Proyecto ETL

Santiago Gomez Castro

Juan Carlos Quintero

MIguel Angel Ruales

Docente: Javier Alejandro Vergara Zorrilla

Universidad Autónoma de Occidente

Facultad de Ingeniería

Santiago de Cali

2024

Eda Api (Jupyter Notebook)

Información del dataset:
Ya habiendo hecho la llamada a la API, podemos verificar las primeras filas del dataset que se nos ha proporcionado.

	period	duoarea	area-name	product	product-name	process	process-name	series	series-description	value	units
0	2024-09-30	R30	PADD 3	EPM0	Total Gasoline	PTE	Retail Sales	EMM_EPM0_PTE_R30_DPG	Gulf Coast All Grades All Formulations Retail ...	2.793	$/GAL
1	2024-09-30	R10	PADD 1	EPD2D	No 2 Diesel	PTE	Retail Sales	EMD_EPD2D_PTE_R10_DPG	East Coast No 2 Diesel Retail Prices (Dollars ...	3.571	$/GAL
2	2024-09-30	R5XCA	PADD 5 EXCEPT CALIFORNIA	EPD2D	No 2 Diesel	PTE	Retail Sales	EMD_EPD2D_PTE_R5XCA_DPG	West Coast (PADD 5) Except California No 2 Die...	3.797	$/GAL
3	2024-09-30	R40	PADD 4	EPMRU	Conventional Regular Gasoline	PTE	Retail Sales	EMM_EPMRU_PTE_R40_DPG	Rocky Mountain Regular Conventional Retail Gas	3.421	$/GAL

Nuestro dataset se compone de las siguientes columnas, siendo principalmente objetos y flotantes en la columna value.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   period              1000 non-null   object 
 1   duoarea             1000 non-null   object 
 2   area-name           1000 non-null   object 
 3   product             1000 non-null   object 
 4   product-name        1000 non-null   object 
 5   process             1000 non-null   object 
 6   process-name        1000 non-null   object 
 7   series              1000 non-null   object 
 8   series-description  1000 non-null   object 
 9   value               988 non-null    float64
 10  units               1000 non-null   object 
dtypes: float64(1), object(10)
memory usage: 86.1+ KB
```

Ahora procedemos a realizar un resumen estadístico de nuestras columnas, comenzando por la única numérica (Value):

	value
count	988.000000
mean	3.695623
std	0.563167
min	2.605000
25%	3.239500
50%	3.612000
75%	4.053500
max	5.395000

En cuanto a las categorías tenemos:

	period	duoarea	area-name	product	product-name	process	process-name	series	series-description	units
count	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
unique	4	29	29	14	14	1	1	310	310	1
top	2024-09-30	R5XCA	PADD 5 EXCEPT CALIFORNIA	EPM0	Total Gasoline	PTE	Retail Sales	EMM_EPMR_PTE_SMN_DPG	Minnesota Regular All Formulations Retail Gaso...	$/GAL
freq	310	46	46	97	97	1000	1000	4	4	1000

Valores en las columnas:
Para entender mejor el contenido del dataset, proseguimos a buscar la frecuencia de los valores de las columnas:

Period

```
df['period'].value_counts()
```

```
period
2024-09-30    310
2024-09-23    310
2024-09-16    310
2024-09-09     70
Name: count, dtype: int64
```

Duoarea y Area-name

```
df['duoarea'].value_counts()
```

```
R5XCA    46
R40      46
R20      46
R50      45
R30      45
R1Z      45
R1Y      44
NUS      44
R1X      44
R10      42
STX      39
SNY      38
YDEN     38
SCO      38
SCA      32
```

```
df['area-name'].value_counts()
```

```
area-name
PADD 5 EXCEPT CALIFORNIA    46
PADD 4                      46
PADD 2                      46
PADD 5                      45
PADD 3                      45
PADD 1C                     45
PADD 1B                     44
U.S.                        44
PADD 1A                     44
PADD 1                      42
TEXAS                       39
NEW YORK                    38
DENVER                      38
COLORADO                    38
CALIFORNIA                  32
```

El código "R5XCA" tiene un valor asociado de 46, que coincide con el área "PADD 5 EXCEPT CALIFORNIA" en la columna "area-name".
Algunas posibilidades que podemos considerar para descifrar esto (o darle sentido) son:

R5 = PADD 5
X = EXCEPT (EXCEPTO)
CA = CALIFORNIA
Este patrón se repite con otros códigos y sus respectivos nombres de área.
Por ejemplo:
R50 = 45 <-> PADD 5 = 45

Product y Product-name

```
df['product'].value_counts()
```

product	
EPM0	97
EPMM	96
EPMP	94
EPMR	91
EPMMR	75
EPMRR	74
EPM0R	73
EPMMU	69
EPMPR	68
EPMPU	66
EPM0U	66
EPMRU	64
EPD2DXL0	34
EPD2D	33

```
df['product-name'].value_counts()
```

product-name	
Total Gasoline	97
Midgrade Gasoline	96
Premium Gasoline	94
Regular Gasoline	91
Gasoline Reformulated Midgrade	75
Reformulated Regular Gasoline	74
Reformulated Motor Gasoline	73
Gasoline Conventional Midgrade	69
Reformulated Premium Gasoline	68
Conventional Premium Gasoline	66
Conventional Gasoline (No Oxy)	66
Conventional Regular Gasoline	64
No 2 Diesel Low Sulfur (0-15 ppm)	34
No 2 Diesel	33

Process y Process-name

```
df['process'].value_counts()
```

process	
PTE	1000

```
df['process-name'].value_counts()
```

process-name	
Retail Sales	1000

Series y Series-name

```
df['series'].value_counts()
```

series	
EMM_EPMR_PTE_SMN_DPG	4
EMM_EPMRR_PTE_R50_DPG	4
EMM_EPM0_PTE_YMIA_DPG	4
EMM_EPM0R_PTE_Y05SF_DPG	4
EMM_EPMM_PTE_SMA_DPG	4
..	..
EMM_EPMPU_PTE_R30_DPG	3
EMM_EPMPR_PTE_YBOS_DPG	3
EMM_EPMP_PTE_Y05LA_DPG	3
EMM_EPMP_PTE_SNY_DPG	3
EMM_EPMRU_PTE_R40_DPG	3

```
df['series-description'].value_counts()
```

series-description	
Minnesota Regular All Formulations Retail Gasoline Prices (Dollars per Gallon)	4
West Coast Regular Reformulated Retail Gasoline Prices (Dollars per Gallon)	4
Miami, FL All Grades All Formulations Retail Gasoline Prices (Dollars per Gallon)	4
San Francisco, CA All Grades Reformulated Retail Gasoline Prices (Dollars per Gallon)	4
Massachusetts Midgrade All Formulations Retail Gasoline Prices (Dollars per Gallon)	4
..	..
Gulf Coast Premium Conventional Retail Gasoline Prices (Dollars per Gallon)	3
Boston, MA Premium Reformulated Retail Gasoline Prices (Dollars per Gallon)	3
Los Angeles Premium All Formulations Retail Gasoline Prices (Dollars per Gallon)	3
New York Premium All Formulations Retail Gasoline Prices (Dollars per Gallon)	3
Rocky Mountain Regular Conventional Retail Gasoline Prices (Dollars per Gallon)	3

"series" también es una codificación de en este caso, Series-name. Encontramos que esta columna agrupa de manera muy acertada lo que representa cada registro, agrupando información de area, producto y unidades. Series, al igual que las otras columnas codificadas son redundantes y podemos borrarlas o darles otra utilidad.

Prices y Unit

```
df['value'].value_counts()
```

```
value
3.118    7
3.844    6
3.818    5
3.105    5
3.093    5
        ..
4.766    1
4.078    1
4.654    1
3.995    1
3.071    1
```

```
df['units'].value_counts()
```

```
units
$/GAL    1000
```

ESPACIO PARA LAS VISUALIZACIONES

Transformaciones

Después de haber analizado ampliamente el dataset, podemos empezar con las transformaciones.

Borrar / Renombrar columnas:

```
#Drop columns
drop_columns = ['duoarea', 'units', 'series']
df = df.drop(columns=drop_columns)

#Rename columns (value)
df = df.rename(columns={'value': 'value($/GAL)'})

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   period              1000 non-null   object
 1   area-name           1000 non-null   object
 2   product             1000 non-null   object
 3   product-name        1000 non-null   object
 4   process             1000 non-null   object
 5   process-name        1000 non-null   object
 6   series-description  1000 non-null   object
 7   value($/GAL)        988 non-null    float64
dtypes: float64(1), object(7)
memory usage: 62.6+ KB
None
```

Cambios en el formato/tipo de columnas:

```python
#Correct the types:
df['period'] = pd.to_datetime(df['period'], format='%Y-%m-%d') #Object to date
df = df.astype({col: 'string' for col in df.select_dtypes(include='object').columns}) #Object to string

#We check the changes
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   period              1000 non-null   datetime64[ns]
 1   area-name           1000 non-null   string
 2   product             1000 non-null   string
 3   product-name        1000 non-null   string
 4   process             1000 non-null   string
 5   process-name        1000 non-null   string
 6   series-description  1000 non-null   string
 7   value($/GAL)        988 non-null    float64
dtypes: datetime64[ns](1), float64(1), string(6)
memory usage: 62.6 KB
None
```

Realizamos limpieza y reemplazamos datos

Area-name

```python
replaces = {
    'PADD 5 EXCEPT CALIFORNIA': 'West Coast (except California)','PADD 4': 'Rocky Mountain',
    'PADD 2': 'Midwest','PADD 5': 'West Coast','PADD 3': 'Gulf Coast','PADD 1C': 'East Coast (Central)',
    'PADD 1B': 'East Coast (North)','PADD 1A': 'East Coast (South)','PADD 1': 'East Coast'
} #Create a dictionary to replace the PADD values to a more explicit name
df['area-name'] = df['area-name'].replace(replaces) #Replaces
```

New Area column

```python
#Make a list for the codes to know if an area is city/state/region
city_list = ['DENVER', 'NEW YORK CITY', 'SAN FRANCISCO', 'MIAMI', 'CLEVELAND',
             'CHICAGO', 'SEATTLE', 'HOUSTON', 'LOS ANGELES', 'BOSTON']

state_list = ['TEXAS', 'NEW YORK', 'COLORADO', 'CALIFORNIA', 'MINNESOTA', 'FLORIDA', 'MASSACHUSETTS',
              'WASHINGTON', 'OHIO']

region_list = ['West Coast (except California)', 'Rocky Mountain', 'Midwest', 'West Coast',
               'Gulf Coast', 'East Coast (Central)', 'East Coast (North)', 'U.S.',
               'East Coast (South)', 'East Coast']

#Create the column 'area' based in 'area-name' values (If they are reffering to a city/state/region)
df['area'] = np.where(df['area-name'].isin(city_list), 'City',
                np.where(df['area-name'].isin(state_list), 'State',
                np.where(df['area-name'].isin(region_list), 'Region', df['area-name'])))
```

```
df['area-name'].value_counts()
```

```
area-name
West Coast (except California)    46
Rocky Mountain                   46
Midwest                          46
West Coast                       45
Gulf Coast                       45
East Coast (Central)             45
East Coast (North)               44
U.S.                             44
East Coast (South)               44
East Coast                       42
TEXAS                            39
```

```
df['area'].value_counts()
```

```
area
Region    447
City      281
State     272
```

- Los PADD (ahora reemplazados por otros nombres) cubren completamente los 50 estados de EE. UU...
- PADD 1 (Costa Este):
- Costa Este (Sur): incluye estados como Florida, Georgia, Carolina del Norte y Carolina del Sur.
- Costa Este (Central): incluye los estados del Atlántico Medio como Nueva York, Nueva Jersey y Pensilvania.
- Costa Este (Norte): incluye estados de Nueva Inglaterra, como Maine, Vermont, Massachusetts, etc.
- Por lo tanto, el PADD 1 cubre los estados del Noreste, el Atlántico Medio y el Sureste.
- PADD 2 (Medio Oeste): incluye los estados del Medio Oeste, como Illinois, Ohio, Michigan, Indiana, Wisconsin, Minnesota, entre otros.
- PADD 3 (Costa del Golfo): incluye los estados de la Costa del Golfo, como Texas, Luisiana, Misisipi, Alabama.
- PADD 4 (Montaña Rocosa): incluye los estados montañosos como Colorado, Utah, Wyoming, Montana, Idaho.
- PADD 5 (Costa Oeste): incluye los estados de la Costa Oeste, como California, Oregón, Washington.
- PADD 5 EXCEPTO CALIFORNIA: es la misma región pero excluye a California y puede incluir a Oregón y Washington.

Product

We will change this column to use it solely to define which product it refers to (in this case, gasoline or diesel).

```
#Make a list for the codes to gasoline/diesel
gasoline_codes = ['EPM0', 'EPMM', 'EPMP', 'EPMR', 'EPMMR', 'EPMRR', 'EPM0R', 'EPMMU', 'EPMPR','EPMPU', 'EPM0U', 'EPMRU']
diesel_codes = ['EPD2DXL0', 'EPD2D']

#Replaces
df['product'] = np.where(df['product'].isin(gasoline_codes), 'Gasoline',
                np.where(df['product'].isin(diesel_codes), 'Diesel', df['product']))
```

✓ 0.0s Python

```
print(df['product'].value_counts())
```

✓ 0.0s Python

```
product
Gasoline    921
Diesel       67
Name: count, dtype: int64
```

```
print(df['product-name'].value_counts())
```

✓ 0.0s

```
product-name
Total Gasoline                      97
Midgrade Gasoline                   96
Premium Gasoline                    94
Regular Gasoline                    91
Gasoline Reformulated Midgrade      75
Reformulated Regular Gasoline       74
Reformulated Motor Gasoline         73
Reformulated Premium Gasoline       68
Gasoline Conventional Midgrade      66
Conventional Premium Gasoline       63
Conventional Gasoline (No Oxy)      63
Conventional Regular Gasoline       61
No 2 Diesel Low Sulfur (0-15 ppm)   34
No 2 Diesel                         33
```

```
print(df['product'].value_counts())
```

✓ 0.0s

```
product
Gasoline    921
Diesel       67
```

Nulls cleaning

Value has some nulls, so we will delete those rows.

```
df.dropna(subset=['value($/GAL)'], inplace=True) #No nulls
```

✓ 0.0s

Así nos resulta nuestro dataset después de los cambios implementados:

```
df.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 988 entries, 0 to 999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   period              988 non-null    datetime64[ns]
 1   area-name           988 non-null    string
 2   product             988 non-null    object
 3   product-name        988 non-null    string
 4   process             988 non-null    string
 5   process-name        988 non-null    string
 6   series-description  988 non-null    string
 7   value($/GAL)        988 non-null    float64
 8   area                988 non-null    object
```

Fact-dimensions (Jupyter notebook)

Realizaremos la creación del modelo dimensional, con dos tablas de hechos: Una para la venta de los carros (Base de datos original/De la primera entrega) y otra para los precios del petróleo al pormenor (API).

1. **Venta de carros**

Dimensiones:

Car dim:

Car

```
# Car dim columns
car_dim = df[['Year', 'Make', 'Model', 'Drivetrain', 'MinMPG', 'MaxMPG',
              'FuelType', 'Transmission', 'Engine', 'ExteriorColor', 'InteriorColor',
              'Used', 'VIN', 'Stock#']].drop_duplicates().reset_index(drop=True)
# ID
car_dim['ID_Car'] = car_dim.index + 1

car_dim.head()
```
✓ 0.2s

Year	Make	Model	Drivetrain	MinMPG	MaxMPG	FuelType	Transmission	Engine	ExteriorColor	InteriorColor	Used	VIN	Stock#	ID_Car
2019	Toyota	Sienna SE	FWD	19	27	Gasoline	Automatic	3.5L V6 24V PDI DOHC	Red	Black	True	5TDXZ3DC2KS015402	22998646	1
2018	Ford	F-150 Lariat	4WD	19	24	Gasoline	Automatic	3.5L V6 24V PDI DOHC Twin Turbo	Shadow Black	Black	True	1FTEW1EG2JFD44217	22418A	2
2017	Ram	1500 Laramie	4WD	15	21	Gasoline	Automatic	5.7L V8 16V MPFI OHV	Granite Crystal Clearcoat Metallic	Black	True	1C6RR7VT5HS842283	NG277871G	3
2021	Honda	Accord Sport SE	FWD	29	35	Gasoline	CVT	1.5L I4 16V GDI DOHC Turbo	Gray	Â□□	True	1HGCV1F49MA038035	54237	4

Seller dim:

Seller

```
# Seller dim columns
seller_dim = df[['SellerName', 'SellerType', 'State', 'Zipcode', 'StreetName']].drop_duplicates().reset_index(drop=True)
#ID
seller_dim['ID_Seller'] = seller_dim.index + 1

seller_dim.head()
```

✓ 0.0s

	SellerName	SellerType	State	Zipcode	StreetName	ID_Seller
0	CarMax Murrieta - Now offering Curbside Pickup...	Dealer	CA	92562	25560 Madison Ave Murrieta	1
1	Giant Chevrolet	Dealer	CA	93292	1001 S Ben Maddox Way Visalia	2
2	Gill Auto Group Madera	Dealer	CA	93637	1100 S Madera Ave Madera	3
3	AutoSavvy Las Vegas	Dealer	NV	89104	2121 E Sahara Ave Las Vegas	4
4	Lexus of Henderson	Dealer	NV	89011	7737 Eastgate Rd Henderson	5

Rating dim:

Ratings

```
# Rating dim columns
rating_dim = df[['ConsumerRating', 'SellerRating', 'ComfortRating', 'InteriorDesignRating',
                 'PerformanceRating', 'ValueForMoneyRating', 'ExteriorStylingRating',
                 'ReliabilityRating', 'DealType']].drop_duplicates().reset_index(drop=True)
# ID
rating_dim['ID_Rating'] = rating_dim.index + 1

# Mostrar la dimensión de ratings
rating_dim.head()
```

✓ 0.1s Python

ConsumerRating	SellerRating	ComfortRating	InteriorDesignRating	PerformanceRating	ValueForMoneyRating	ExteriorStylingRating	ReliabilityRating	DealType	ID_Rating
4.6	3.3	4.7	4.6	4.6	4.4	4.6	4.7	Great	1
4.8	4.8	4.9	4.8	4.8	4.6	4.8	4.7	Good	2
4.7	4.6	4.8	4.7	4.8	4.6	4.8	4.7	Good	3
5.0	4.6	4.9	5.0	4.9	5.0	5.0	5.0	NaN	4
4.8	4.8	4.9	4.8	4.8	4.7	4.8	4.9	Good	5

Tabla de hechos:

Sells fact table:

Sells fact table

```python
# Merge original df with dimensions to asign IDs
df_hechos_vendedor = pd.merge(df, seller_dim, on=['SellerName', 'SellerType', 'State', 'Zipcode', 'StreetName'], how='left')

df_hechos_vehiculo = pd.merge(df, car_dim, on=['Year', 'Make', 'Model', 'Drivetrain', 'MinMPG', 'MaxMPG',
                                               'FuelType', 'Transmission', 'Engine' ,'ExteriorColor', 'InteriorColor',
                                               'Used', 'VIN', 'Stock#'], how='left')

df_hechos_ratings = pd.merge(df, rating_dim, on=['ConsumerRating', 'SellerRating', 'ComfortRating','InteriorDesignRating',
                                                 'PerformanceRating','ValueForMoneyRating', 'ExteriorStylingRating',
                                                 'ReliabilityRating', 'DealType'], how='left')

#Fact table columns
tabla_hechos = df_hechos_vendedor[['Price', 'Mileage', 'ConsumerReviews', 'SellerReviews']].copy()

# Add IDs from dimensions
tabla_hechos['ID_Car'] = df_hechos_vehiculo['ID_Car']
tabla_hechos['ID_Rating'] = df_hechos_ratings['ID_Rating']
tabla_hechos['ID_Seller'] = df_hechos_vendedor['ID_Seller']

# Sell ID
tabla_hechos['ID_Sell'] = df.index + 1

#Reorder the columns
tabla_hechos = tabla_hechos[['ID_Sell', 'ID_Car', 'ID_Seller', 'ID_Rating', 'Price',
                             'Mileage', 'ConsumerReviews', 'SellerReviews']]
```

```python
tabla_hechos.head(3)
```

✓ 0.0s

	ID_Sell	ID_Car	ID_Seller	ID_Rating	Price	Mileage	ConsumerReviews	SellerReviews
0	1	1	1	1	39998	29403	45	3
1	2	2	2	2	49985	32929	817	131
2	3	3	3	3	41860	23173	495	249

2. Petroleo (API)

Dimensiones:

Area dim:

```python
# Area dim columns
area_dim = apidf[['area', 'area-name']].drop_duplicates().reset_index(drop=True)
#ID
area_dim['area_ID'] = area_dim.index + 1

area_dim.head()
```

✓ 0.0s

	area	area-name	area_ID
0	Region	Gulf Coast	1
1	Region	East Coast	2
2	Region	West Coast (except California)	3
3	Region	Rocky Mountain	4
4	Region	East Coast (South)	5

Product dim:

```python
# Product dim
product_dim = apidf[['product', 'product-name']].drop_duplicates().reset_index(drop=True)
#ID
product_dim['product_ID'] = product_dim.index + 1

product_dim.head()
```

0.0s

product	product-name	product_ID
Gasoline	Total Gasoline	1
Diesel	No 2 Diesel	2
Gasoline	Conventional Regular Gasoline	3
Gasoline	Reformulated Regular Gasoline	4
Gasoline	Regular Gasoline	5

Details dim:

```python
# Details dim columns
details_dim = apidf[['process', 'process-name', 'series-description']].drop_duplicates().reset_index(drop=True)
#ID
details_dim['details_ID'] = details_dim.index + 1

details_dim.head()
```

0.0s

process	process-name	series-description	details_ID
PTE	Retail Sales	Gulf Coast All Grades All Formulations Retail ...	1
PTE	Retail Sales	East Coast No 2 Diesel Retail Prices (Dollars ...	2
PTE	Retail Sales	West Coast (PADD 5) Except California No 2 Die...	3
PTE	Retail Sales	Rocky Mountain Regular Conventional Retail Gas...	4
PTE	Retail Sales	New England (PADD 1A) Regular Conventional Ret...	5

Tabla de hechos

Fuel fact table:

Fuel Fact

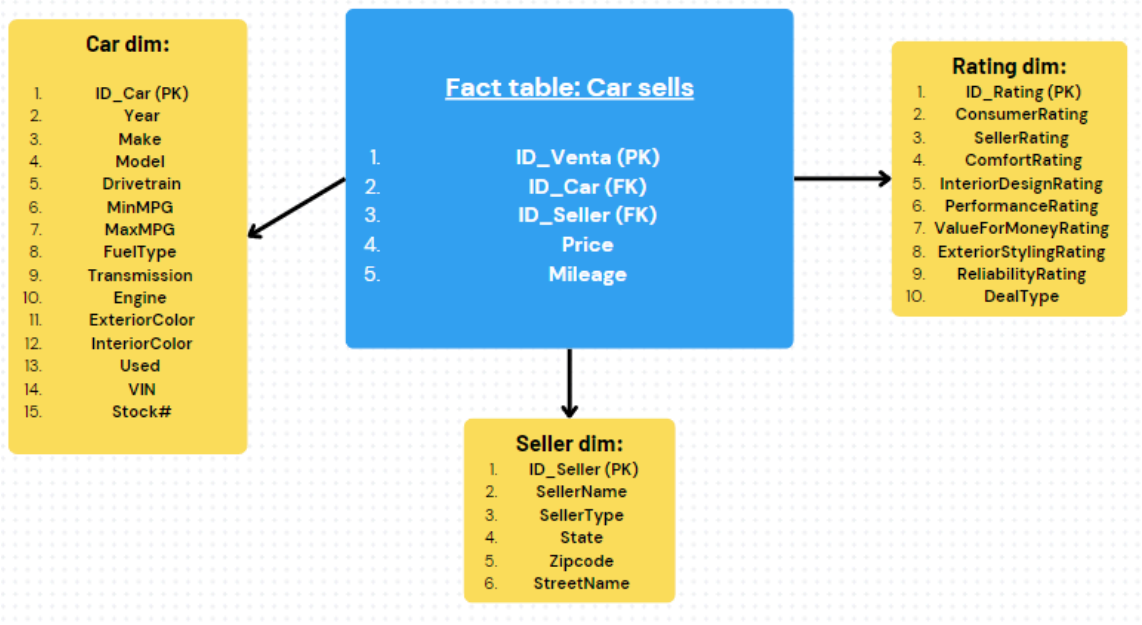
```python
# Merge original df with dimensions to asign IDs
df_fuel_area = pd.merge(apidf, area_dim, on=['area', 'area-name'], how='left')
df_fuel_product = pd.merge(apidf, product_dim, on=['product', 'product-name'], how='left')
df_fuel_details = pd.merge(apidf, details_dim, on=['process', 'process-name', 'series-description'], how='left')

#Fact table columns
fuel_fact = df_fuel_area[['period', 'value($/GAL)']].copy()

# Add IDs from dimensions
fuel_fact['area_ID'] = df_fuel_area['area_ID']
fuel_fact['product_ID'] = df_fuel_product['product_ID']
fuel_fact['details_ID'] = df_fuel_details['details_ID']

# Fuel ID
fuel_fact['fuel_ID'] = fuel_fact.index + 1

#Reorder the columns
fuel_fact = fuel_fact[['fuel_ID', 'period', 'area_ID', 'product_ID', 'details_ID', 'value($/GAL)']]
```

```
fuel_fact.head()
```
✓ 0.0s

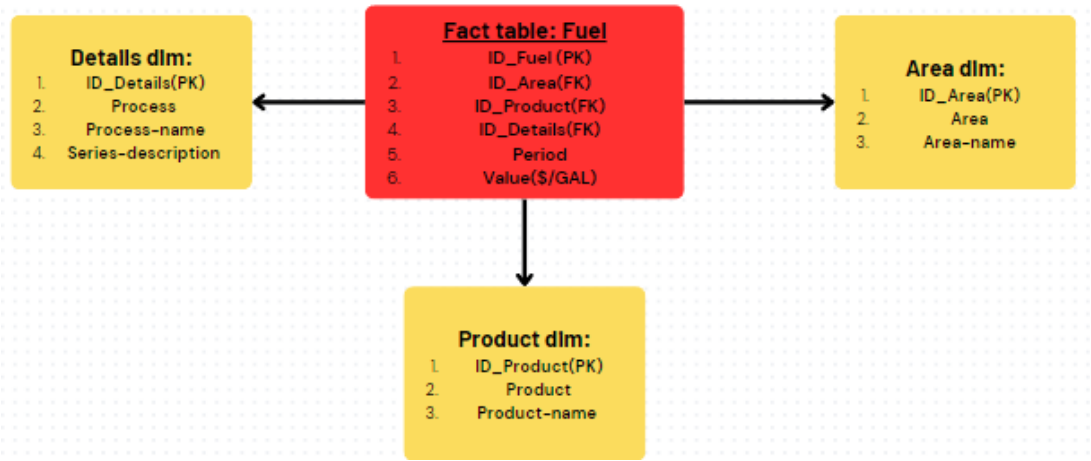
	fuel_ID	period	area_ID	product_ID	details_ID	value($/GAL)
0	1	2024-09-30	1	1	1	2.793
1	2	2024-09-30	2	2	2	3.571
2	3	2024-09-30	3	2	3	3.797
3	4	2024-09-30	4	3	4	3.421
4	5	2024-09-30	5	3	5	3.058

Diagrama modelo dimensional:

En torno a la tabla de hechos para ventas de carros:

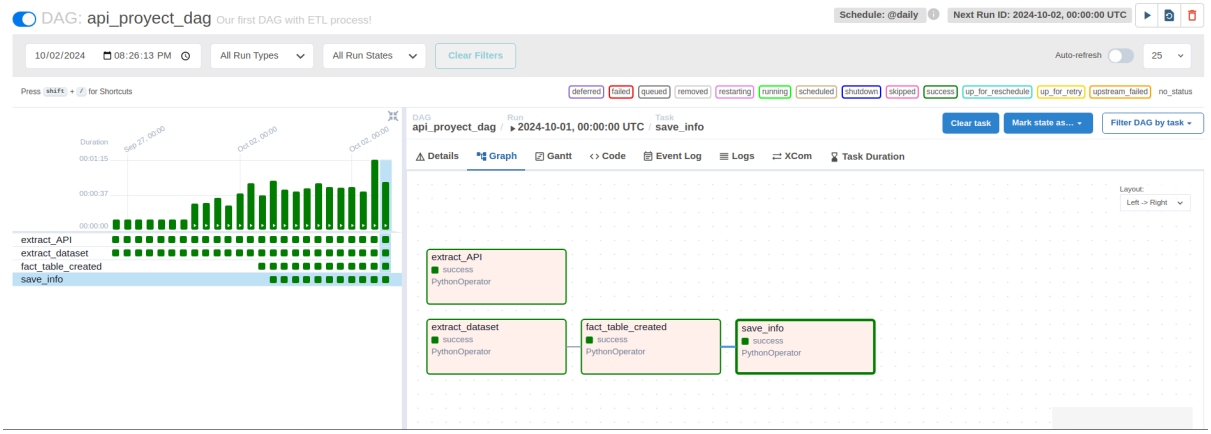


En torno a la tabla de hechos para combustible:

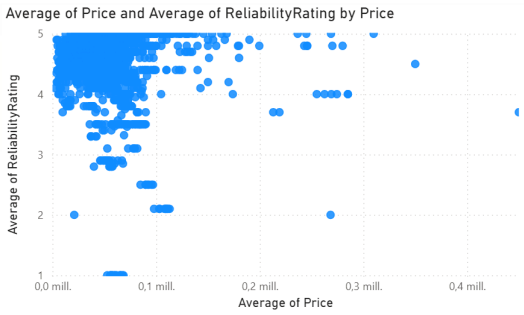


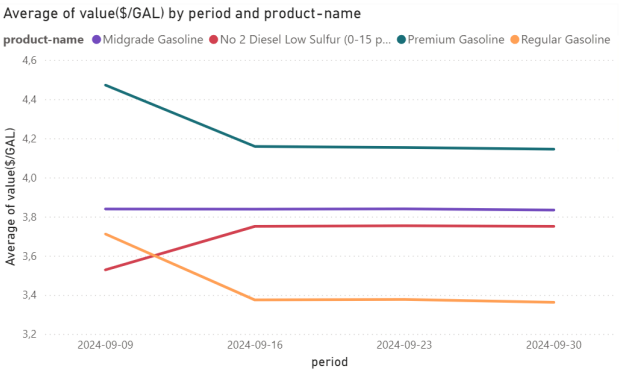
Airflow:

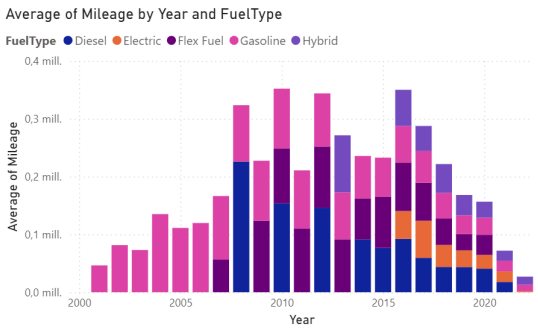
Airflow utiliza unos procesos llamados Dagas donde hay porciones de código que cumplen con diferentes tareas, usamos el dag de extracción de los datos, ambos desde la base de datos Postgresql, iniciamos con el dag de creación de las tablas de hechos y dimensionales para que el último dag suba toda esta información nuevamente a la base de datos Postgresql.

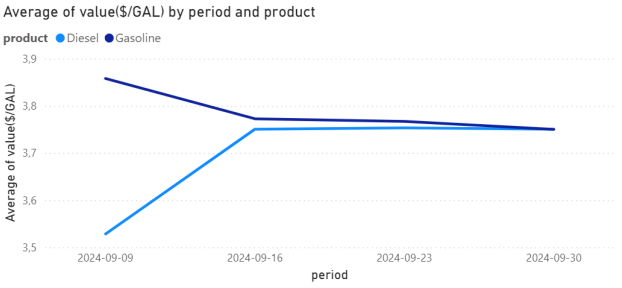


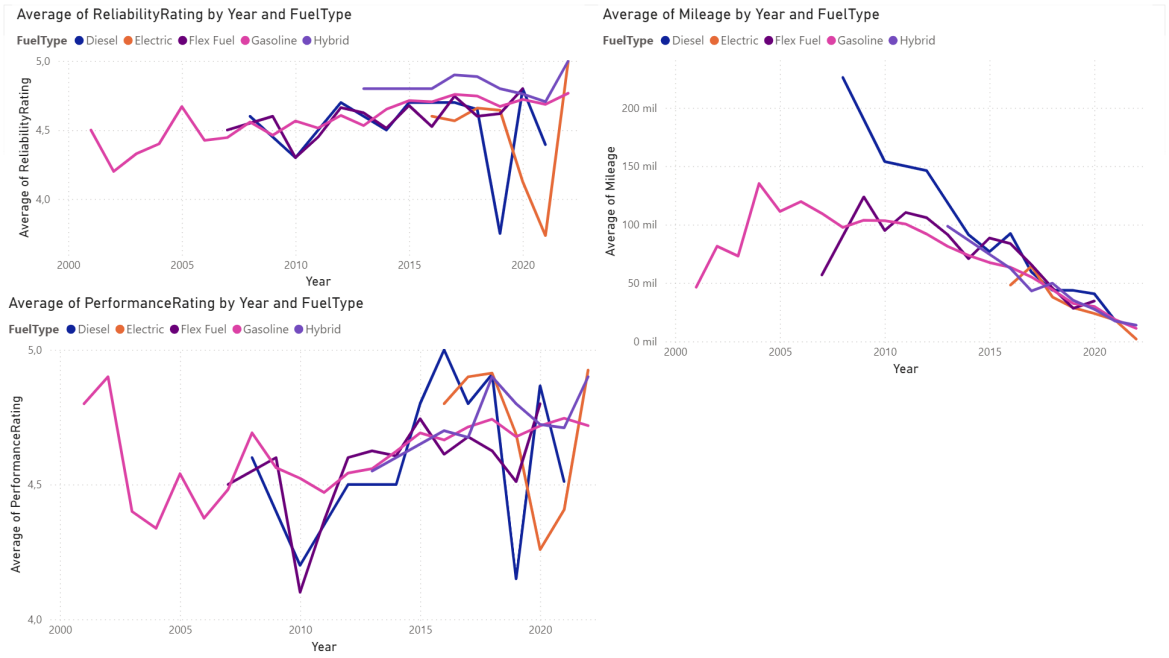
Dashboard:











1. Relación entre el Precio y el Índice de Confiabilidad (Average of Price and Average of ReliabilityRating by Price)

En el gráfico de dispersión que muestra la relación entre el precio promedio y el índice de confiabilidad, se observa una tendencia clara donde la mayoría de los vehículos con precios más bajos presentan índices de confiabilidad entre 4 y 5 puntos. Sin embargo, conforme el precio de los vehículos aumenta, no parece haber una correlación directa que garantice una mayor confiabilidad, lo que sugiere que el precio no es el único factor que influye en la confiabilidad del vehículo.

2. Variación del Precio del Combustible por Periodo y Tipo de Producto (Average of value($/GAL) by period and product-name)

Los gráficos que muestran la variación del precio promedio de diferentes tipos de combustible a lo largo de un periodo específico revelan una tendencia a la baja en los precios del combustible Premium y Regular. El Diesel y la Gasolina de Grado Medio presentan una tendencia estable en los precios. Este comportamiento puede estar vinculado a fluctuaciones en los precios globales del petróleo, la oferta y la demanda, o políticas económicas en el periodo analizado.

3. Promedio de Kilometraje por Año y Tipo de Combustible (Average of Mileage by Year and FuelType)

El gráfico de barras que muestra el promedio de kilometraje por tipo de combustible a lo largo de los años indica un aumento constante en el uso de combustibles como la gasolina y el diésel hasta el año 2015, seguido de una disminución progresiva hasta el 2020. Este comportamiento puede estar relacionado con la creciente

popularidad de vehículos eléctricos e híbridos. Las barras indican que, antes de 2015, los vehículos de gasolina y diésel tenían una mayor participación en términos de kilometraje promedio.

4. Confiabilidad Promedio por Año y Tipo de Combustible (Average of ReliabilityRating by Year and FuelType)

El gráfico de líneas que compara el índice de confiabilidad promedio de los diferentes tipos de combustible a lo largo de los años muestra que los vehículos a gasolina y diésel han mantenido una estabilidad en sus índices de confiabilidad, mientras que los vehículos eléctricos han tenido fluctuaciones más significativas, con una tendencia reciente a aumentar su confiabilidad. Esto podría deberse a avances tecnológicos en los vehículos eléctricos o una mayor inversión en su desarrollo.

5. Desempeño Promedio por Año y Tipo de Combustible (Average of PerformanceRating by Year and FuelType)

El gráfico de desempeño promedio muestra una variabilidad más alta en los vehículos eléctricos y flexibles hasta el año 2010, pero después de ese periodo, se estabiliza. La gasolina y el diésel presentan una línea más estable en cuanto a su desempeño promedio, lo que podría estar vinculado a la madurez de estas tecnologías.

6. Tendencia del Kilometraje Promedio por Año (Average of Mileage by Year and FuelType)

En el gráfico que mide el kilometraje promedio por año y tipo de combustible, se puede ver una clara tendencia a la disminución en el kilometraje promedio de vehículos a diésel, especialmente después de 2010. Esto puede estar relacionado con la regulación de emisiones y la adopción de vehículos más eficientes o alternativos, como los híbridos y eléctricos, que van tomando mayor relevancia en los años recientes.

7. Conclusiones

En resumen, el análisis de los datos revela varias tendencias interesantes sobre el mercado de vehículos, destacando la importancia de la eficiencia y la confiabilidad a lo largo del tiempo. Las fluctuaciones en el mercado del combustible y la evolución de la tecnología automotriz (en particular los vehículos eléctricos e híbridos) han generado un impacto considerable tanto en el kilometraje como en el índice de confiabilidad de los diferentes tipos de vehículos. PostgreSQL se utilizó para almacenar estos datos y Power BI para facilitar su análisis y visualización de manera clara.