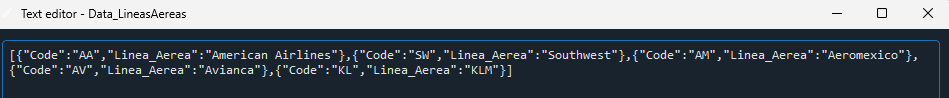
1. **Obtén los datos de la siguientes fuente de apis:**[**https://analytics.deacero.com/api/teenus/get-data/[api\_key]?format=json**](https://analytics.deacero.com/api/teenus/get-data/%5Bapi_key%5D?format=json)

Hago un request.get con la liga proporcionada agregando la key en la parte correspondiente.



Convierto la Response a Texto





Hago un json.loads del text

A screenshot of a computer program

Description automatically generated

Lo convierto en un dataframe utilizando pandas.DataFrame()

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Realizo el mismo procedimiento para el resto de las keys

1. **Se tiene un requerimiento de análisis de las fuentes de datos de pasajeros y viajes. Como podrás observar, las listas se han estado llevando por año. En esta primera parte del reto se te pide:**
   * **Unir cada conjunto de datos en una sola lista.**
   * **Explicar el proceso realizado.**
   * **En caso de detectar anomalías generadas por esta unión, deberás indicar el tipo de anomalía que se presenta, como se puede resolver y resolverlo de ser posible.**

En esta parte entiendo que lo que se debe realizar es unir los datos de “Pasajeros 2016” con los de “Pasajeros 2017” y los de “Vuelos 2016” con los de “Vuelos 2017”.

Como la meta final parece ser unir los datos de las tablas de Pasajeros con los de Vuelos, debo asegurarme de que no existan problemas con la variable que relaciona ambas tablas. En este caso, la variable es “ID\_Pasajero” en la tabla de pasajeros y “Cve\_Cliente” en la tabla de vuelos

Valores Duplicados en “Pasajeros 2017”

Utilizo .nunique() para identificar los valores únicos de “ID\_Pasajero” para asegurarme de no tener valores duplicados. Es importante no tener valores duplicados ya que planeo hacer un merge “Many:1”, en el que el vuelo solo puede tener 1 valor de cliente. Si se tienen valores duplicados en la tabla de Pasajeros entonces la tabla de Vuelos tomará varios valores de edad en lugar de solo uno.

Veo que la tabla de Pasajeros 2016 no tiene duplicados,



mientras que la tabla de Pasajeros 2017 tiene 33 valores únicos de ID\_Pasajero y 38 valores únicos de “Pasajero” el cual es el nombre.

A computer screen with white text

Description automatically generated

Decido crear una nueva variable llamada “ClaveUnica” en la que concateno las 3 variables de cada pasajero “ID\_Pasajero”,”Pasajero” y “Edad”. Vuelo a utilizar .nunique() y encuentro que hay 38 valores únicos de mi nueva variable “ClaveUnica”.



A screenshot of a computer

Description automatically generated



Utilizo .drop\_duplicates (keep='first',subset=["ClaveUnica"]) para eliminar las variables que claramente son repetidas.

A screenshot of a computer

Description automatically generated

Aún hay 5 valores que no cuadran debido al que solo se tienen 33 valores únicos de ID\_Pasajero. Investigo cada valor individual y encuentro que los ID\_Pasajero problema son 742, 717, 570, 582, 562. Todos estos números se repiten 2 veces en la tabla y están unidos a personas con diferente nombre y edad. Las edades son muy diferentes y para propósitos de este ejercicio no creo que exista una mejor acción a tomar. Decido realizar un drop de las variables duplicadas esta vez utilizando “ID\_Pasajero”. De esta manera no estoy seleccionando directamente cuales registros omitir de los datos. La alternativa pudiera ser tomar un promedio de las edades y escoger un nombre aleatorio, sin embargo, no creo que sea la decisión correcta debido a que en algunos casos habían clientes de 20 años y de 60 años y no sería exactamente representativo considerarlos como un cliente de 40 años ya que los 3 están en edades que tradicionalmente se categorizan como separadas (joven adulto, edad media, edad avanzada).

Al tener una tabla final de Pasajeros 2017 realizo una ultima revisión y veo que todos los valores de “ID\_Pasajero” son únicos. Realizo una unión de las tablas de Pasajeros 2016 y Pasajeros 2017 con pandas.concat. Acabo con una tabla de 133 filas, lo cual es lo que esperaba. Realizo un último .nunique() para la variable “ID\_Pasajero” y obtengo como resultado que los 133 IDs son unique.

A computer code with white text

Description automatically generated with medium confidence

“Vuelos 2016” y “Vuelos 2017”

No encontré problemas al trabajar con estas tablas. La variable “Cve\_Cliente” se puede repetir ya que no hay problema si un cliente toma mas de un vuelo. Vuelvo a crear la variable “ClaveUnica” concatenando todas las variables de cada fila como string. Esto para revisar que no se tengan registros duplicados del mismo cliente en el mismo vuelo. No encuentro duplicados. Realizo el mismo proceso para unir las tablas. Vuelvo a revisar que no se tengan registros duplicados al unir las tablas y encuentro 400 valores únicos para las 400 filas de la tabla.

A screen shot of a computer

Description automatically generated

1. **De lo obtenido anteriormente se requiere relacionar las listas de vuelos y pasajeros, que permitan analizar el perfil del pasajero por cada vuelo efectuado. De tal forma que se puedan obtener datos consolidados. En esta parte deberás:**
   * **Explicar el proceso que utilizado para unir los pasajeros y los vuelos.**
   * **Qué tipo de relación y por qué.**

Merge Many:1

Para unir los pasajeros y los vuelos se debe de utilizar la variable “Cve\_Cliente” y “ID\_Pasajero”. Planeo hacer un merge many:1 utilizando la tabla “Vuelos 2016-2017” como la tabla principal (many) y la tabla “Pasajeros 2016-2017” como la que se utiliza para tomar los datos del merge (1).

Motivo

El merge Many:1 es el adecuado cuando se utiliza la tabla de vuelos como la principal. Esto debido a que la variable que relaciona a ambas tablas se repite múltiples veces en la de vuelos y es única en la de pasajeros. Realizando este merge, Python simplemente agrega al final de cada vuelo los valores de Edad y Nombre del pasajero utilizando su ID. Tiene sentido realizarlo de esta manera ya que un cliente puede tomar múltiples vuelos en el tiempo pero un cliente no puede comprar el mismo exacto boleto 2 veces.

Procedimiento

El resultado final esperado es una tabla de 400 filas con las utlimas 3 columnas contendiendo las variables de “ID\_Pasajero”, “Pasajero” y “Edad”. Como solo hay datos de 133 pasajeros, se espera que varios pasajeros tomen más de un viaje, por lo cual no hay problema si se repiten los nombres y las edades. Para facilitar el merge, primero cambio el nombre de la variable “ID\_Pasajero” a “Cve\_Cliente” para tener el mismo nombre de la variable en ambas tablas. Realizo el merge utilizando el siguiente código:

A blue background with white text

Description automatically generated

En este código declaro la tabla de vuelos como la de la izquierda, la de pasajeros como la de la derecha y realizo el merge utilizando la variable “Cve\_Cliente” como la que relaciona ambas.

A screenshot of a computer

Description automatically generated



1. **Ahora se requiere que los datos consolidados de los vuelos y pasajeros se puedan unir con los datos de las Líneas Aéreas. En el caso de que la línea aérea no se pueda relacionar con la de vuelos y pasajeros se deberá indicar que se trata de “Otra” y finalmente se deberá dejar únicamente las columnas:**
   * **Fecha del viaje**
   * **Clase**
   * **Precio**
   * **Ruta**
   * **Edad**
   * **Línea Aérea**

**En esta parte deberás indicar:**

* + **¿Qué tipo de proceso consideraste para unir los datos que se piden?**
  + **¿Qué columnas utilizaste para lograr esa relación?**
  + **¿Qué tipo de unión utilizaste para unir los datos?**
  + **¿Qué tipo de proceso utilizaste para dejar únicamente las columnas que se piden?**

“Fecha del Viaje”, “Clase”, “Precio” y “Ruta”

Los datos de “Fecha del Viaje”, “Clase”, “Precio” y “Ruta” se tienen desde un inicio al utilizar la tabla de Vuelos como la principal.

”Edad”

Para agregar los datos de “Edad” se realizó el merge many:1 con la tabla de Vuelos utilizando la variable “Cve\_Cliente” para relacionar los datos. El proceso lo explico en la pregunta 3.

A screenshot of a computer

Description automatically generated

“Línea Aérea”

La variable de Línea Aerea requiere de un proceso similar al de la variable edad, con la condición adicional que, para los códigos no especificados en la tabla de Lineas Aéreas se guarden como “Otra”. La tabla de Lineas Aéreas tienen la columna “Code” la cual contiene los mismos datos que la columna originalmente llamada “Cve\_LA” en la tabla de Vuelos modificada con los datos de Pasajeros añadidos. Cambio el nombre de la columna “Cve\_LA” a “Code” para tener el mismo nombre de columna en ambas tablas. Realizo el mismo tipo de merge que utilicé anteriormente. Esto resulta en una nueva columna llamada “Linea\_Aerea” con los datos correctos para los códigos que se encuentran en la tabla de Lineas Aéreas. Los datos que no se encuentran en la tabla de Lineas Aéreas aparecen como nan (valores nulos) en la tabla en la que se realiza el merge.

A screenshot of a computer

Description automatically generated

Como se nos pide que a los códigos para los que no tenemos nombre los identifiquemos como “Otra”, utilizo .fillna(“Otra”)



A screenshot of a computer

Description automatically generated

Columnas Finales

Para quedarnos solo con las columnas de “Fecha del Viaje”, “Clase”, “Precio”, “Ruta”, “Edad” y “Línea Aérea” utilizo un drop en las columnas que no voy a necesitar. Como no necesitamos la columna Code, por ejemplo, utilizo el código

mergelinea = mergelinea.drop(["Code"],axis=1)

Donde “mergelinea” es mi tabla final







1. **Por último, se requiere el promedio semestral (el primer semestre es de Ene - Jun y el segundo es de Jul - Dic) del precio agrupado por Año, Clase, Ruta y las Línea Aérea como columnas.**

El código que utilicé para realizar los cálculos es el siguiente:

A screen shot of a computer program

Description automatically generated

En las siguientes tablas resumo los datos:

|  |  |  |
| --- | --- | --- |
| Año |  |  |
|  | 2016 | 2017 |
| Ene-Jun | 164.88 | 146.73 |
| Jul-Dic | 147.74 | 163.14 |

|  |  |  |  |
| --- | --- | --- | --- |
| Clase |  |  |  |
|  | Business | Economy | First Class |
| Ene-Jun | 157.64 | 147.14 | 166.67 |
| Jul-Dic | 160.14 | 150.82 | 155.16 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ruta |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | DAL-AMA | DAL-ATL | DAL-AUS | DAL-ELP | DAL-GRR | DAL-HOU | DAL-LGA | DAL-MDW | DAL-OKC | DAL-PHX | DAL-SEA | DAL-SLC | DAL-TUL |
| Ene-Jun | 131.5 | 155.54 | 160 | 148.75 | 185.05 | 99.38 | 154.15 | 144.53 | 166.18 | 176.05 | 102 | 179.1 | 162.56 |
| Jul-Dic | 140.27 | 139.36 | 166.41 | 148.22 | 139.8 | 143.54 | 157.75 | 134.79 | 154.76 | 170.92 | Null | 171.67 | 168.52 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Linea |  |  |  |  |  |  |
|  | Aeromexico | American Airlines | Avianca | KLM | Southwest | Otra |
| Ene-Jun | 168.68 | 153.09 | 157.68 | 160.87 | 149.75 | 122.6 |
| Jul-Dic | 169.69 | 157.68 | 154.56 | 167.87 | 137.8 | 99 |

Los Resultados aparecen en Python de la Siguiente manera:

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated