M. Sc. Liliana Millán Núñez liliana.millan@itam.mx

Febrero 2022

Proyecto 1

Utilizarás los datos de precios comerciales de gasolina en diferentes partes de México para realizar las 5 partes del proyecto.

¿Cuándo se entrega?

** de marzo del 2022 a más tardar a las 23:59:59 CST** por correo a liliana.millan@itam.mx con asunto del correo `examen suficiencia`.

¿Que se entrega?

5 scripts:

- `proyecto_1.sh`
- `proyecto_1_transform_1.sh`
- `proyecto_1_transform_2.sh`
- `upload.sql`
- `proyecto_1_load.sh`

Parte 1 - Extract

Genera un script `proyecto_1.sh` que:

1. Verifica si el archivo `estaciones.xml` existe en la ruta donde se encuentra este `script`.

- En caso de no existir baja el archivo de estaciones de servicio de la ruta https://bit.ly/2V1Z3sm con `curl` o `wget` y renombra el archivo (con línea de comando) a `estaciones.xml`.
- En caso de existir, no hace nada.

1. Verifica si el archivo `precios.xml` existe en la ruta donde se encuentra este *script*.

- En caso de no existir baja el archivo de precios comerciales de la ruta https://bit.ly/2JNcTha con `curl` o `wget` y renombra el archivo a `precios.xml` (con línea de comando).
- En caso de existir, no hace nada.

Parte 2 - Transform 1 - precios.xml

1. Genera un script de bash `proyecto_1_transform_1.sh` que recibe como parámetro de entrada el archivo que generaste en la parte 1 de `precios.xml` y genera como salida un archivo `precios.csv` con la siguiente información:

```
estacion_servicio,regular,premium,diesel
11703,20.19,22.89,
11702,20.14,23.35,
11701,16.49,23.35,20.99
```

La información con la generas el archivo `precios.csv` viene del archivo `precios.xml`, toda la transformación requerida debe realizarse en el *script* `proyecto_1_transform_1.sh`

- ¡Ocupa expresiones regulares!, si tu solución no ocupa expresiones regulares tendrás -0.5 en esta pregunta.
- En caso de que no haya precio de alguna de las gasolinas debes dejar el espacio vacío. Por ejemplo, si la estación de servicio `1111` tiene solo el precio de gasolina diesel tendrás que generar la salida: `1111,,,20.44`.

1. Con `grep` averigua ¿Cuántas gasolineras sirven gasolina `regular` en este conjunto de datos? Imprime en terminal `gasolineras que sirven gasolina regular: <z>`. Donde `<z>` corresponde al número de gasolineras.
2. Con `grep` averigua ¿Cuántas gasolineras sirven gasolina `diesel` en este conjunto de datos? Imprime en terminal `gasolineras que sirven gasolina diesel: <y>`. Donde `<y>` corresponde al número de gasolineras.
3. Con `grep` averigua ¿Cuántas gasolineras sirven `premium` en este conjunto de datos? Imprime en terminal `gasolineras que sirven gasolina premium: <x>`
4. Con `AWK` averigua ¿De cuántas gasolineras **diferentes** tienes datos de precios? * Te puedes apoyar de un archivo auxiliar creado con `grep` o `sed`.
   Imprime en terminal `gasolineras diferentes: <m>`. Donde `<m>` corresponde al número de gasolineras.
5. ¿Cuántos renglones de precios de gasolina tienes (una vez que ya tienes 1 renglón por estación de gasolina)? Imprime en terminal `observaciones de precios: <n>`. Donde `<n>` corresponde al número de gasolineras.

Parte 3 - Transform 2 - estaciones.xml

1. Genera un script de bash `proyecto_1_transform_2.sh` que recibe como parámetro de entrada el archivo de `estaciones.xml` y genera como salida un archivo `estaciones.csv` con la siguiente información:

```
id_estacion,nombre,latitud,longitud
2039,estacion de servicio calafia sa de cv,32.47641,-116.9214
2040,las mejores estaciones sa de cv,20.3037,-99.74484
```

- La `x` corresponde a la longitud y la `y` a la latitud ¡Ten cuidado con el orden en el csv!

El contenido del *script* `proyecto_1_transform_2.sh` debe:

2. Con `sed` cambiar a minúsculas el contenido del atributo `name` (sin crear otro archivo).
3. Con `sed` elimina los acentos del contenido del atributo `name` (sin crear otro archivo).
4. Con `sed` elimina signos de puntuación del atributo `name` (sin crear otro archivo). Signos de puntuación: `,;.:-`. Por ejemplo: `estacion rael, s. de r.l. de c.v.` quedaría como `estacion rael s de rl de cv`
5. Con `AWK` averigua ¿De cuántas estaciones diferentes tienes geolocalización? Imprimir en terminal `estaciones diferentes: <n>`

Parte 4 - Load

Genera un script `upload.sql` que contenga los queries necesarios para crear lo siguiente en PostgreSQL:

1. Borre el esquema `raw` si existe (`drop schema if exists raw cascade`)
2. Cree el esquema `raw` (`create schema raw`)
3. Borre la tabla `raw.estaciones` si existe (`drop table if exists raw.estaciones;`)
4. Cree la tabla `estaciones` dentro del esquema `raw` (`create table raw.estaciones(...)`)
5. Borre la tabla `raw.precios` si existe (`drop table if exists raw.precios;`)
6. La tabla `precios` dentro del esquema `raw` (`create table raw.precios(...)`)

Genera un script `proyecto_1_load.sh` que contenga las siguientes instrucciones:

4. Ejecución del script de sql `psql -f updload.sql service=fuentes_datos` (Necesitarás crear el archivo de configuración `~/.pg_service.conf`) Puedes leer de ese archivo aquí
5. El comando para poblar la tabla `raw.precios` -> `psql -c '\copy raw.precios from precios.csv with csv header;' service=fuentes_datos`

6. El comando para poblar la tabla `raw.estaciones` -> `psql -c '\copy raw.estaciones from estaciones.csv with csv header;' service=fuentes_datos`

Parte 5 - Wrap it up

Modifica tu *script* `proyecto_1.sh` para agregar las siguientes partes:

1. Ejecución del *script* de bash `proyecto_1_transform_1.sh` recibiendo como parámetro externo el archivo de `precios.xml`
2. Ejecución del *script* de bash `proyecto_1_transform_2.sh` recibiendo como parámetro externo el archivo de `estaciones.xml`
3. Ejecución del *script* de bash `proyecto_1_load.sh`

Ejemplo de ejecución: `bash proyecto_1.sh`

Notas

El siguente *snippet* de código permite leer un archivo línea por línea a través de `bash`.

```
input=/path/a/tu/archivo
while read -r line
do
done < $input
```

El párametro `-o` en grep permite extraer el substring que cumple con una expresión regular.

Puedes generar otros archivos auxiliares y ocuparlos dentro de tus scripts principales en caso de necesitarlo. En este caso, necesitarás "cargarlos" sin que se envíen como parametros externos.

Parte 6 - Análisis de datos

¿Que se entrega?

- Notebook `analisis.ipynb`

Supuestos

- Tienes una base de datos en postgresql con las siguientes tablas: `raw.estaciones`, `raw.precios` (tienen las mismas columnas que ocupaste en tu proyecto parte 1).
- Tienes un archivo `credentials.yaml` en un directorio `conf` a la misma altura en donde están tu *notebook* `analisis.ipynb` con tus credenciales de bd:

```
---
db:
   user: ""
   pass: ""
   host: ""
   port: "5432"
   db: "fuentes_datos"
```

Parte 7: Análisis con Pandas

1. Ocupa la función `get_db_conn()` a través de la cuál obtenemos una conexión a la BD (postgres) con `psycopg2` leyendo del archivo `credentials.yaml` (utilizando la función `read_yaml()`) las credenciales de acceso a la BD.

2. Genera un *query sql* en donde obtengas el `id_estacion, nombre, longitud, latitud, regular, preimum, diesel` de todos los precios. <- Necesitarás un `inner join`.

En pandas:

- Utilizando `shape`, cuántas observaciones tienes.
- Utilizando `is.na()`:
  - cuántas estaciones no tienen gasolina `regular`
  - cuántas estaciones no tienen gasolina `premium`
  - cuántas estaciones no tienen gasolina `diesel`
  - cuántas estaciones solo tienen gasolina `regular`
  - cuántas estaciones solo tienen gasolina `regular` y `premium`
  - cuántas estaciones solo tienen gasolina `regual` y `diesel`
- qué estaciones (nombre) dieron gasolina `premium` "gratis" de acuerdo a los datos.
- Cuántas estaciones de gasolina venden de los 3 tipos de gasolina.

1. En caso que tus columnas `latitud` y `longitud` no cumplan con el formato de coordenadas geoespaciales: latitud debe estar en una escala del 10 al 99 y longitud debe estar en el rango de 99 al 120; tendrás que convertirlas a ese rango.

- Genera un `describe()`. Interpreta el significado de cada métrica en el `describe` para las columnas `regular, premium, diesel`.
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más caro de gasolina `regular`
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más caro de gasolina `premium`
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más caro de gasolina `diesel`
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más barato de gasolina `regular`
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más barato de gasolina `premium`
- Cuál es la estación de gasolina -nombre y coordenadas- con el precio más barato de gasolina `diesel`

1. CDMX

- Debido a que estamos en CDMX -> verifica las latitudes y longitudes posibles de la CDMX <-, ¿a qué estación de gasolina -nombre y coordenadas- deberíamos ir a llenar el tanque para cada tipo de gasolina? (no importa la distancia)
- De este subconjunto, genera un *data frame* que nos diga ¿cuántas estaciones de gasolina venden cada tipo de gasolina? Toma el tipo de gasolina como independiente, es decir, una estación puede vender de las 3 gasolinas, en este caso aparacerá en cada conteo, tanto para el regular como para el premium como para el diesel.

1. Genera un nuevo *data frame* que junte las columnas `regular, premium, diesel` en dos nuevas columnas: `tipo_gasolina` y `precio`, la primera se llena con el tipo de gasolina: `regular`, `premium`, `diesel` y la segunda con el valor correspondiente al precio.

- Utilizando `shape`, cuantas observaciones tienes.
- Genera un `describe()`. Interpreta el significado de cada métrica en el `describe` para la columna `precio`. ¿Cómo cambian las estadísticas?

Parte 8 Visualización

Seaborn

6. De la pegunta 5. Genera un boxplot (en una sola gráfica) que contenga las distribuciones de los 3 tipos de gasolina. Agrega el promedio. Recuerda poner nombres a los ejes y unidades (si aplica). Cada tipo de gasolina debe estar en un color diferente. Interpreta la gráfica, cuál consideras que es el *insight* más interesante.
7. De la pregunta 5. Genera un solo histograma (en una sola gráfica) con los precios que incluya la distribución de cada tipo de gasolina. Incluye un parámetro `alpha` que te permita agregar transparencia para que se vean las 3 distribuciones en la misma gráfica. Cada tipo de gasolina va de color diferente.

Plotly

8. Genera un mapa que muestre las estaciones de las que tienes registros. ¿Existen estaciones que no están en México? En caso afirmativo, ¿cuántas, qué crees que haya pasado? `plotly.express.scatter_geo`
9. Genera un mapa que muestre las estaciones que venden regular. ¿Existen estaciones que no están en México? En caso afirmativo, ¿cuántas, qué crees que haya pasado?
10. Genera un mapa que muestre las estaciones que venden premium. ¿Existen estaciones que no están en México? En caso afirmativo, ¿cuántas, qué crees que haya pasado?
11. Genera un mapa que muestre las estaciones que venden diesel. ¿Existen estaciones que no están en México? En caso afirmativo, ¿cuántas, qué crees que haya pasado?