontró en el análisis exploratorio, hay usuarios que no tienen ningún valor útil en sus columnas.

3.4.3.1. Modelo de datos resultante



3.4.4. ETL refined

Esta etl se puede encontrar en [etl-refined-data](). Tiene un flujo muy parecido a la anterior

1. La etl comienza con la creación de la sesión de spark y la definición de las variables que contienen las trazas de logs.

2. después se definen las funciones útiles que se van a utilizar. Explicadas en Funciones útiles.

3. Se define la función extract_data(log_timestamp, spark), para extraer todos los datos de los datos almacenados en el curated bucket y guardarlos en unos dataframes sobre los que se va a operar.

4. Se definen funciones de limpieza para cada fichero del dataset. Dentro de estas funciones se realizan todas las operaciones necesarias para enriquecer los datos con el objetivo de llegar a los requisitos establecidos por el usuario.

5. Se define la función load_refined_data(), que se utiliza para cargar dentro del bucket de refined los dataframes resultantes de la ejecución del proceso.

6. En última instancia se definen los pasos a ejecutar por parte del script para cada uno de los ficheros del dataset, dentro del cuadro “MAIN”.

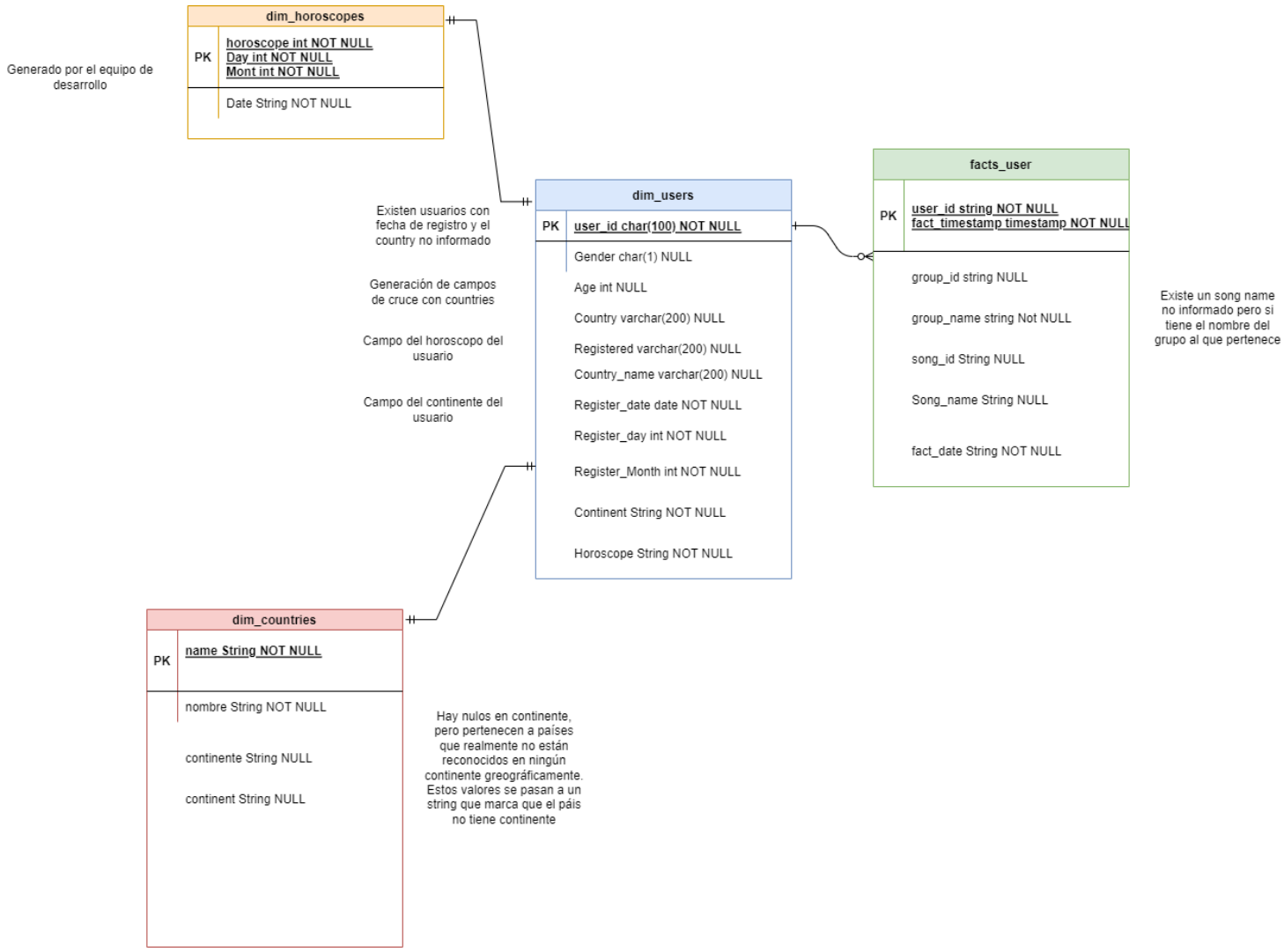
En esta etl el proceso de transformación está dedicado a la dimensión de usuarios. El objetivo es enriquecer esta tabla para que cumpla con las expectativas del cliente. Para ello, es necesario que tenga:

1. Los campos del continente al que pertenece el usuario, por lo que es necesario cruzar con la dimensión de países a partir del país del usuario y quedarse con el continente.

2. El campo del horóscopo al que pertenece el usuario en función de la fecha de registro. Para ello hay que cruzar con la dimensión de horóscopos a partir del campo de día y mes de registro, por lo que es necesario generarlos. Como se modificó la fecha de registro para que fuera un campo data, se puede utilizar la función month() para obtener el mes. Sin embargo, para el día, la función day(), no está soportada por la versión de spark que utiliza Glue, por lo que es necesario implementarla utilizando una UDF, lo que se hace con la función day(date), definida dentro de la ETL.

Una vez finalizada esta ETL, el modelo de datos a explotar esta completamente alcanzado, por lo que el siguiente paso, consistirá en actualizar metadatos, integrar Athena y calcular los KPI solicitados.

3.4.3. Modelo de datos final alcanzado



3.5. Implementación de la capa analítica en Athena

Una vez obtenido el modelo de datos final, ahora la capa refined está completamente preparada para proceder con su explotación por parte del cliente. Para eso, como ya se ha explicado, Athena proporciona una capa sobre la que lanzar cualquier consulta SQL y sobre la que realizar analítica.

Athena a partir de la integración que se ha explicado en el apartado 2.1.2., se integra perfectamente con el almacenamiento de s3, y permite consultar sus datos mediante GlueCatalog. Además, Athena puede conectarse a terceros, para poder elaborar cuadros de mando, o explotar las consultas SQL que se elaboran mediante otras aplicaciones.

El cliente solicitó conocer una serie de métricas y tenerlas incluidas en un powerBI, con el objetivo de poder tomar ciertas decisiones de negocio. Para ello, el cálculo de estas KPI se

ha realizado mediante consultas SQL, guardadas como vistas dentro de la base de datos de Athena, de forma que son totalmente visibles y consultables desde la propia consola de Athena, o desde herramientas como powerBI. Utilizar vistas nos permite proporcionar el usuario los datos de sus KPI totalmente actualizados con cada ejecución de los procesos ETL, de forma que los datos estarán totalmente sincronizados. Estas vistas y KPI son:

- **most_recurrent_horoscope**, para conocer el horóscopo mas recurrente del conjunto de usuarios

- **most_recurrent_horoscope_favourite_group**, para conocer el grupo de música mas escuchado por el horóscopo mas recurrente de los usuarios.

- **num_users_by_horoscope**, para conocer el número de usuarios por horóscopo que hay

- **total_facts_by_continent**, para conocer el número total de reproducciones de canciones que se han hecho en cada continente

- **total_horoscope_users_by_country**, para conocer el número total de usuarios de cada horoscopo hay en cada país

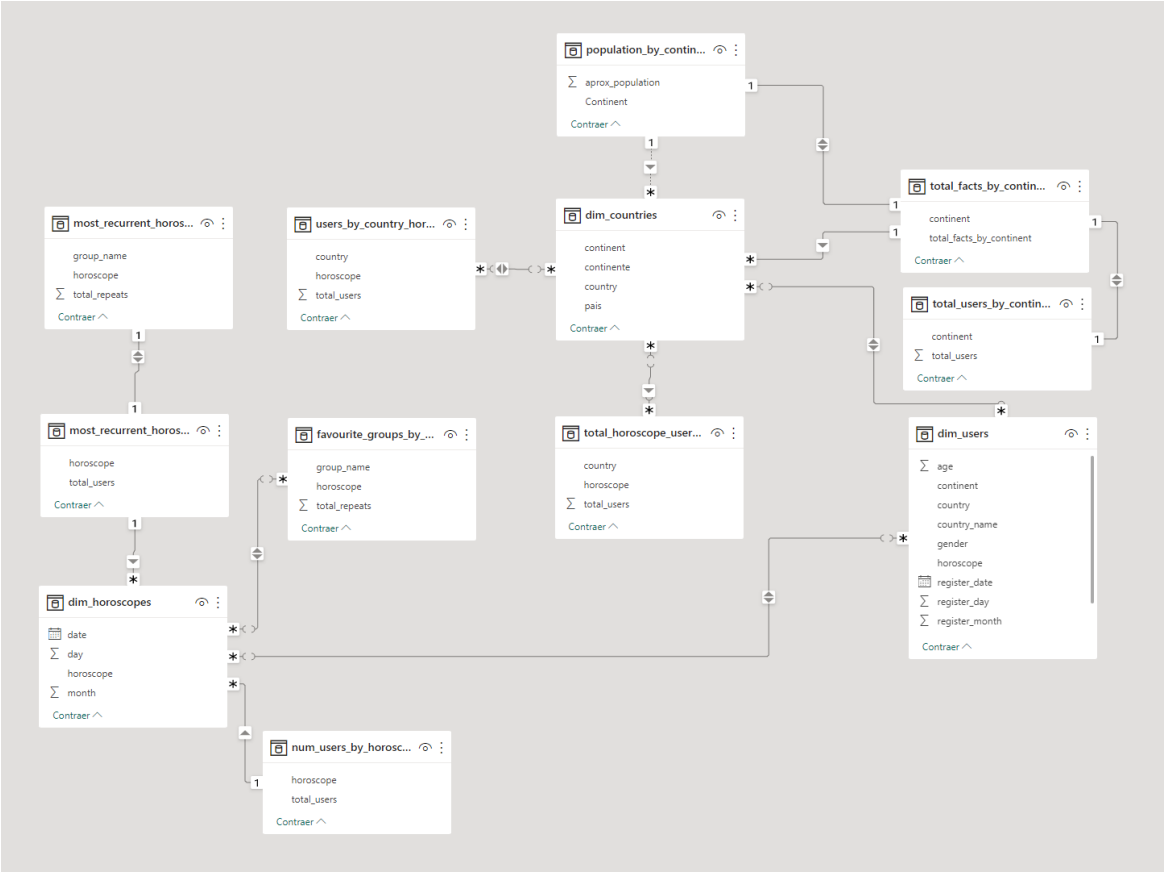
- **total_users_by_continent**, para conocer el número total de usuarios que hay en cada continente

- **favourite_groups_by_horoscope**, el conteo del número de reproducciones que tiene cada grupo por parte del horóscopo más recurrente en la dimensión de usuarios.

El código de las consultas puede ser consultado tanto desde Athena consultando la base de datos pr2-grupo3-rodaan-refined-db, del GlueCatalog AwsDataCatalog, o bien dentro del repositorio del proyecto en directorio **Athena/SQL/views/**

3.6. Desarrollo del cuadro de mando

El desarrollo del cuadro de mando se ha realizado en Power BI, se intentó conectar Athena con Power BI mediante ODBC pero no se pudo debido a los permisos con las credenciales, por tanto, se decidió cargar los datos de forma local, se realizaron las consultas en Athena y se almacenaron los resultados en archivos con formato CSV, una vez cargados los datos en Power BI, se procedió a realizar las relaciones de las distintas tablas y campos para posteriormente empezar el desarrollo visual del tablero para poder mostrar las distintas KPI solicitadas.

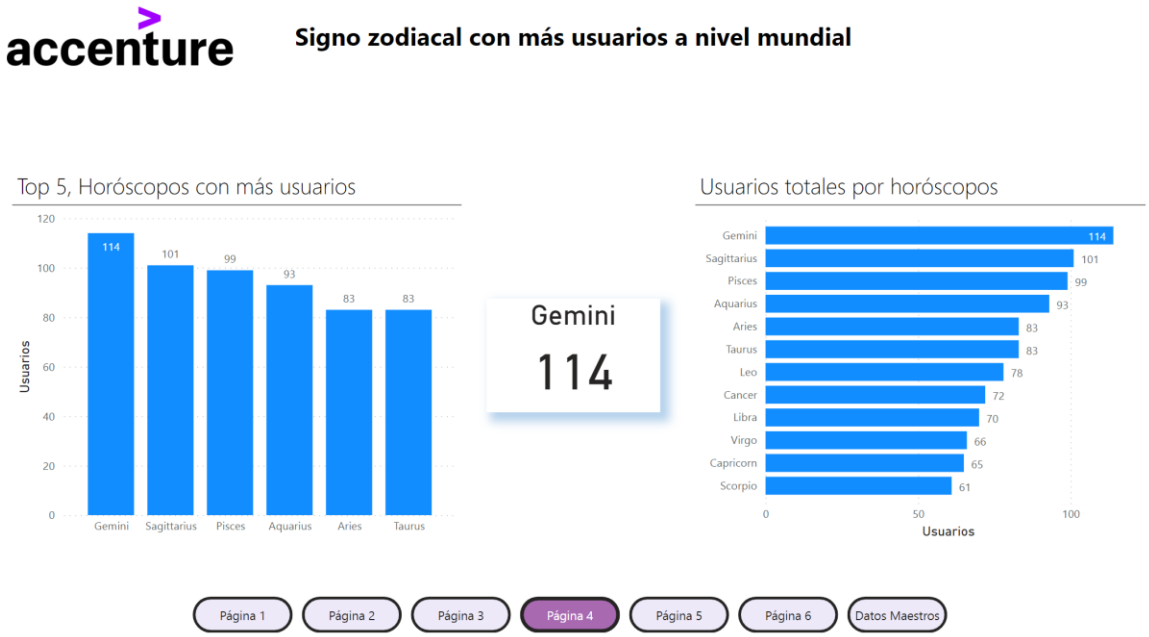


Para la primera KPI (“Necesitan saber a qué continente pertenece cada usuario”) se parte de una tabla maestra que es dim_users que contiene todos los datos de los usuarios, de la cual se puede extraer lo solicitado en la KPI.

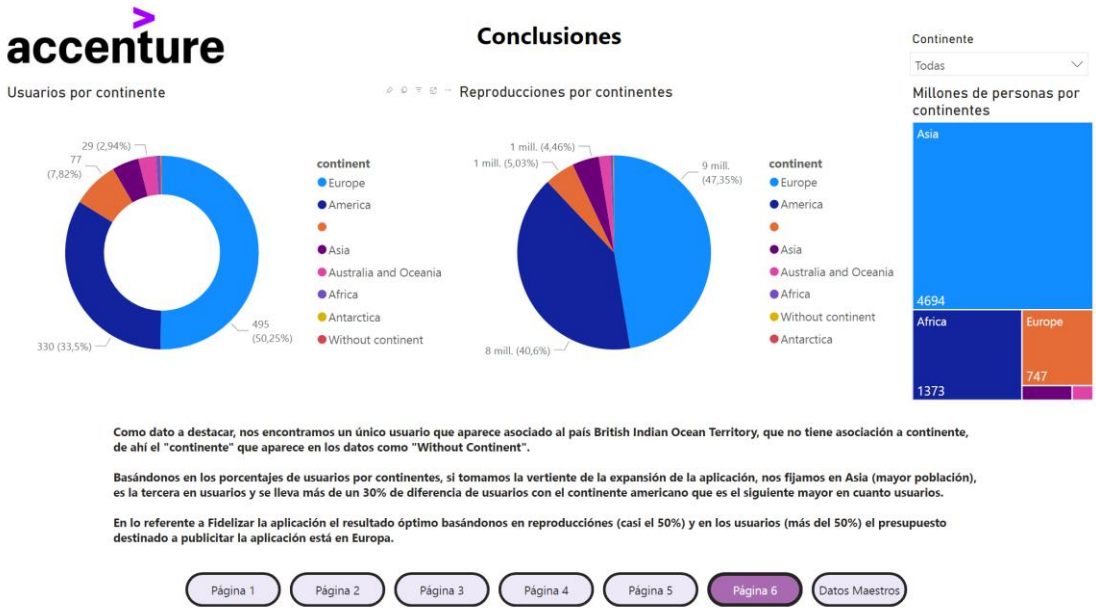
Para la primera página del panel de control, se ha añadido el resultado de la primera KPI en formato de tabla, donde se puede observar el ID de los usuarios, país y continente al que

pertenecen. Como información adicional se ha añadido un filtro y un buscador, además de un gráfico de columnas que refleja la cantidad de usuarios por cada continente.
Para la segunda KPI, donde se solicita saber la cantidad de usuarios por país que pertenecen a cada uno de los signos del horóscopo, la cual se muestra en la página 3, se añadió una tabla donde se observa la cantidad de usuarios por país y horóscopos, además de un gráfico de columnas para mejorar la visualización de la KPI, para ayuda de los usuarios, se añadió una serie de filtros y caja de búsqueda, estos gráficos muestran la información que se encuentra en la tabla “total_horoscopes_users_by_country”.

Para la siguiente KPI solicitada, donde se solicita saber el signo del horóscopo que tiene la mayor cantidad de usuarios a nivel mundial. Se añadió una nueva página que consta de 3 gráficos, por un lado, se muestra una tarjeta en la parte central con el dado solicitado en la KPI, por otro lado, se decidió realizar dos gráficos de barras con información adicional como la cantidad de usuarios totales que pertenecen a cada uno de los signos del horóscopo, además de los cinco horóscopos que más usuarios tienen, estos datos son extraídos de las tablas “most_recurrent_horoscope” y “num_users_by_horoscope”.



Para la última KPI, que relaciona a los grupos de música con los signos del horóscopo, se añadió una nueva página la cual muestra la información solicitada, además de un gráfico de barras con los principales grupos más reproducidos por el signo solicitado en la anterior KPI que es Gemini, para representar esta información se ha hecho uso de las tablas “favourite_groups_by_horoscope”y “most_recurrent_horoscope_favourite_group”.
Se ha añadido una página para representar las conclusiones del proyecto, la cual muestra información adicional que puede resultar de ayuda para los usuarios, como la cantidad de usuarios por continente y el número de reproducciones por continentes.



Finalmente se decidió añadir una última página con información de la tabla maestra, la cual cuenta con una serie de filtros que pueden ayudar a buscar información como edad de los usuarios, género, ID del usuario entre otros.

4. Memoria de problemas encontrados

A lo largo de la implantación de la infraestructura y el desarrollo de la solución se han encontrado varios problemas, a los que se ha tenido que dar solución para continuar con el normal desarrollo de la práctica:

1. Búsqueda de data set para Horóscopos y Continentes.

2. Problemas en las versiones de Python y Spark que usa Glue Jobs por defecto:

    a. Función day de pyspark no estaba presente, teniendo que desarrollar una función propia.

3. No hay almacenamiento temporal para librerías en Glue Jobs o carecemos de los permisos necesarios:
    a. Metemos las funciones de las librerías en el mismo Job.

4. Al agregar control de versiones mediante repositorios de git a Glue job, daba muchos problemas:
    a. Subir manualmente al repositorio.

5. Se buscaba automatizar el proceso mediante StepFunctions, pero no tenemos permisos:
    a. Se ejecuta manualmente.
6. No se podía conectar Athena con Power BI, por tema de permisos:
    a. Se intenta generar RDS (en vez de Athena) para conectar POWER BI, pero no se puede por tema de permisos:
        i. Optamos por descargar las fuentes de datos de Athena (tablas y vistas) para trabajar en local en Power BI.