Evidencia_1

March 15, 2024

1 Evidencia 1. Flujo de trabajo reproducible del proyecto de ciencia de datos

Pandalytics - Equipo 1

- **A00832444** | Andrea Garza
- **A01197991** | Hiram Maximiliano Muñoz Ramírez
- **A00517124** | Erick Orlando Hernández Vallejo
- **A01197655** | Raúl Isaí Murillo Alemán
- **A01235692** | David Gerardo Martíne Hidrogo

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model

%matplotlib inline
```

[2]:
```
df = pd.read_excel('ternium-data/Exp_2_Pintura_Estructura Analisis de consumo y␣
↪.xlsx', sheet_name='Base de datos por pintura')
df
```

[2]:

	Linea	Mes 2	Mes num	Mes	Pintura
0	Pintado 1	Abril	4	Abril	0226-BREATHTAKING BLUE FC
1	Pintado 1	Abril	4	Abril	0121-BLANCO STD KYNAR
2	Pintado 1	Abril	4	Abril	0919-ROJO TERNIUM CR
3	Pintado 1	Abril	4	Abril	1487-AMERICAN STERLING III
4	Pintado 1	Abril	4	Abril	0435-APOLLO GRAY KRYSTAL KOTE
..	…	…	…	…	…
561	Pintado 2	Agosto	8	Agosto	0099-CLEAR P/ESPUMA ROHS
562	Pintado 2	Agosto	8	Agosto	0075-CLEAR LAUNDRY BACKER ROHS
563	Pintado 2	Agosto	8	Agosto	0023-CLEAR EPOXICO P/ESPUMA
564	Pintado 2	Agosto	8	Agosto	1028-CLEAR BLUE EPOXY

```
565  Pintado 2  Agosto        8  Agosto        0034-CLEAR AZUL P/ESPUMA

        Real         Teo        Dif   Magnitud  Error de Consumo  \
0     203.25   185.69356   17.55644   17.55644               NaN
1    1412.75  1690.59626 -277.84626  277.84626               NaN
2     487.36   472.13485   15.22515   15.22515               NaN
3     760.00   729.61405   30.38595   30.38595               NaN
4    6906.00  7242.44193 -336.44193  336.44193               NaN
..       …         …          …          …                …
561  5866.75  6436.95611 -570.20611  570.20611               NaN
562  1444.00   995.55940  448.44060  448.44060               NaN
563  1073.25  1184.15414 -110.90414  110.90414               NaN
564   200.00   191.27243    8.72757    8.72757               NaN
565     0.00    18.37746  -18.37746   18.37746               NaN

     Rendimeinto Std  Metros cuadrados reales  Rendimeinto Real  \
0          24.146982              4483.939000         22.061200
1          20.997375             39047.893000         27.639634
2          22.572178              3111.390000          6.384172
3          26.404494             18777.002000         24.706582
4          26.666667            187253.169513         27.114563
..               …                       …                …
561        40.944882            293525.200300         50.031994
562        77.165354             86956.173000         60.218956
563        39.370079             56419.297265         52.568644
564        42.519685              9389.738000         46.948690
565        39.370079               818.960000          0.000000

     Diferencia de Rendimiento  Diferencia porcentual de rendimiento
0                    -2.085781                                  NaN
1                     6.642259                                  NaN
2                   -16.188007                                  NaN
3                    -1.697913                                  NaN
4                     0.447896                                  NaN
..                         …                                   …
561                   9.087112                                  NaN
562                 -16.946398                                  NaN
563                  13.198565                                  NaN
564                   4.429005                                  NaN
565                 -39.370079                                  NaN

[566 rows x 15 columns]
```

[3]:
```python
def IQR(df: pd.DataFrame, col_name: str):
    column = df[col_name]
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
```

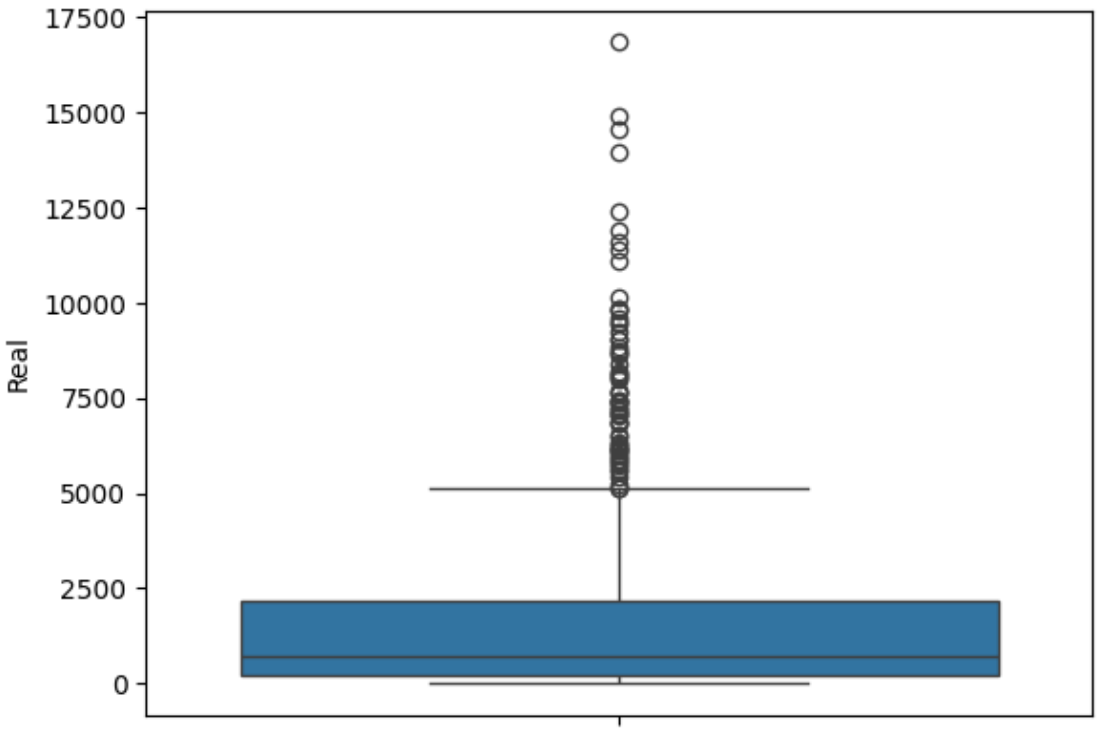
```
    IQR = Q3 - Q1
    print('Rango de valores:', Q1 - 1.5*IQR, 'a', Q3 + 1.5*IQR, '\n')
    print('Valores atípicos: ')
    sns.boxplot(data=column)
    return df[(column < Q1 - 1.5 * IQR) | (column > Q3 + 1.5 * IQR)][col_name]
```

[4]:
```
IQR(df, 'Real')
```

```
Rango de valores: -2697.0281250000003 a 5120.216875

Valores atípicos:
```

[4]:
```
4       6906.000
11      5420.000
43      5770.000
53      8627.000
55      6526.825
          ...
544     5587.000
546     8120.860
549    13953.000
559     5862.970
561     5866.750
Name: Real, Length: 66, dtype: float64
```

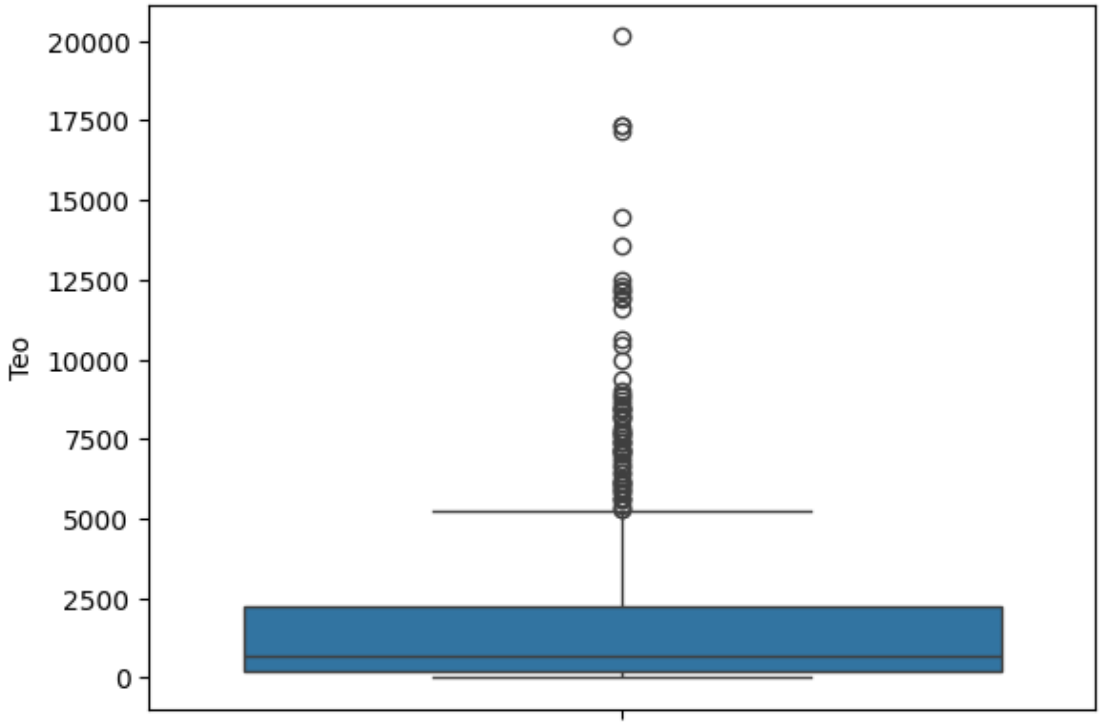


[5]:
```
IQR(df, 'Teo')
```

```
Rango de valores: -2792.4401850000004 a 5252.155595

Valores atípicos:
```

[5]:
```
4       7242.44193
11      5969.16377
43      5631.75768
53      8411.72570
55      7047.02210
          …
544     6933.50861
546     9979.68031
549    17318.40716
559     7124.01670
561     6436.95611
Name: Teo, Length: 67, dtype: float64
```

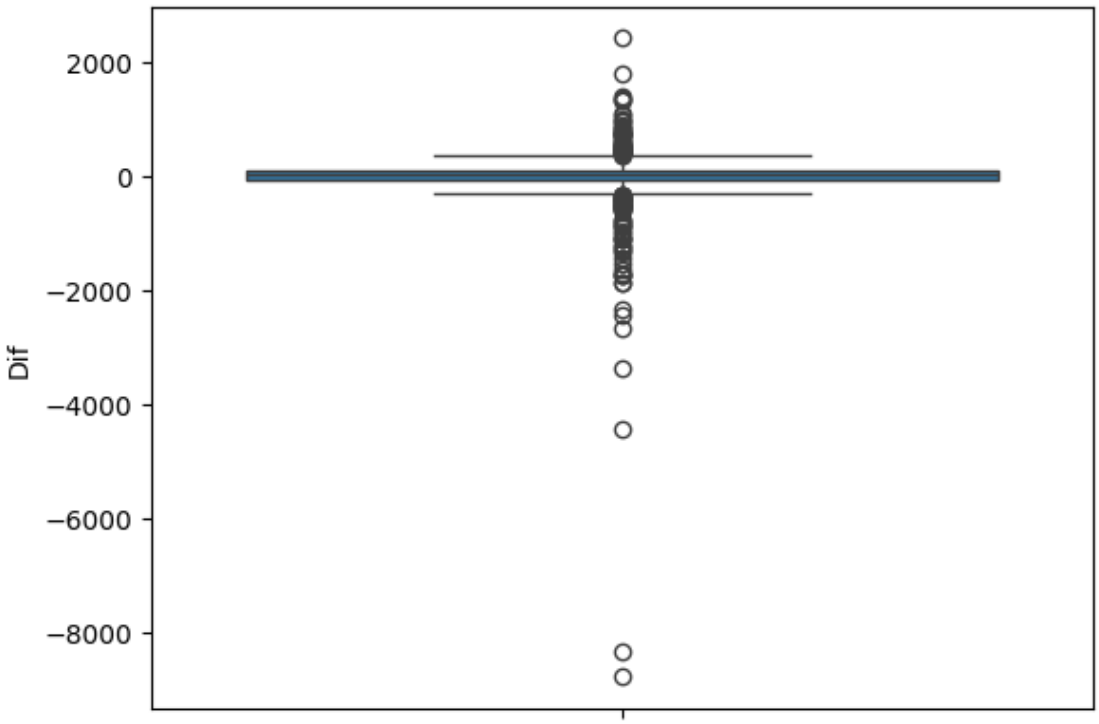


[6]:
```
IQR(df, 'Dif')
```

```
Rango de valores: -328.55079125000094 a 362.27057875000145

Valores atípicos:
```

[6]:
```
4      -336.44193
11     -549.16377
44     -350.85534
55     -520.19710
60     1385.45562
          ...
552    -535.12886
558     586.64611
559   -1261.04670
561    -570.20611
562     448.44060
Name: Dif, Length: 113, dtype: float64
```



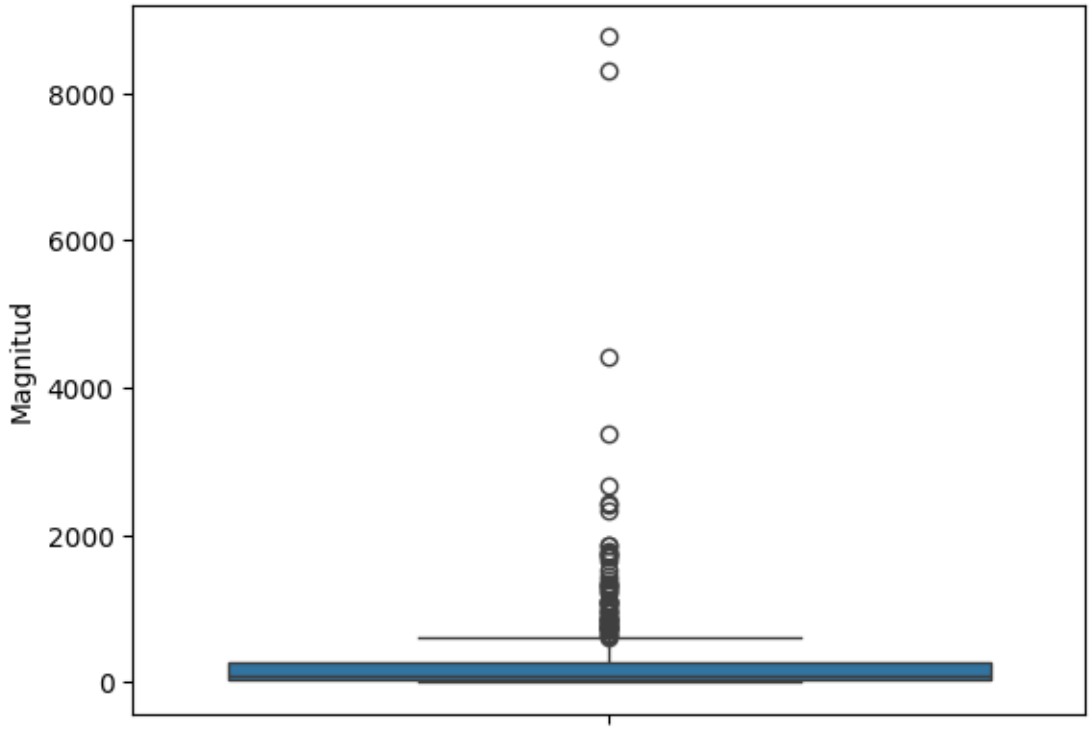
[7]:
```
IQR(df, 'Magnitud')
```

```
Rango de valores: -314.8948012499981 a 611.5858087499969

Valores atípicos:
```

[7]:
```
60     1385.45562
64      721.72014
71      847.53530
83      908.47115
86      844.06401
```

```
          ...
546    1858.82031
547    1522.02455
549    3365.40716
551    1230.91046
559    1261.04670
Name: Magnitud, Length: 61, dtype: float64
```

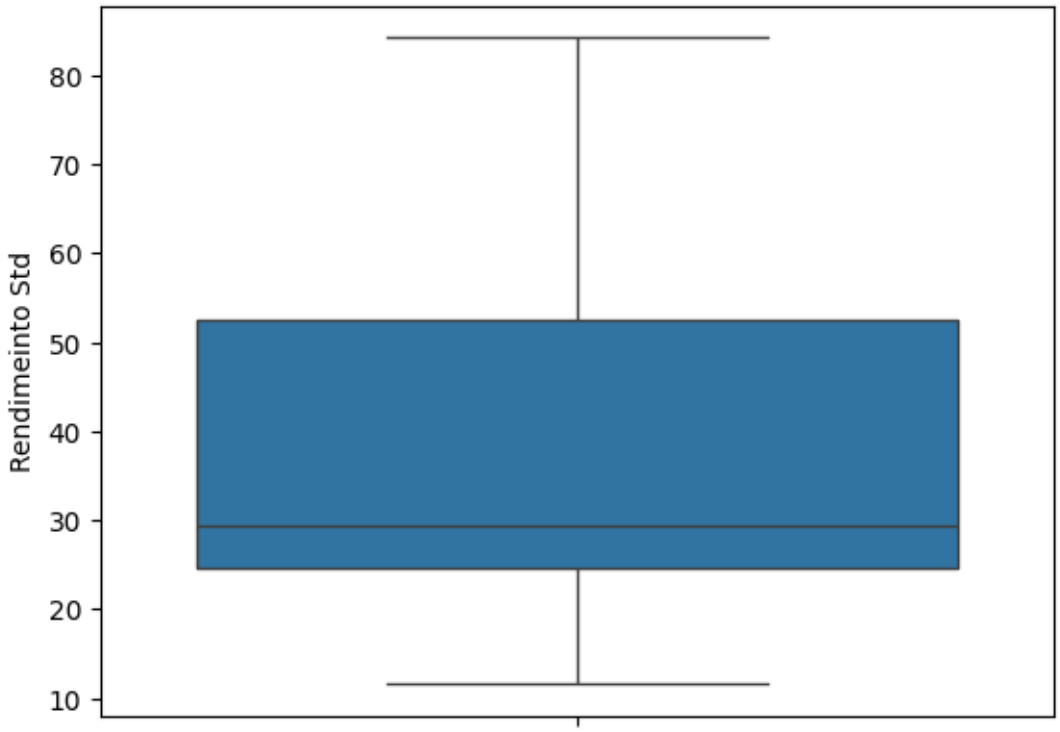


[8]:
```
IQR(df, 'Rendimeinto Std')
```

```
Rango de valores: -17.112344805213958 a 94.31235070335309

Valores atípicos:
```

[8]:
```
Series([], Name: Rendimeinto Std, dtype: float64)
```

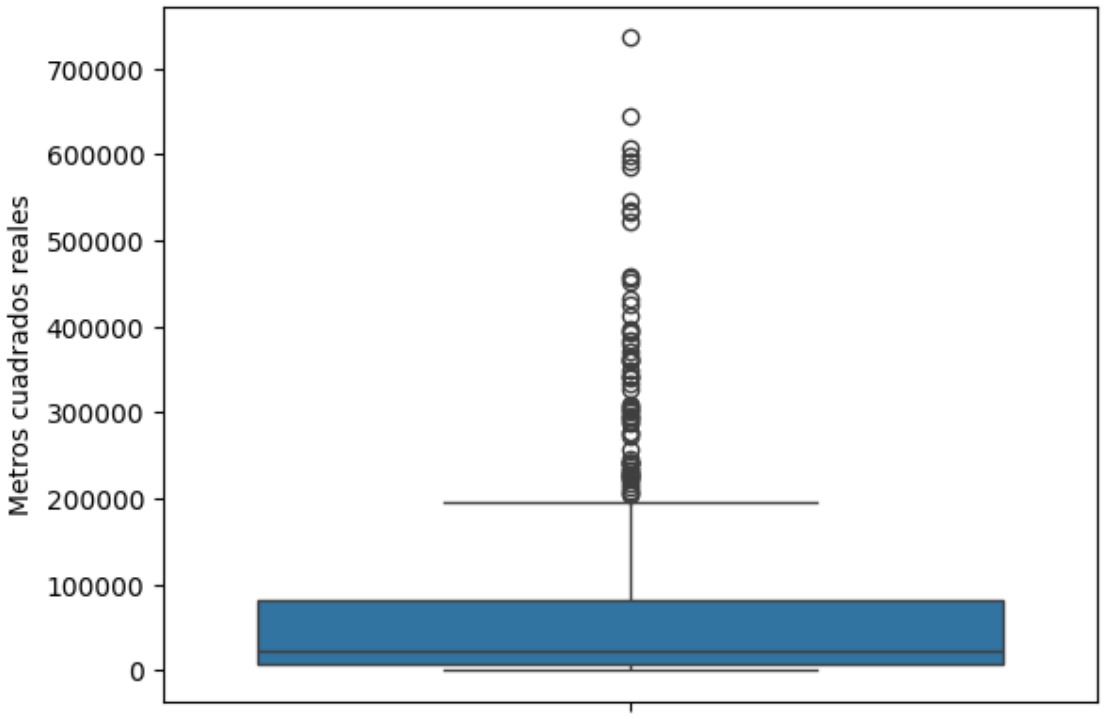


[9]:
```
IQR(df, 'Metros cuadrados reales')
```

```
Rango de valores: -106333.40400000002 a 196106.97800000003

Valores atípicos:
```

[9]:
```
43     233453.109550
44     289066.786250
53     306925.404953
55     327446.515953
60     645519.726000
          ...
546    591994.299059
549    607458.612628
551    241784.498170
559    339425.412000
561    293525.200300
Name: Metros cuadrados reales, Length: 66, dtype: float64
```



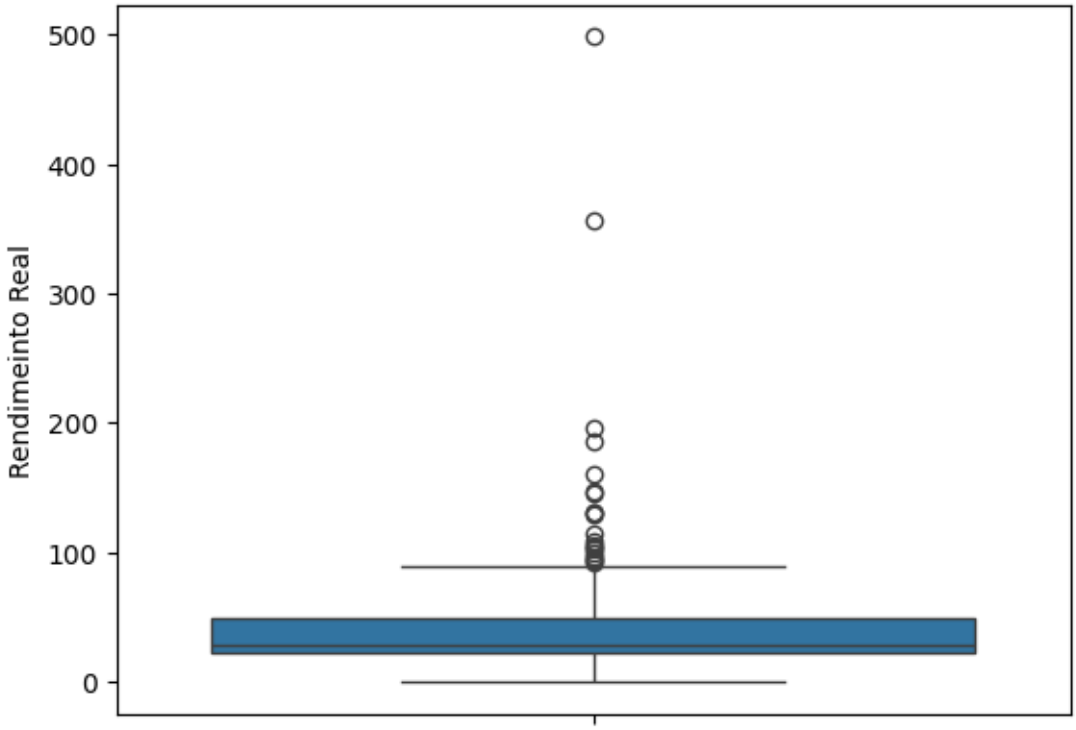
[10]:
```
df_rendimiento_real = IQR(df, 'Rendimeinto Real')
df_rendimiento_real
```

```
Rango de valores: -16.091759129316113 a 89.12842912158746

Valores atípicos:
```

[10]:
```
48     498.364792
89     130.989822
95     131.018000
104    104.290326
110    145.881013
149     93.276784
209     96.324885
223     93.093621
266    115.223591
289    357.075960
321    102.596357
322    147.248567
373    196.182900
382    100.160010
424    109.009787
430    105.291514
449    160.787680
```

```
453    130.110183
510    186.497333
551     95.119333
Name: Rendimeinto Real, dtype: float64
```

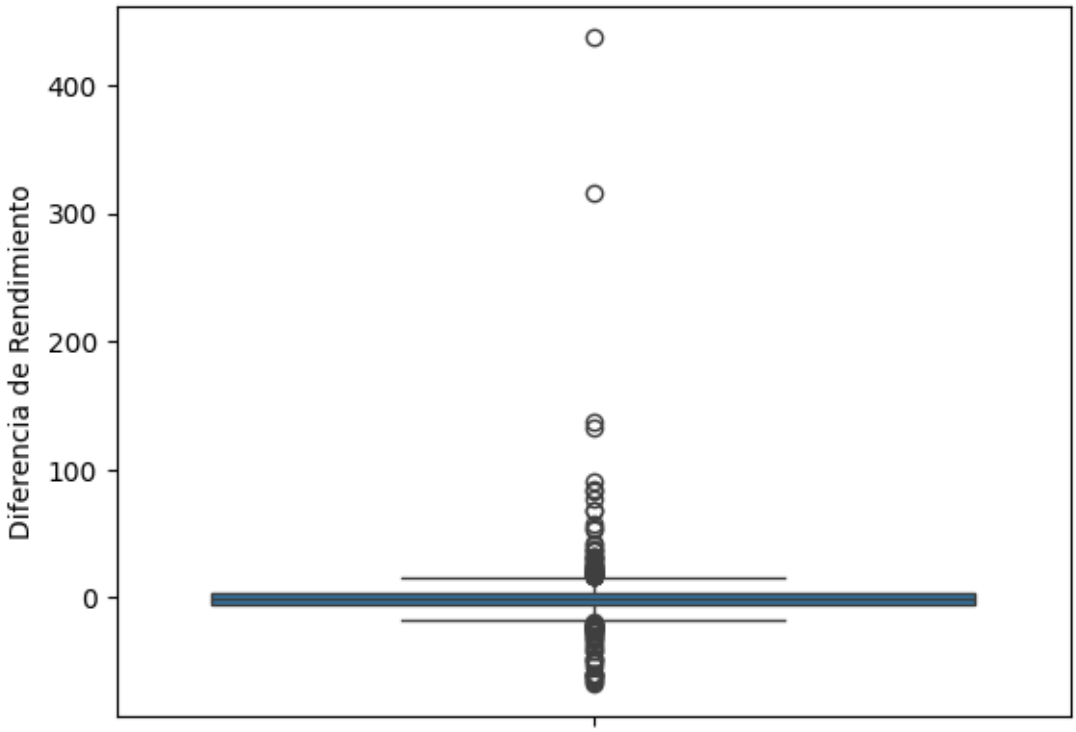


[11]:
```
IQR(df, 'Diferencia de Rendimiento')
```

```
Rango de valores: -18.253028506038156 a 16.454211636583047

Valores atípicos:
```

[11]:
```
31      19.178998
43     -18.595307
45     -24.688879
48     436.947469
51      31.998019
          ...
547    -20.537458
551     28.977601
553    -62.992126
559     16.948202
565    -39.370079
Name: Diferencia de Rendimiento, Length: 92, dtype: float64
```



1.0.1 Imputación utilizando LOCF para datos tipo object

Para este ejemplo, convertí en nulo el campo de mes en las filas donde hubiera valores atípicos dentro del rendimiento real. Después, para la imputación de la columna utilice LOCF (Last Observation Carried Forward), es decir reemplazar los nulos por el valor no nulo más reciente antes de su fila. En mi opinión, esta técnica funciona bastante bien para esta base de datos ya que para las columnas de Línea y Mes el valor de los datos no cambia constantemente en cada fila.

[12]:
```
print('Meses antes de ser convertidos en nulo: \n',df['Mes'][df['Mes'].index.
 ↪isin(df_rendimiento_real.index)])

df['Mes'] = df['Mes'].where(~df['Mes'].index.isin(df_rendimiento_real.index),␣
 ↪np.nan)
print('\nNulos:\n', df['Mes'][df['Mes'].index.isin(df_rendimiento_real.index)])

df['Mes'].ffill(inplace=True)
print('\nResultado después de aplicar LOCF:\n', df['Mes'][df['Mes'].index.
 ↪isin(df_rendimiento_real.index)])
```

```
Meses antes de ser convertidos en nulo:
 48      Abril
89      Abril
95      Abril
104     Abril
```

```
110     Abril
149      Mayo
209      Mayo
223      Mayo
266     Junio
289     Junio
321     Junio
322     Junio
373     Julio
382     Julio
424     Julio
430     Julio
449     Julio
453     Julio
510    Agosto
551    Agosto
Name: Mes, dtype: object

Nulos:
 48     NaN
89     NaN
95     NaN
104    NaN
110    NaN
149    NaN
209    NaN
223    NaN
266    NaN
289    NaN
321    NaN
322    NaN
373    NaN
382    NaN
424    NaN
430    NaN
449    NaN
453    NaN
510    NaN
551    NaN
Name: Mes, dtype: object

Resultado después de aplicar LOCF:
 48      Abril
89      Abril
95      Abril
104     Abril
110     Abril
149      Mayo
```

```
209      Mayo
223      Mayo
266     Junio
289     Junio
321     Junio
322     Junio
373     Julio
382     Julio
424     Julio
430     Julio
449     Julio
453     Julio
510    Agosto
551    Agosto
Name: Mes, dtype: object
```

```
/tmp/ipykernel_147505/2832809473.py:6: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Mes'].ffill(inplace=True)
```

1.0.2 Imputación utilizando regresión para datos numéricos

Para la imputación de datos numéricos inicialmente reemplacé los datos atípicos dentro de la columna de Rendimiento Real con nulos. Planeaba reemplazar todos los nulos con el promedio del Rendimiento Real, pero según lo investigado, esta técnica puede dificultar la estimación de la varianza y el error, por lo que opte por utilizar la regresión. En mi opinión, debido a la correlación de las variables con el Rendimiento Real, en especial Rendimiento std y Diferencia de Rendimiento, la regresión fue una buena elección.

[13]:
```
print('Datos iniciales antes de convertir en nulo:\n',df['Rendimeinto␣
  ↪Real'][df['Rendimeinto Real'].index.isin(df_rendimiento_real.index)])
df['Rendimeinto Real'] = df['Rendimeinto Real'].where(~df['Rendimeinto Real'].
  ↪index.isin(df_rendimiento_real.index), np.nan)
```

```
Datos iniciales antes de convertir en nulo:
 48     498.364792
89     130.989822
95     131.018000
104    104.290326
```

```
110    145.881013
149     93.276784
209     96.324885
223     93.093621
266    115.223591
289    357.075960
321    102.596357
322    147.248567
373    196.182900
382    100.160010
424    109.009787
430    105.291514
449    160.787680
453    130.110183
510    186.497333
551     95.119333
Name: Rendimeinto Real, dtype: float64
```

[14]:
```
X_test = df[df['Rendimeinto Real'].isnull()]
X_test = X_test[['Real','Teo',        'Dif', 'Magnitud', 'Rendimeinto Std',␣
 ↪'Metros cuadrados reales', 'Diferencia de Rendimiento']]
```

[15]:
```
new_df = df.dropna(subset=['Rendimeinto Real'])
new_df = new_df[['Real','Teo',        'Dif', 'Magnitud', 'Rendimeinto Std',␣
 ↪'Metros cuadrados reales', 'Diferencia de Rendimiento', 'Rendimeinto Real']]
```

[16]:
```
y = new_df['Rendimeinto Real']
X = new_df.drop('Rendimeinto Real', axis=1)
```

[17]:
```
lr = linear_model.LinearRegression()
lr.fit(X, y)
y_pred = lr.predict(X_test)
print('Valores calculados para los datos faltantes:\n', pd.Series(y_pred))
```

```
Valores calculados para los datos faltantes:
 0     498.364792
1     130.989822
2     131.018000
3     104.290326
4     145.881013
5      93.276784
6      96.324885
7      93.093621
8     115.223591
9     357.075960
10    102.596357
11    147.248567
12    196.182900
```

```
13    100.160010
14    109.009787
15    105.291514
16    160.787680
17    130.110183
18    186.497333
19     95.119333
dtype: float64
```

[18]:
```
df.loc[df['Rendimeinto Real'].isnull(), 'Rendimeinto Real'] = y_pred
print("Valores 'nuevos' en el dataframe: \n", df['Rendimeinto␣
  ↪Real'][df['Rendimeinto Real'].index.isin(df_rendimiento_real.index)])
print("\nNulos en la columna de Rendimiento Real: ", df['Rendimeinto Real'].
  ↪isnull().sum())
```

```
Valores 'nuevos' en el dataframe:
 48     498.364792
89     130.989822
95     131.018000
104    104.290326
110    145.881013
149     93.276784
209     96.324885
223     93.093621
266    115.223591
289    357.075960
321    102.596357
322    147.248567
373    196.182900
382    100.160010
424    109.009787
430    105.291514
449    160.787680
453    130.110183
510    186.497333
551     95.119333
Name: Rendimeinto Real, dtype: float64

Nulos en la columna de Rendimiento Real:  0
```

1.0.3 Referencias:

Rutecki, M. (2022, 6 diciembre). *Outlier detection methods!.* Kaggle. https://www.kaggle.com/code/marcinrutecki/outlier-detection-methods Subrahmanya, S. (2018, 7 junio). *Missing Data Imputation using Regression.* Kaggle. https://www.kaggle.com/code/shashankasubrahmanya/missing-data-imputation-using-regression

Hyun, K. (2013). *The prevention and handling of the missing data.* Korean Journal of Anesthesiology, 64(5), 402–406. https://doi.org/10.4097/kjae.2013.64.5.402

[19]:
```
import pandas as pd
import numpy as np

paint_df = pd.read_feather('data/analisis_consumo_pintura.feather')
paint_square_meters_df = pd.read_feather('data/pintura_metros_cuadrados_reales.
 ↪feather')
paint_performance_df = pd.read_feather('data/rendimiento_pintura.feather')
production_df = pd.read_feather('data/production.feather')

coating_df = pd.read_feather('data/pinturas_revestidos_jul20_ago23.feather')
defects_df=pd.read_feather('data/defects.feather')
```

1.1 1 *Pipeline* de datos

Se definió una *pipeline* de procesamiento de datos en un *script* de Python independiente, con el cuál se tiene un proceso replicable para llegar a una base de datos limpia. Esta *pipeline* consta de 6 diferentes pasos:

- Importación de datos
- Concatenación de resúmenes de producción
- Separación de analisis de líneas de producción y sus defectos
- Selección y renombramiento de columnas
- Eliminación de columnas adicionales
- Conversión a formato *Feather*.

A continuación se muestra un resumen de todo el proceso de la *pipeline* de datos:

1.1.1 1.1 Importación de datos

La importación de datos consiste en importar todos los datos relevantes de los archivos de excel proveídos por Ternium. Hay un total de 12 archivos diferentes.Cada uno de estos archivos contiene varias *worksheets* dentro. Para la mayoría, solamente necesitamos importar una de estas. Solamente el archivo de Análisis de consumo de pintura requirió importar mas *worksheets*:

- Pinturas y revestidos
  - Pintado 1 UNI Agosto
- Análisis de consumo de pintura
  - Base de datos por pintura
  - Metros
  - Rendimientos INDU
- 10 archivos de resúmenes de producción
  - Resúmen producción

1.1.2 1.2 Concatenación de resúmenes de producción

Para unificar toda la información de resúmenes de producción, concatenamos todos estas tablas, para así acabar con una base de datos más cohesiva y extensiva con la cual trabajar. Se ignoro el índice original de las 10 tablas, ya que realmente no tenían ningún significado.

1.1.3 1.3 Separación de analisis de líneas de producción y sus defectos

Los resúmenes de producción contienen un grupo de columnas que se repiten 5 veces. Estás columnas representan hasta 5 posibles defectos en cada línea de producción. Por lo tanto, estas columnas representan una relación uno a muchos entre una línea de producción y sus posibles defectos. Se definió una función que separa estos grupos de 5 columnas hacia una nueva tabla de defectos, con cada defecto recibiendo una referencia hacia su respectiva línea de producción. Finalmente, se eliminaron las columnas de defectos de la tabla original, ya que ahora se pueden referenciar por medio de una operación de merge con la nueva tabla de defectos.

1.1.4 1.4 Tren de pensamiento

Se seleccionaron columnas relevantes de cada base de datos para dejar en la base de datos. Estas columnas fueron renombradas para mejor documentar su propósito. A continuación se tiene una lista de las columnas renombradas para cada base de datos, así como las razones por las cuales las seleccionamos.

1.4.1 Pinturas y revestidos

- **'Denominación objeto' por 'production_line':** Posible identificador para cada línea de producción
- **'Material' por 'paint_id'**: Pintura usada, posible identificador para pinturas
- **'Texto breve de material' por 'paint_name'**: Nombre de la pintura usada, posible identificador para pinturas
- **'Valor var.' por 'monetary_value_usd'**: Valor monetario en dólares, útil para calcular costos
- **'Ctd.total reg.' por 'total_liters_used'**: Litros usados, dato clave
- **'Planta' por 'production_plant'**: Identificación de planta, para filtrar datos
- **'Proveedor' por 'supplier'**: Proveedor, para análisis de datos agrupados
- **'Registrado' por 'date'**: Fecha de los datos, dato clave
- **'Hora' por 'hour'**: Hora de los datos, dato clave
- **'Precio' por 'price_per_liter'**: Precio de cada litro, útil para calcular costos

1.4.2 Análisis de consumo de pintura

- **'Linea' por 'production_line'**: Línea de producción
- **'Mes' por 'month'**: Mes del consumo de pintura, dato clave
- **'Mes num' por 'month_number'**: Número del mes
- **'Pintura' por 'paint'**: Pintura, posible identificador para pinturas
- **'Real' por 'real_consumption'**: Consumo real, dato clave
- **'Teo' por 'theoretical_consumption'**: Consumo tórico, dato clave
- **'Dif' por 'consumption_difference'**: Diferencia de consumos, dato clave
- **'Rendimeinto Std' por 'average_yield'**: Rendimiento promedio, dato clave
- **'Rendimeinto Real' por 'real_yield'**: Rendimiento real, dato clave
- **'Diferencia de Rendimiento' por 'yield_difference'**: Diferencia de rendimientos, dato clave
- **'Metros cuadrados reales' por 'real_produced_square_meters'**: Metros cuadrados reales producidos, dato clave

1.4.3 Rendimientos de pinturas por metro cuadrado

- **'Linea' por 'production_line'**: Identificador para cada línea de producción
- **'Mes' por 'month'**: Mes del dato
- **'Num mes' por 'month_number'**: Numero del mes
- **'Pintura' por 'paint_name'**: Nombre de la pintura
- **'Metros cuadrados reales (m2)' por 'real_square_meters'**: Metros cuadrados de rendimiento de la pintura

1.4.4 Rendimientos de pinturas metros cuadrados por litro

- **'Pintura' por 'paint_name'**: Nombre de la pintura
- **'Clave' por 'paint_code'**: Identificador de la pintura
- **'Rendimiento Canning [m2/L]' por 'paint_performance_m2/l'**: Rendimiento de metros cuadrados por litro

1.4.5 Resumen de producción

- **'Linea' por 'production_line'**: Posible identificador para cada línea de producción
- **'Material Entrada' por 'input_material_code'**: Id del material de entrada, dato clave
- **'Material Salida' por 'output_material_code'**: Id del material de salida, dato clave
- **'Fecha Inicio' por 'start_date'**: Fecha de inicio de la producción, dato clave
- **'Fecha Fin' por 'end_date'**: Fecha de fin de la producción, dato clave
- **'Cliente' por 'client_name'**: Cliente, posible agrupación
- **'Código Clear Inf' por 'inferior_clear_code'**: Código del *clear* inferior
- **'Código Clear Sup' por 'superior_clear_code'**: Código del *clear* superior
- **'Ancho 1' por 'width1_mm'**: Ancho 1, dato clave
- **'Ancho 2' por 'width2_mm'**: Ancho 2, dato clave
- **'Ancho 3' por 'width3_8mm'**: Ancho 3, dato clave
- **'Ancho' por 'width_mm'**: Ancho promedio, dato clave
- **'Espesor 1' por 'thickness1_mm'**: Espesor 1, dato clave
- **'Espesor 2' por 'thickness2_mm'**: Espesor 2, dato clave
- **'Espesor 3' por 'thickness3_mm'**: Espesor 3, dato clave
- **'Espesor' por 'thickness_mm'**: Espesor promedio, dato clave
- **'Peso Entrada' por 'input_weight_kg'**: Peso de entrada, dato clave
- **'Peso' por 'weight_kg'**: Peso de salida, dato clave
- **'Largo' por 'length_m'**: largo total, dato clave
- **'Color Inferior' por 'inferior_color_code'**: Código del color inferior
- **'Color Superior' por 'superior_color_code'**: Código del color superior
- **'Primer Superior' por 'superior_primer_code'**: Código del *primer* superior
- **'Primer Inferior' por 'inferior_primer_code'**: Código del *primer* inferior
- **'Ruta Teórica' por 'route'**: Ruta teórica de la producción, nos sirve para filtrar las líneas de producción de UNI

1.4.6 Defectos de producción

- **'Codigo defecto' por 'defect_code'**: Posible identificador para tipo de defecto
- **'Defecto' por 'defect_name'**: Nombre de defecto
- **'Ubicacion' por 'location'**: Ubicación en la línea donde sucedió el defecto

- **'Es Contencion' por 'is_containment'**: Si el defecto fue contención de un suceso
- **'Es Prevencion' por 'is_preventive'**: Si el defecto fue prevención de un riesgo
- **'Intensidad' por 'intensity'**: Intensidad del suceso
- **'Cara' por 'face'**: Cara donde sucedió el defecto
- **'Lado' por 'side'**: Lado donde sucedió el defecto
- **'Frecuencia' por 'frequency'**: Frecuencia del defecto
- **'Fecha Registro' por 'register_date'**: Fecha del suceso

1.1.5 1.5 Eliminación de columnas adicionales

Una vez seleccionadas y renombradas las columnas relevantes, se excluyeron de las tablas todas las columnas adicionales.

1.1.6 1.6 Conversión a formato *feather*

Feather es un format de archivos que permite almacenar tablas o *dataframes* de una manera eficiente y agnostica al lenguaje. Para el reto, es mucho mas eficiente trabajar con los datos en formato *Feather* que directamente con los archivos de Excel, por lo cual se decidió utilizar este formato para almacenar los artefactos generados por la *pipeline* de datos.

El proceso de escribir una tabla a formato *Feather* automáticamente infiere los tipos de todas las columnas, convirtiendo las columnas númericas a sus respectivos tipos, p. ej. '1.02' a tipo *float* y '232' a tipo *int*. Cualquier otra columna será convertida a tipo *object*, con las cuales se debe llevar a cabo una conversión manual. Esta conversión es realizada en secciones posteriores de este documento.

1.2 2 Filtrado de datos

Para cuestiones del reto, solamente es de interés trabajar con datos que sean de la planta Ternium Universidad. Por lo tanto, se necesita filtrar las tablas a que únicamente incluyan valores provenientes de ahi.

1.2.1 2.1 Pinturas y revestidos

Para la tabla de pinturas y revestidos, la columna `production_plant` indica a cual planta pertenece cada línea de producción. Por lo tanto, se filtraron valores diferentes a 'Uni'.

[20]:
```
coating_df = coating_df[coating_df['production_plant'] == 'Uni']
coating_df
```

[20]:

	production_line	paint_id	paint_name	\
2	Pintado 1 Prod. Univ	U0222_AKZO	0222-AZUL RAL-5003 FC	
35	Pintado 2 Prod. Univ	I3329_AKZO	0318-TOPAZ METALLIC 14H2593	
36	Pintado 2 Prod. Univ	I3329_AKZO	0318-TOPAZ METALLIC 14H2593	
44	Pintado 1 Prod. Univ	I3052_VALS	0121-BLANCO STD KYNAR	
176	Pintado 1 Prod. Univ	I3052_VALS	0121-BLANCO STD KYNAR	
…	…	…	…	
211698	Pintado 1 UNI	I0402_VALS	0400-GRIS FONDO	
211699	Pintado 1 UNI	I4078_VALS	WHITE KRYSTAL KOTE	
211701	Pintado 1 UNI	I0100_VALS	0102-BLANCO STD	

```
211704         Pintado 1 UNI  I1270_BECK          0079-BECKRYPRIM 246
211705         Pintado 1 UNI  I1150_AKZO              0038-PRIMER 917

        monetary_value_usd  total_liters_used production_plant supplier  \
2                   5976.52            189.250              Uni     AKZO
35                  3314.07            189.267              Uni     AKZO
36                  3314.07            189.267              Uni     AKZO
44                  3126.00            200.000              Uni     VALS
176                 2625.84            168.000              Uni     VALS
…                     …                  …                …       …
211698              -423.00            -75.000              Uni     VALS
211699              -896.00            -70.000              Uni     VALS
211701             -1170.00           -200.000              Uni     VALS
211704             -1470.00           -200.000              Uni     BECK
211705             -1481.83           -189.250              Uni     AKZO

               date      hour  price_per_liter
2       24.07.2020  00:12:52        31.580026
35      24.07.2020  22:28:26        17.510026
36      24.07.2020  21:28:35        17.510026
44      25.07.2020  14:00:07        15.630000
176     25.07.2020  14:32:47        15.630000
…              …       …               …
211698  18.08.2023  00:00:00         5.640000
211699  18.08.2023  00:00:00        12.800000
211701  18.08.2023  00:00:00         5.850000
211704  18.08.2023  00:00:00         7.350000
211705  18.08.2023  00:00:00         7.830013

[56658 rows x 10 columns]
```

1.2.2 2.2 Resumen de producción

Para la tabla de resumen de producción, la columna `route` contiene la ruta teoríca que cada pintura sigue. Para verificar si estos son datos relevantes al caso, fue decidido que se filtraria esta columna con base en las partes de cada ruta. Cada ruta esta estructurada de manera que dice 'planta-planta-planta-planta', separados por guiones. Por lo tanto, se separaron todos los valores y se busco que tuvieran el valor 'UNI' dentro.

[21]:
```
routes = production_df['route'].apply(lambda route: all(map(lambda value: 'UNI' in value, route.split('-'))))
production_df = production_df[routes]
production_df
```

[21]:
```
      input_material_code output_material_code          start_date  \
0           4A058886UP100        4A058886UP101 2023-03-31 22:57:32
1           2B126983UG400        2B126983UP100 2023-03-31 23:42:50
```

```
2           2B126983UG400        2B126983UP101 2023-04-01 00:03:27
3           2B126983UG400        2B126983UP102 2023-04-01 00:23:34
4           2B126984UG400        2B126984UP100 2023-04-01 00:43:19
…                   …                    …                   …
17644       3B530354UG400        3B530354UP201 2023-08-31 21:19:28
17645       3B530354UG400        3B530354UP202 2023-08-31 21:35:48
17646       3B530354UG400        3B530354UP203 2023-08-31 21:51:53
17647       4A250820UN100        4A250820UP204 2023-08-31 22:07:26
17648       4A250820UN100        4A250820UP205 2023-08-31 22:20:40

                 end_date                                      client_name  \
0     2023-03-31 23:42:50                    CONSTRUCCIONES REFRIGERADAS SA
1     2023-04-01 00:03:27  CONSORCIO MINERO BENITO JUAREZ PEÑA COLORADA S…
2     2023-04-01 00:23:34  CONSORCIO MINERO BENITO JUAREZ PEÑA COLORADA S…
3     2023-04-01 00:43:19  CONSORCIO MINERO BENITO JUAREZ PEÑA COLORADA S…
4     2023-04-01 00:53:22  CONSORCIO MINERO BENITO JUAREZ PEÑA COLORADA S…
…                   …                                                  …
17644 2023-08-31 21:35:48                          INDUSTRIAS ACROS WHIRLPOOL
17645 2023-08-31 21:51:53                          INDUSTRIAS ACROS WHIRLPOOL
17646 2023-08-31 22:07:26                          INDUSTRIAS ACROS WHIRLPOOL
17647 2023-08-31 22:20:40                          INDUSTRIAS ACROS WHIRLPOOL
17648 2023-08-31 22:36:11                          INDUSTRIAS ACROS WHIRLPOOL

       inferior_clear_code superior_clear_code  width1_mm  width2_mm  \
0                     None                None     1223.0     1223.0
1                     None                None     1221.0     1221.0
2                     None                None     1221.0     1221.0
3                     None                None     1221.0     1221.0
4                     None                None     1222.0     1222.0
…                      …                   …        …          …
17644  0028-CLEAR P/ESPUMA                None      919.0      919.0
17645  0028-CLEAR P/ESPUMA                None      919.0      919.0
17646  0028-CLEAR P/ESPUMA                None      919.0      919.0
17647  0028-CLEAR P/ESPUMA                None      918.0      918.0
17648  0028-CLEAR P/ESPUMA                None      918.0      918.0

       width3_mm  …  thickness3_mm  thickness_mm  input_weight_kg  \
0         1223.0  …          0.464        0.4640             5290
1         1221.0  …          0.739        0.7390            20660
2         1221.0  …          0.739        0.7390            20660
3         1221.0  …          0.739        0.7390            20660
4         1222.0  …          0.759        0.7590            11525
…          …  …             …            …                 …
17644      919.0  …          0.383        0.3833            15753
17645      919.0  …          0.383        0.3833            15753
17646      919.0  …          0.383        0.3833            15753
17647      918.0  …          0.372        0.3723             6750
```

```
17648      918.0  …          0.372        0.3723             6750

       weight_kg  length_m    inferior_color_code        superior_color_code  \
0           5205      1175                   None                       None
1           6860       954  0121-BLANCO STD KYNAR  0226-BREATHTAKING BLUE FC
2           6845       952  0121-BLANCO STD KYNAR  0226-BREATHTAKING BLUE FC
3           6500       904  0121-BLANCO STD KYNAR  0226-BREATHTAKING BLUE FC
4           3165       427  0121-BLANCO STD KYNAR  0226-BREATHTAKING BLUE FC
…          …       …                      …                     …
17644       4183      1494                   None     0104-BLANCO SUPERMATIC
17645       4113      1468                   None     0104-BLANCO SUPERMATIC
17646       3963      1415                   None     0104-BLANCO SUPERMATIC
17647       3343      1248                   None     0104-BLANCO SUPERMATIC
17648       3688      1377                   None     0104-BLANCO SUPERMATIC

        superior_primer_code inferior_primer_code  \
0                       None                 None
1            0038-PRIMER 917      0038-PRIMER 917
2            0038-PRIMER 917      0038-PRIMER 917
3            0038-PRIMER 917      0038-PRIMER 917
4                       None                 None
…                      …                   …
17644  0021-UNIVERSAL PRIMER                 None
17645  0021-UNIVERSAL PRIMER                 None
17646  0021-UNIVERSAL PRIMER                 None
17647  0021-UNIVERSAL PRIMER                 None
17648  0021-UNIVERSAL PRIMER                 None

                               route
0      G4_UNI-P1_UNI-LAB_UNI-EMBP1_UNI
1      G4_UNI-P1_UNI-LAB_UNI-EMBP1_UNI
2      G4_UNI-P1_UNI-LAB_UNI-EMBP1_UNI
3      G4_UNI-P1_UNI-LAB_UNI-EMBP1_UNI
4      G4_UNI-P1_UNI-LAB_UNI-EMBP1_UNI
…                                  …
17644  G4_UNI-P2_UNI-LAB_UNI-EMBP2_UNI
17645  G4_UNI-P2_UNI-LAB_UNI-EMBP2_UNI
17646  G4_UNI-P2_UNI-LAB_UNI-EMBP2_UNI
17647  G4_UNI-P2_UNI-LAB_UNI-EMBP2_UNI
17648  G4_UNI-P2_UNI-LAB_UNI-EMBP2_UNI

[16026 rows x 23 columns]
```

1.3 3 Verificación de inconsistencias en los datos

1.3.1 3.1 Análisis de consumo de pintura

Primero, para todas las columnas de tipo objeto de la base de datos de análisis de consumo de pintura se convirtieron a sus respectivos tipos. En la tabla principal, hay dos: - production_line: solo tiene dos valores, por lo que es **categoríca** - month: es el mes textual, por lo que es **categoríca** - paint: es el identificador de la pintura, por lo que es un **string**

En la tabla de rendimiento por metro cuadrado, hay 3: - production_line: solo tiene dos valores, por lo que es **categoríca** - month: es el mes textual, por lo que es **categoríca** - paint_name: es el identificador de la pintura, por lo que es un **string**

En la tabla de rendimiento de metro cuadrado por litro, hay 2: - paint_name: el nombre de la pintura, por lo que es un **string** - paint_code: el identificador de la pintura, por lo que es un **string**

[22]:
```
paint_df['production_line'] = paint_df['production_line'].astype('category')
paint_df['month'] = paint_df['month'].astype('category')
paint_df['paint'] = paint_df['paint'].astype('string')
paint_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 566 entries, 0 to 565
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   production_line               566 non-null    category
 1   month                         566 non-null    category
 2   month_number                  566 non-null    int64
 3   paint                         566 non-null    string
 4   real_consumption              566 non-null    float64
 5   theoretical_consumption       566 non-null    float64
 6   consumption_difference        566 non-null    float64
 7   average_yield                 566 non-null    float64
 8   real_yield                    566 non-null    float64
 9   yield_difference              566 non-null    float64
 10  real_produced_square_meters   566 non-null    float64
dtypes: category(2), float64(7), int64(1), string(1)
memory usage: 41.4 KB
```

[23]:
```
paint_square_meters_df['production_line'] =␣
 ↪paint_square_meters_df['production_line'].astype('category')
paint_square_meters_df['month'] = paint_square_meters_df['month'].
 ↪astype('category')
paint_square_meters_df['paint_name'] = paint_square_meters_df['paint_name'].
 ↪astype('string')
paint_square_meters_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 620 entries, 0 to 619
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   production_line     620 non-null    category
 1   month               620 non-null    category
 2   month_number        620 non-null    int64
 3   paint_name          620 non-null    string
 4   real_square_meters  620 non-null    float64
dtypes: category(2), float64(1), int64(1), string(1)
memory usage: 16.2 KB
```

[24]:
```
paint_performance_df['paint_name'] = paint_performance_df['paint_name'].
 ↪astype('string')
paint_performance_df['paint_code'] = paint_performance_df['paint_code'].
 ↪astype('string')
paint_performance_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1997 entries, 0 to 1996
Data columns (total 3 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   paint_name              1997 non-null   string
 1   paint_code              1997 non-null   string
 2   paint_performance_m2/l  1958 non-null   float64
dtypes: float64(1), string(2)
memory usage: 46.9 KB
```

3.1.1 Diferencia entre consumo real y teórico Verificamos que la columna de diferencia de consumo se calculara correctamente mediante una serie que contiene la resta del consumo real menos el consumo teórico y comparamos esta diferencia con la de la columna del archivo para verificar si todos los registro son correctos.

[25]:
```
calculated_consumption_difference = paint_df['real_consumption'] -␣
 ↪paint_df['theoretical_consumption']
if paint_df['consumption_difference'].equals(calculated_consumption_difference):
    print('La columna "consumption_difference" se calculó correctamente.')
else:
    print('La columna "consumption_difference" no se calculó correctamente.')
```

```
La columna "consumption_difference" se calculó correctamente.
```

3.1.2 Metros cuadrados reales Verificamos que los metros cuadrados reales se calcularan correctamente en el dataframe de análisis de consumo de pintura por medio de un dataframe generado por mmedio de un merge con el dataframe de los metros cuadrados de cada pintura para localizar aquellos puntos en los que tanto la línea de producción, el mes y el nombre de la pintura coincidan para comparar los metros cuadrados de cada dataframe

[26]:
```
paint_merged_df = pd.merge(paint_df, paint_square_meters_df, how='left',␣
↪left_on=['production_line', 'month', 'paint'], right_on=['production_line',␣
↪'month', 'paint_name'])
if paint_merged_df['real_produced_square_meters'].
↪equals(paint_merged_df['real_square_meters']):
    print('Los metros cuadrados se calcularon correctamente.')
else:
    print('Los metros cuadrados no se calcularon correctamente.')
```

```
Los metros cuadrados se calcularon correctamente.
```

3.1.3 Rendimiento Estandar VS Rendimiento Real Se calculó el nuevo rendimiento std utilizando el promedio del rendimientos por pintura, en vez de utilizar el primer valor encontrado en la hoja de rendimientos. Después, se agrego una columna new_average_yield (nuevo rendimiento std) dentro pintura_df.

[27]:
```
new_performance = paint_performance_df[['paint_name','paint_performance_m2/l']].
↪groupby('paint_name', as_index=False).mean()
paint_df = paint_df.merge(new_performance, left_on='paint',␣
↪right_on='paint_name')
paint_df.rename(columns = { 'paint_performance_m2/l': 'new_average_yield'},␣
↪inplace=True)
paint_df.drop(['paint_name'], axis=1, inplace=True)
paint_df
```

[27]:

	production_line	month	month_number	paint	\
0	Pintado 1	Abril	4	0226-BREATHTAKING BLUE FC	
1	Pintado 1	Abril	4	0121-BLANCO STD KYNAR	
2	Pintado 1	Abril	4	0919-ROJO TERNIUM CR	
3	Pintado 1	Abril	4	1487-AMERICAN STERLING III	
4	Pintado 1	Abril	4	0435-APOLLO GRAY KRYSTAL KOTE	
..	…	…	…	…	
561	Pintado 2	Agosto	8	0099-CLEAR P/ESPUMA ROHS	
562	Pintado 2	Agosto	8	0075-CLEAR LAUNDRY BACKER ROHS	
563	Pintado 2	Agosto	8	0023-CLEAR EPOXICO P/ESPUMA	
564	Pintado 2	Agosto	8	1028-CLEAR BLUE EPOXY	
565	Pintado 2	Agosto	8	0034-CLEAR AZUL P/ESPUMA	

	real_consumption	theoretical_consumption	consumption_difference	\
0	203.25	185.69356	17.55644	
1	1412.75	1690.59626	-277.84626	
2	487.36	472.13485	15.22515	
3	760.00	729.61405	30.38595	
4	6906.00	7242.44193	-336.44193	
..	…	…	…	
561	5866.75	6436.95611	-570.20611	
562	1444.00	995.55940	448.44060	

```
563          1073.25              1184.15414             -110.90414
564           200.00               191.27243                8.72757
565             0.00                18.37746              -18.37746

     average_yield  real_yield  yield_difference  real_produced_square_meters  \
0        24.146982   22.061200         -2.085781                  4483.939000
1        20.997375   27.639634          6.642259                 39047.893000
2        22.572178    6.384172        -16.188007                  3111.390000
3        26.404494   24.706582         -1.697913                 18777.002000
4        26.666667   27.114563          0.447896                187253.169513
..             …           …                 …                           …
561      40.944882   50.031994          9.087112                293525.200300
562      77.165354   60.218956        -16.946398                 86956.173000
563      39.370079   52.568644         13.198565                 56419.297265
564      42.519685   46.948690          4.429005                  9389.738000
565      39.370079    0.000000        -39.370079                   818.960000

     new_average_yield
0             24.146982
1             22.047244
2             22.572178
3             24.719101
4             26.666667
..                  …
561           39.370079
562           77.165354
563           44.291339
564           40.944882
565           49.448819

[566 rows x 12 columns]
```

Se calculo el rendimiento real usando la fórmula que ternium nos proporcionó (metros cuadrados reales / rendimiento real), en donde se almacena en una variable. Después, generamos la diferencia entre lo que nosotros calculamos con los datos dentro de la base de datos. Finalmente, el código filtra los datos y verifica si hay diferencias mayores a 0 y de ser así las imprime, si no imprime que no hay.

[28]:
```python
#Rendimiento real
calculated_real_yield =  paint_df['real_produced_square_meters'] / paint_df[␣
 ↪'real_consumption']
real_yield_difference =  paint_df['real_yield'] - calculated_real_yield

#Checa si hay valores mayor a 0
ry_differences = real_yield_difference[real_yield_difference > 0]
if not ry_differences.empty:
    print(differences)
```

```
else:
    print("Real Yield : No differences found.")
```

```
Real Yield : No differences found.
```

Aquí, se generó una nueva variable en donde almacena la diferencia entre rendimiento real (proporcionado en la bs) y el rendimiento std (utilizando el promedio). Igual que el código anterior si encontró dato mayor a 0 los imprime, de no ser así imprime un mensaje comentando que no hay diferencias.

En este resultado podemos ver como si encontró diferencias entre real y el nuuevo rendimiento std, donde varían de decimales hasta enteros.

[29]:
```
#Rendimiento std
##rendimiento real menos rendimineto estandar actualizado
average_yield_difference = paint_df['real_yield'] -␣
 ↪paint_df['new_average_yield']

#Checa si hay valores mayor a 0
ay_differences = average_yield_difference[average_yield_difference > 0]
if not ay_differences.empty:
    print(ay_differences)
else:
    print("Average Yield : No differences found.")
```

```
1       5.592390
4       0.447896
5       1.824107
6       0.890923
7       0.007035
         …
559    13.452139
560     6.048982
561    10.661915
563     8.277306
564     6.003808
Length: 245, dtype: float64
```

1.3.2 3.2 Resumen de producción

Primero se convirtieron los datos de tipo object del resumen de producción. Se encontrarón 10 columnas: - input_material_code: codigo del material de entrada, **string** - output_material_code: material de salida, **string** - client_name: nombre de cliente, **string** - inferior_clear_code: nombre de *clear* inferior, **string** - superior_clear_code:nombre de *clear* superior, **string** - inferior_color_code: nombre de color inferior, **string** - superior_color_code: nombre de color superior, **string** - superior_primer_code: nombre de *primer* superior, **string** - inferior_primer_code: nombre de *primer* inferior, **string** - route: ruta a tomar, **string**

[30]:
```
production_df['input_material_code'] = production_df['input_material_code'].
 ↪astype('string')
production_df['output_material_code'] = production_df['output_material_code'].
 ↪astype('string')
production_df['client_name'] = production_df['client_name'].astype('string')
production_df['inferior_clear_code'] = production_df['inferior_clear_code'].
 ↪astype('string')
production_df['superior_clear_code'] = production_df['superior_clear_code'].
 ↪astype('string')
production_df['inferior_color_code'] = production_df['inferior_color_code'].
 ↪astype('string')
production_df['superior_color_code'] = production_df['superior_color_code'].
 ↪astype('string')
production_df['superior_primer_code'] = production_df['superior_primer_code'].
 ↪astype('string')
production_df['inferior_primer_code'] = production_df['inferior_primer_code'].
 ↪astype('string')
production_df['route'] = production_df['route'].astype('string')
production_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16026 entries, 0 to 17648
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   input_material_code   16026 non-null  string
 1   output_material_code  16026 non-null  string
 2   start_date            16026 non-null  datetime64[ns]
 3   end_date              16026 non-null  datetime64[ns]
 4   client_name           16026 non-null  string
 5   inferior_clear_code   5829 non-null   string
 6   superior_clear_code   1859 non-null   string
 7   width1_mm             16026 non-null  float64
 8   width2_mm             16026 non-null  float64
 9   width3_mm             16026 non-null  float64
 10  width_mm              16026 non-null  float64
 11  thickness1_mm         16026 non-null  float64
 12  thickness2_mm         16026 non-null  float64
 13  thickness3_mm         16026 non-null  float64
 14  thickness_mm          16026 non-null  float64
 15  input_weight_kg       16026 non-null  int64
 16  weight_kg             16026 non-null  int64
 17  length_m              16026 non-null  int64
 18  inferior_color_code   5027 non-null   string
 19  superior_color_code   10688 non-null  string
 20  superior_primer_code  5729 non-null   string
 21  inferior_primer_code  2967 non-null   string
```

```
 22  route                 16026 non-null  string
dtypes: datetime64[ns](2), float64(8), int64(3), string(10)
memory usage: 2.9+ MB
```

3.2.1 Promedio de anchos En el resultado podemos observar como si llego a detectar diferencias dentro del promedio del espesor, la cantidad de diferencias salió alta debido al margen de diferencia de los decimales. Al checar con un valor mayor a 0 sale un margen alto de datos.

[31]:
```
# Calculate average width
average_width_mm = production_df[['width1_mm', 'width2_mm', 'width3_mm']].
 ↪mean(axis=1)
#print(average_width_mm)
# Calculate real width difference
real_width_mm = production_df['width_mm'] - average_width_mm

# Check for differences
aw_differences = real_width_mm[real_width_mm >  0 ]
if not aw_differences.empty:
    print(aw_differences)
else:
    print("No Differences")
```

```
52       0.003333
98       0.000333
246      0.666667
253      0.666667
632      0.003333
           …
16096    0.000333
16097    0.000333
16546    0.003333
16547    0.003333
16548    0.003333
Length: 166, dtype: float64
```

Si nos vamos a diferencia mayor a 1, disminuye el margen de diferencias.

[32]:
```
# Check for differences
aw_differences = real_width_mm[real_width_mm >  1 ]
if not aw_differences.empty:
     print('Differences found: \n' , aw_differences)
else:
    print("No Differences")
```

```
Differences found:
 1841    3.333333
dtype: float64
```

3.2.2 Promedio de espesores En el resultado podemos observar como si llego a detectar diferencias dentro del promedio del espesor, la cantidad de diferencias salió alta debido al margen de diferencia de los decimales. Al checar con un valor mayor a 0 sale un margen alto de datos.

[33]:
```
average_thickness_mm =production_df[['thickness1_mm', 'thickness2_mm',␣
 ↪'thickness3_mm']].mean(axis=1)

difference_thickness_mm =  production_df['thickness1_mm'] - average_thickness_mm

# Check for differences
t_differences = difference_thickness_mm[difference_thickness_mm >  0]
if not t_differences.empty:
     print('Differences found: \n' , t_differences)
else:
    print("No Differences")
```

```
Differences found:
 101      0.001333
102      0.001333
103      0.000667
104      0.001000
105      0.002667
           …
17638    0.000333
17639    0.000333
17640    0.000333
17641    0.000333
17642    0.000333
Length: 3131, dtype: float64
```

Si nos vamos a diferencia mayor a 1, disminuye el margen de diferencias.

[34]:
```
# Check for differences
t_differences = difference_thickness_mm[difference_thickness_mm >  1]
if not t_differences.empty:
    print('Differences found: \n' , t_differences)
else:
    print("No Differences")
```

```
No Differences
```

3.2.3 Colores repetidos Verificamos si es que tenemos renglones/registros en dónde aparezcan los mismos colores. Esto es en las columnas “Inferior Clear Code”,” Superior Clear Code”, “Clear Inferior”, “Clear superior”, “Color Inferior”, “Color Superior”, “Primer Superior” y “Primer Inferior”.

Esa verificación fue llevada a cabo al filtrar el dataframe para solo quedarnos con renglones/registros en donde existan pinturas repetidas, se ignoraron los valores None en cada renglon/Registro.

Al tener el dataframe filtrado procedemos a obtener información relevante de los procesos de pintado

como sacar las tipos de pinturas más utilizadas en cada proceso del pintado; estás siendo clear, superior y color

[35]:
```
#Columnas para evaluar pinturas repetidas
relevant_columns = ['inferior_clear_code', 'superior_clear_code',␣
 ↪'inferior_color_code', 'superior_color_code', 'superior_primer_code',␣
 ↪'inferior_primer_code']

#Creamos copia de producción
filtered_df = production_df.copy()

# Función para verificar si hay valores repetidos en un registro específico     ␣
 ↪                                                                              ␣
 ↪
def check_duplicate_colors(row):
    values = [] # se crea una lista de valores en las columnas
    for col_name in relevant_columns: #revisamos las columnas relevantes       ␣
 ↪                                                                              ␣
 ↪                                                                              ␣
 ↪
        if(row[col_name] != None): # si la columna tiene un identificador de␣
 ↪pintura
            values.append(row[col_name]) #lo agregamos a la lista

    # si todos los valores son unicos y la lista no esta vacía, significa que␣
 ↪se llevaron a cabo procesos de pintado y cada uno utilizo pinturas distintas
    return len(values) != len(set(values)) and len(values) > 0

# Utilizamos la función con el dataframe filtrado
duplicate_indices = filtered_df[filtered_df.apply(check_duplicate_colors,␣
 ↪axis=1)].index.tolist()

# Imprimir indices de renglones con valroes repetidos
#print("Indices of rows with repeated values in 'inferior_color_code',␣
 ↪'superior_color_code', 'superior_primer_code', and 'inferior_primer_code':",␣
 ↪duplicate_indices)

# Renglones con valores repetidos en las columnas de interes
rows_with_repeated_values = filtered_df.loc[duplicate_indices,␣
 ↪['inferior_clear_code', 'superior_clear_code', 'inferior_color_code',␣
 ↪'superior_color_code', 'superior_primer_code', 'inferior_primer_code']]

rows_with_repeated_values
```

[35]:
```
       inferior_clear_code superior_clear_code    inferior_color_code  \
0                     <NA>                <NA>                   <NA>
1                     <NA>                <NA>  0121-BLANCO STD KYNAR
```

```
2                    <NA>                <NA>  0121-BLANCO STD KYNAR
3                    <NA>                <NA>  0121-BLANCO STD KYNAR
4                    <NA>                <NA>  0121-BLANCO STD KYNAR
…                     …                   …                    …
17644  0028-CLEAR P/ESPUMA                <NA>                 <NA>
17645  0028-CLEAR P/ESPUMA                <NA>                 <NA>
17646  0028-CLEAR P/ESPUMA                <NA>                 <NA>
17647  0028-CLEAR P/ESPUMA                <NA>                 <NA>
17648  0028-CLEAR P/ESPUMA                <NA>                 <NA>

             superior_color_code   superior_primer_code inferior_primer_code
0                            <NA>                   <NA>                 <NA>
1      0226-BREATHTAKING BLUE FC        0038-PRIMER 917      0038-PRIMER 917
2      0226-BREATHTAKING BLUE FC        0038-PRIMER 917      0038-PRIMER 917
3      0226-BREATHTAKING BLUE FC        0038-PRIMER 917      0038-PRIMER 917
4      0226-BREATHTAKING BLUE FC                   <NA>                 <NA>
…                           …                     …                    …
17644     0104-BLANCO SUPERMATIC  0021-UNIVERSAL PRIMER                 <NA>
17645     0104-BLANCO SUPERMATIC  0021-UNIVERSAL PRIMER                 <NA>
17646     0104-BLANCO SUPERMATIC  0021-UNIVERSAL PRIMER                 <NA>
17647     0104-BLANCO SUPERMATIC  0021-UNIVERSAL PRIMER                 <NA>
17648     0104-BLANCO SUPERMATIC  0021-UNIVERSAL PRIMER                 <NA>

[16026 rows x 6 columns]
```

Información de pinturas existentes en los procesos

[36]:
```
# Inicializamos diccionarios para almacenar las pinturas únicas por proceso
unique_paints_by_process = {
    'clear': set(),
    'color': set(),
    'primer': set()
}

# Iteramos sobre cada columna para identificar las pinturas únicas y su proceso␣
 ↪correspondiente
for col in rows_with_repeated_values.columns:
    if 'clear' in col:
        unique_paints_by_process['clear'].update(rows_with_repeated_values[col].
 ↪dropna().unique())
    elif 'color' in col:
        unique_paints_by_process['color'].update(rows_with_repeated_values[col].
 ↪dropna().unique())
    elif 'primer' in col:
        unique_paints_by_process['primer'].
 ↪update(rows_with_repeated_values[col].dropna().unique())
```

```python
# Convertir sets a listas para una manipulación más fácil si es necesario
for process in unique_paints_by_process:
    unique_paints_by_process[process] = list(unique_paints_by_process[process])

# Mostrar las pinturas únicas por proceso
for process, paints in unique_paints_by_process.items():
    print(f"Proceso de {process}:")
    for paint in paints:
        print(f" - {paint}")

# Para obtener una lista general de todas las pinturas únicas, sin importar el proceso
all_unique_paints = set().union(*unique_paints_by_process.values())
print("\nLista general de todas las pinturas únicas:")
for paint in all_unique_paints:
    print(f" - {paint}")

# Determinar en qué proceso(s) se utiliza cada pintura única
paints_process_usage = {paint: [] for paint in all_unique_paints}
for paint in all_unique_paints:
    for process, paints in unique_paints_by_process.items():
        if paint in paints:
            paints_process_usage[paint].append(process)

# Mostrar en qué proceso(s) se utiliza cada pintura
print("\nUso de pinturas en procesos:")
for paint, processes in paints_process_usage.items():
    print(f" - Pintura {paint} se utiliza en el(los) proceso(s) de {' y '.join(processes)}.")
```

```
Proceso de clear:
 - 1078-CLEAR TOP 20 GLOSS
 - 0093-50 GLOSS CLEAR ONE PASS
 - 0097-CLEAR APPLIANCE BACKER MC
 - 0024-CLEAR LAUNDRY BACKER
 - 0023-CLEAR EPOXICO P/ESPUMA
 - 0075-CLEAR LAUNDRY BACKER ROHS
 - 0066-SUPER HIGH GLOSS CLEAR II
 - 0028-CLEAR P/ESPUMA
 - 0051-80 GLOSS CLEAR BACKER.
 - 0068-APPLIANCE 10 GLOSS CLEAR
 - 0034-CLEAR AZUL P/ESPUMA
 - 1028-CLEAR BLUE EPOXY
 - 0099-CLEAR P/ESPUMA ROHS
 - 0029-CLEAR INOXIDABLE
 - 1009-APPLIANCE 30 GLOSS CLR OP
 - 1022-EPOXY CHROMATE BACKER
 - 0054-CLEAR KRYSTAL KOTE
```

- 1077-CLEAR PRINT INOX MATE
- 1055-CLEAR TOP LAVADORAS
- 0092-80 GLOSS CLEAR LAV
- 0027-80 GLOSS CLEAR BACKER II
- 0744-AMARILLO TERNIUM CR FC
- 0063-CLEAR KYNAR UV
- 1152-BLANCO TERNIUM CR TOP
- 0025-CLEAR EPOXY (WASH COAT)
- 0036-BACK PRIMER
- 1070-CLEAR P/ESPUMA GRIS
- 0058-SUPER HIGH GLOSS CLEAR
- 1026-CLEAR PRINT INOX
- 0056-50 GLOSS APPLIANCE CLEAR

Proceso de color:

- 0543-BLACK METALLIC
- 0139-NEW PRIME WHITE
- 0901-ROJO STD
- 1475-CHACOAL GRAY CKA3Y49403
- 0973-ROJO COLONIAL SP
- 0488-DIAMOND GRAY LAV
- 0147-BLANCO 15 II MOD ROHS
- 1164-BLANCO RAL 9010
- 2453-GRAY BACKER EDGE
- 1489-IRON ORE GRAY
- 0130-BLANCO MET 9010
- 1130-BLANCO RAL 9003 POLY
- 0606-VERDE KOOP SILIC.
- 2485-CAST IRON GRAY MET ROHS
- 1456-GRIS CLARO RC
- 0195-NEAT WHITE LAV ROHS
- 2409-PLATA 113 REF
- 0517-BLACK POLYESTER ROHS
- 0111-BLANCO 700 P/EXT
- 0466-SLATE GRAY HVAC TEXT
- 0443-PRIVATE LABEL GRAY
- 0202-AZUL REY
- 0936-ROJO WRINKLED CR
- 0156-BLANCO QUANTUM SAB
- 0110-DURAPLUS BLANCO STD
- 0525-NEGRO GRAFITO
- 1131-BLANCO RAL 9003 BACKER
- 0406-SILVER POLY
- 0101-BLANCO 15 II MOD.
- 0102-BLANCO STD
- 0969-ROJO OSCURO RC
- 1168-BLANCO RAL 9010 USDA
- 0433-PLATA 89
- 0587-MONOBACK BRILLANTE AR

- 1487-AMERICAN STERLING III
- 0700-ARENA STD
- 1493-GRAY 2001
- 0623-VERDE WRINKLED CR
- 1506-NEGRO MM MITSUI. II
- 0193-WHITE 12H2427 LAV ROHS
- 0691-JUNIPER GREEN METALLIC ROHS
- 1511-BLACK ORE METALLIC
- 0526-BLACK REF SAB
- 0435-APOLLO GRAY KRYSTAL KOTE
- 0463-GUN METAL GRAY
- 0426-GRIS PIZARRA TERNIUM
- 2474-EDGE PLATINUM GRAY TEXT
- 0672-VERDE MORISCO BB
- 0311-BROWN T413 SILIC.
- 0407-COOL GRAY ICP
- 0173-BLANCO SM
- 0121-BLANCO STD KYNAR
- 0446-WARM DARK GRAY SMOOTH
- 2158-BLANCO POLAR RF AR
- 1490-IRON ORE GRAY TXT
- 1510-BLACK ECOPRINT MATE
- 0590-MED GLOSS BLACK XT30
- 0934-ROJO RAL-8004 BACKER
- 0427-GRIS BACKER TERNIUM
- 0501-NEGRO ESTUFAS
- 0405-TAUPE METALLIC
- 0421-APOLLO GREY REF
- 0439-PRIVATE LABEL GRAY TEXT
- 0900-ROJO STD SILIC.
- 0502-NEGRO AT II
- 0318-TOPAZ METALLIC 14H2593
- 0601-VERDE PRIMSA
- 0702-ARENA STD SILIC.
- 0174-BLANCO STD USDA
- 0911-ROJO JANITZIO BB
- 0309-CAFE MOCHA
- 0159-DURAPLUS BCO COOL ROOF 75
- 0226-BREATHTAKING BLUE FC
- 2469-EDGEGARD QUARTZ GRAY TXT
- 0209-AZUL MILITAR SILIC
- 0966-ROJO RAL 8004 BC
- 2157-BLANCO STD SILIC
- 0995-ROSA KB LAV
- 0106-BLANCO CONFAD
- 0933-ROJO RAL-8004 POLY
- 2102-BLANCO RAL 9003 BC
- 0620-VERDE PINO RAL-6028 POLY

- 0153-BLANCO IMPERIAL JUV SILIC
- 0145-BLANCO COOL ROOF 75
- 1418-PRINTING SILVER INOX
- 0177-WHITE 800043 LAUNDRY
- 1340-DARK BROWN RAL 8022 SILIC
- 0447-WARM DARK GRAY TEXT.
- 0401-RANGE GRAY
- 0144-WHITE POLYESTER ROHS
- 0404-GRIS FONDO MC
- 0114-BLANCO QL
- 0503-NEGRO ERNA
- 2153-BLANCO PURO USDA ROHS AST
- 0730-ARENA STD USDA
- 1492-GRIS KB LAV
- 0728-SAND TEXTURED
- 0104-BLANCO SUPERMATIC
- 0944-LIBERTY RED PANT 2035C
- 0422-BACKER 0507 CR
- 1922-ROJO JANITZIO MONOCAPA
- 0524-BLACK KRYSTAL KOTE
- 0400-GRIS FONDO
- 0664-AQUA KB LAV
- 1431-GRIS 356 TEXT.
- 0158-WHITE KRYSTAL KOTE
- 2460-GRIS 356 REF
- 0504-BLACK QL
- 0498-GRIS FONDO 70 GLOSS
- 0919-ROJO TERNIUM CR
- 0534-BLACK TEXT KOTE
- 0204-DURAPLUS AZUL MILITAR
- 0744-AMARILLO TERNIUM CR FC
- 1145-BLANCO TERNIUM CR
- 0529-BLACK QUANTUM SAB II
- 0487-GRIS TERNIUM CR FC
- 0440-BALTIC GRAY HVAC
- 0553-DARK SLATE METALLIC
- 3139-CHARACTER WHITE ROHS

Proceso de primer:

- 0079-BECKRYPRIM 246
- 1015-CR FREE WHITE PRIMER
- 0078-BECKRYPRIM 243
- 0018-HIGH BUILT PRIMER
- 0018-HIGH BUILT PRIMER…
- 0043-PRIMARIO NEGRO 4866
- 0039-CR FREE POLYURETAN PRIMER
- 1005-COIL PRIMER ROHS
- 0008-PRIMARIO 4041
- 1043-WHITE PRIMER LAUNDRY ROHS

- 0021-UNIVERSAL PRIMER
- 0003-PRIMARIO 4435
- 1056-PRIMER LAUNDRY NCC ROHS
- 0055-BLACK PRIMER ROHS
- 1013-DYNAPRIME 2821
- 0038-PRIMER 917
- 0084-EDGEGARD PRIMER
- 0064-PO PRIMER 6334A
- 1030-PRIMER 4494
- 0001-PRIMER 4457
- 0090-PRIMER 60000

Lista general de todas las pinturas únicas:
- 0543-BLACK METALLIC
- 0139-NEW PRIME WHITE
- 0901-ROJO STD
- 0093-50 GLOSS CLEAR ONE PASS
- 0097-CLEAR APPLIANCE BACKER MC
- 1475-CHACOAL GRAY CKA3Y49403
- 0973-ROJO COLONIAL SP
- 0024-CLEAR LAUNDRY BACKER
- 0488-DIAMOND GRAY LAV
- 0147-BLANCO 15 II MOD ROHS
- 0079-BECKRYPRIM 246
- 1164-BLANCO RAL 9010
- 1015-CR FREE WHITE PRIMER
- 2453-GRAY BACKER EDGE
- 1489-IRON ORE GRAY
- 0130-BLANCO MET 9010
- 1130-BLANCO RAL 9003 POLY
- 0606-VERDE KOOP SILIC.
- 2485-CAST IRON GRAY MET ROHS
- 0066-SUPER HIGH GLOSS CLEAR II
- 0018-HIGH BUILT PRIMER
- 1456-GRIS CLARO RC
- 0195-NEAT WHITE LAV ROHS
- 2409-PLATA 113 REF
- 0034-CLEAR AZUL P/ESPUMA
- 1028-CLEAR BLUE EPOXY
- 0517-BLACK POLYESTER ROHS
- 0111-BLANCO 700 P/EXT
- 1043-WHITE PRIMER LAUNDRY ROHS
- 0466-SLATE GRAY HVAC TEXT
- 0443-PRIVATE LABEL GRAY
- 0202-AZUL REY
- 0936-ROJO WRINKLED CR
- 0156-BLANCO QUANTUM SAB
- 0110-DURAPLUS BLANCO STD

- 0525-NEGRO GRAFITO
- 1131-BLANCO RAL 9003 BACKER
- 1152-BLANCO TERNIUM CR TOP
- 0406-SILVER POLY
- 1030-PRIMER 4494
- 0101-BLANCO 15 II MOD.
- 0102-BLANCO STD
- 0969-ROJO OSCURO RC
- 1168-BLANCO RAL 9010 USDA
- 0433-PLATA 89
- 0001-PRIMER 4457
- 0090-PRIMER 60000
- 1078-CLEAR TOP 20 GLOSS
- 0587-MONOBACK BRILLANTE AR
- 1487-AMERICAN STERLING III
- 0700-ARENA STD
- 1493-GRAY 2001
- 0623-VERDE WRINKLED CR
- 1506-NEGRO MM MITSUI. II
- 0023-CLEAR EPOXICO P/ESPUMA
- 0193-WHITE 12H2427 LAV ROHS
- 0028-CLEAR P/ESPUMA
- 0691-JUNIPER GREEN METALLIC ROHS
- 0043-PRIMARIO NEGRO 4866
- 1511-BLACK ORE METALLIC
- 0526-BLACK REF SAB
- 0435-APOLLO GRAY KRYSTAL KOTE
- 0463-GUN METAL GRAY
- 0426-GRIS PIZARRA TERNIUM
- 2474-EDGE PLATINUM GRAY TEXT
- 0008-PRIMARIO 4041
- 1022-EPOXY CHROMATE BACKER
- 0672-VERDE MORISCO BB
- 0311-BROWN T413 SILIC.
- 0407-COOL GRAY ICP
- 0173-BLANCO SM
- 0121-BLANCO STD KYNAR
- 0446-WARM DARK GRAY SMOOTH
- 2158-BLANCO POLAR RF AR
- 1490-IRON ORE GRAY TXT
- 0003-PRIMARIO 4435
- 1510-BLACK ECOPRINT MATE
- 0590-MED GLOSS BLACK XT30
- 0934-ROJO RAL-8004 BACKER
- 0427-GRIS BACKER TERNIUM
- 0501-NEGRO ESTUFAS
- 0405-TAUPE METALLIC
- 0421-APOLLO GREY REF

- 0439-PRIVATE LABEL GRAY TEXT
- 0064-PO PRIMER 6334A
- 0900-ROJO STD SILIC.
- 0058-SUPER HIGH GLOSS CLEAR
- 1026-CLEAR PRINT INOX
- 0502-NEGRO AT II
- 0318-TOPAZ METALLIC 14H2593
- 0601-VERDE PRIMSA
- 0702-ARENA STD SILIC.
- 0174-BLANCO STD USDA
- 0911-ROJO JANITZIO BB
- 0309-CAFE MOCHA
- 0159-DURAPLUS BCO COOL ROOF 75
- 0226-BREATHTAKING BLUE FC
- 2469-EDGEGARD QUARTZ GRAY TXT
- 0209-AZUL MILITAR SILIC
- 0966-ROJO RAL 8004 BC
- 2157-BLANCO STD SILIC
- 0995-ROSA KB LAV
- 0078-BECKRYPRIM 243
- 0106-BLANCO CONFAD
- 0051-80 GLOSS CLEAR BACKER.
- 0018-HIGH BUILT PRIMER…
- 0068-APPLIANCE 10 GLOSS CLEAR
- 0933-ROJO RAL-8004 POLY
- 0099-CLEAR P/ESPUMA ROHS
- 0029-CLEAR INOXIDABLE
- 1005-COIL PRIMER ROHS
- 2102-BLANCO RAL 9003 BC
- 0620-VERDE PINO RAL-6028 POLY
- 1009-APPLIANCE 30 GLOSS CLR OP
- 0153-BLANCO IMPERIAL JUV SILIC
- 0054-CLEAR KRYSTAL KOTE
- 1077-CLEAR PRINT INOX MATE
- 0021-UNIVERSAL PRIMER
- 1055-CLEAR TOP LAVADORAS
- 0145-BLANCO COOL ROOF 75
- 1418-PRINTING SILVER INOX
- 0177-WHITE 800043 LAUNDRY
- 1340-DARK BROWN RAL 8022 SILIC
- 0447-WARM DARK GRAY TEXT.
- 1013-DYNAPRIME 2821
- 0084-EDGEGARD PRIMER
- 0401-RANGE GRAY
- 0025-CLEAR EPOXY (WASH COAT)
- 1070-CLEAR P/ESPUMA GRIS
- 0144-WHITE POLYESTER ROHS
- 0056-50 GLOSS APPLIANCE CLEAR

- 0404-GRIS FONDO MC
- 0114-BLANCO QL
- 0503-NEGRO ERNA
- 2153-BLANCO PURO USDA ROHS AST
- 0730-ARENA STD USDA
- 1492-GRIS KB LAV
- 0728-SAND TEXTURED
- 0104-BLANCO SUPERMATIC
- 0944-LIBERTY RED PANT 2035C
- 0075-CLEAR LAUNDRY BACKER ROHS
- 0422-BACKER 0507 CR
- 1922-ROJO JANITZIO MONOCAPA
- 0524-BLACK KRYSTAL KOTE
- 0400-GRIS FONDO
- 0664-AQUA KB LAV
- 1431-GRIS 356 TEXT.
- 0039-CR FREE POLYURETAN PRIMER
- 0158-WHITE KRYSTAL KOTE
- 2460-GRIS 356 REF
- 0504-BLACK QL
- 0092-80 GLOSS CLEAR LAV
- 0027-80 GLOSS CLEAR BACKER II
- 0498-GRIS FONDO 70 GLOSS
- 0919-ROJO TERNIUM CR
- 0534-BLACK TEXT KOTE
- 0204-DURAPLUS AZUL MILITAR
- 0744-AMARILLO TERNIUM CR FC
- 1056-PRIMER LAUNDRY NCC ROHS
- 1145-BLANCO TERNIUM CR
- 0529-BLACK QUANTUM SAB II
- 0055-BLACK PRIMER ROHS
- 0038-PRIMER 917
- 0063-CLEAR KYNAR UV
- 0487-GRIS TERNIUM CR FC
- 0440-BALTIC GRAY HVAC
- 0036-BACK PRIMER
- 0553-DARK SLATE METALLIC
- 3139-CHARACTER WHITE ROHS

Uso de pinturas en procesos:
- Pintura 0543-BLACK METALLIC se utiliza en el(los) proceso(s) de color.
- Pintura 0139-NEW PRIME WHITE se utiliza en el(los) proceso(s) de color.
- Pintura 0901-ROJO STD se utiliza en el(los) proceso(s) de color.
- Pintura 0093-50 GLOSS CLEAR ONE PASS se utiliza en el(los) proceso(s) de clear.
- Pintura 0097-CLEAR APPLIANCE BACKER MC se utiliza en el(los) proceso(s) de clear.
- Pintura 1475-CHACOAL GRAY CKA3Y49403 se utiliza en el(los) proceso(s) de

color.
 - Pintura 0973-ROJO COLONIAL SP se utiliza en el(los) proceso(s) de color.
 - Pintura 0024-CLEAR LAUNDRY BACKER se utiliza en el(los) proceso(s) de clear.
 - Pintura 0488-DIAMOND GRAY LAV se utiliza en el(los) proceso(s) de color.
 - Pintura 0147-BLANCO 15 II MOD ROHS se utiliza en el(los) proceso(s) de color.
 - Pintura 0079-BECKRYPRIM 246 se utiliza en el(los) proceso(s) de primer.
 - Pintura 1164-BLANCO RAL 9010 se utiliza en el(los) proceso(s) de color.
 - Pintura 1015-CR FREE WHITE PRIMER se utiliza en el(los) proceso(s) de primer.
 - Pintura 2453-GRAY BACKER EDGE se utiliza en el(los) proceso(s) de color.
 - Pintura 1489-IRON ORE GRAY se utiliza en el(los) proceso(s) de color.
 - Pintura 0130-BLANCO MET 9010 se utiliza en el(los) proceso(s) de color.
 - Pintura 1130-BLANCO RAL 9003 POLY se utiliza en el(los) proceso(s) de color.
 - Pintura 0606-VERDE KOOP SILIC. se utiliza en el(los) proceso(s) de color.
 - Pintura 2485-CAST IRON GRAY MET ROHS se utiliza en el(los) proceso(s) de color.
 - Pintura 0066-SUPER HIGH GLOSS CLEAR II se utiliza en el(los) proceso(s) de clear.
 - Pintura 0018-HIGH BUILT PRIMER se utiliza en el(los) proceso(s) de primer.
 - Pintura 1456-GRIS CLARO RC se utiliza en el(los) proceso(s) de color.
 - Pintura 0195-NEAT WHITE LAV ROHS se utiliza en el(los) proceso(s) de color.
 - Pintura 2409-PLATA 113 REF se utiliza en el(los) proceso(s) de color.
 - Pintura 0034-CLEAR AZUL P/ESPUMA se utiliza en el(los) proceso(s) de clear.
 - Pintura 1028-CLEAR BLUE EPOXY se utiliza en el(los) proceso(s) de clear.
 - Pintura 0517-BLACK POLYESTER ROHS se utiliza en el(los) proceso(s) de color.
 - Pintura 0111-BLANCO 700 P/EXT se utiliza en el(los) proceso(s) de color.
 - Pintura 1043-WHITE PRIMER LAUNDRY ROHS se utiliza en el(los) proceso(s) de primer.
 - Pintura 0466-SLATE GRAY HVAC TEXT se utiliza en el(los) proceso(s) de color.
 - Pintura 0443-PRIVATE LABEL GRAY se utiliza en el(los) proceso(s) de color.
 - Pintura 0202-AZUL REY se utiliza en el(los) proceso(s) de color.
 - Pintura 0936-ROJO WRINKLED CR se utiliza en el(los) proceso(s) de color.
 - Pintura 0156-BLANCO QUANTUM SAB se utiliza en el(los) proceso(s) de color.
 - Pintura 0110-DURAPLUS BLANCO STD se utiliza en el(los) proceso(s) de color.
 - Pintura 0525-NEGRO GRAFITO se utiliza en el(los) proceso(s) de color.
 - Pintura 1131-BLANCO RAL 9003 BACKER se utiliza en el(los) proceso(s) de color.
 - Pintura 1152-BLANCO TERNIUM CR TOP se utiliza en el(los) proceso(s) de clear.
 - Pintura 0406-SILVER POLY se utiliza en el(los) proceso(s) de color.
 - Pintura 1030-PRIMER 4494 se utiliza en el(los) proceso(s) de primer.
 - Pintura 0101-BLANCO 15 II MOD. se utiliza en el(los) proceso(s) de color.
 - Pintura 0102-BLANCO STD se utiliza en el(los) proceso(s) de color.
 - Pintura 0969-ROJO OSCURO RC se utiliza en el(los) proceso(s) de color.
 - Pintura 1168-BLANCO RAL 9010 USDA se utiliza en el(los) proceso(s) de color.
 - Pintura 0433-PLATA 89 se utiliza en el(los) proceso(s) de color.
 - Pintura 0001-PRIMER 4457 se utiliza en el(los) proceso(s) de primer.
 - Pintura 0090-PRIMER 60000 se utiliza en el(los) proceso(s) de primer.
 - Pintura 1078-CLEAR TOP 20 GLOSS se utiliza en el(los) proceso(s) de clear.
 - Pintura 0587-MONOBACK BRILLANTE AR se utiliza en el(los) proceso(s) de color.

- Pintura 1487-AMERICAN STERLING III se utiliza en el(los) proceso(s) de color.
- Pintura 0700-ARENA STD se utiliza en el(los) proceso(s) de color.
- Pintura 1493-GRAY 2001 se utiliza en el(los) proceso(s) de color.
- Pintura 0623-VERDE WRINKLED CR se utiliza en el(los) proceso(s) de color.
- Pintura 1506-NEGRO MM MITSUI. II se utiliza en el(los) proceso(s) de color.
- Pintura 0023-CLEAR EPOXICO P/ESPUMA se utiliza en el(los) proceso(s) de clear.
- Pintura 0193-WHITE 12H2427 LAV ROHS se utiliza en el(los) proceso(s) de color.
- Pintura 0028-CLEAR P/ESPUMA se utiliza en el(los) proceso(s) de clear.
- Pintura 0691-JUNIPER GREEN METALLIC ROHS se utiliza en el(los) proceso(s) de color.
- Pintura 0043-PRIMARIO NEGRO 4866 se utiliza en el(los) proceso(s) de primer.
- Pintura 1511-BLACK ORE METALLIC se utiliza en el(los) proceso(s) de color.
- Pintura 0526-BLACK REF SAB se utiliza en el(los) proceso(s) de color.
- Pintura 0435-APOLLO GRAY KRYSTAL KOTE se utiliza en el(los) proceso(s) de color.
- Pintura 0463-GUN METAL GRAY se utiliza en el(los) proceso(s) de color.
- Pintura 0426-GRIS PIZARRA TERNIUM se utiliza en el(los) proceso(s) de color.
- Pintura 2474-EDGE PLATINUM GRAY TEXT se utiliza en el(los) proceso(s) de color.
- Pintura 0008-PRIMARIO 4041 se utiliza en el(los) proceso(s) de primer.
- Pintura 1022-EPOXY CHROMATE BACKER se utiliza en el(los) proceso(s) de clear.
- Pintura 0672-VERDE MORISCO BB se utiliza en el(los) proceso(s) de color.
- Pintura 0311-BROWN T413 SILIC. se utiliza en el(los) proceso(s) de color.
- Pintura 0407-COOL GRAY ICP se utiliza en el(los) proceso(s) de color.
- Pintura 0173-BLANCO SM se utiliza en el(los) proceso(s) de color.
- Pintura 0121-BLANCO STD KYNAR se utiliza en el(los) proceso(s) de color.
- Pintura 0446-WARM DARK GRAY SMOOTH se utiliza en el(los) proceso(s) de color.
- Pintura 2158-BLANCO POLAR RF AR se utiliza en el(los) proceso(s) de color.
- Pintura 1490-IRON ORE GRAY TXT se utiliza en el(los) proceso(s) de color.
- Pintura 0003-PRIMARIO 4435 se utiliza en el(los) proceso(s) de primer.
- Pintura 1510-BLACK ECOPRINT MATE se utiliza en el(los) proceso(s) de color.
- Pintura 0590-MED GLOSS BLACK XT30 se utiliza en el(los) proceso(s) de color.
- Pintura 0934-ROJO RAL-8004 BACKER se utiliza en el(los) proceso(s) de color.
- Pintura 0427-GRIS BACKER TERNIUM se utiliza en el(los) proceso(s) de color.
- Pintura 0501-NEGRO ESTUFAS se utiliza en el(los) proceso(s) de color.
- Pintura 0405-TAUPE METALLIC se utiliza en el(los) proceso(s) de color.
- Pintura 0421-APOLLO GREY REF se utiliza en el(los) proceso(s) de color.
- Pintura 0439-PRIVATE LABEL GRAY TEXT se utiliza en el(los) proceso(s) de color.
- Pintura 0064-PO PRIMER 6334A se utiliza en el(los) proceso(s) de primer.
- Pintura 0900-ROJO STD SILIC. se utiliza en el(los) proceso(s) de color.
- Pintura 0058-SUPER HIGH GLOSS CLEAR se utiliza en el(los) proceso(s) de clear.
- Pintura 1026-CLEAR PRINT INOX se utiliza en el(los) proceso(s) de clear.
- Pintura 0502-NEGRO AT II se utiliza en el(los) proceso(s) de color.
- Pintura 0318-TOPAZ METALLIC 14H2593 se utiliza en el(los) proceso(s) de

color.
 - Pintura 0601-VERDE PRIMSA se utiliza en el(los) proceso(s) de color.
 - Pintura 0702-ARENA STD SILIC. se utiliza en el(los) proceso(s) de color.
 - Pintura 0174-BLANCO STD USDA se utiliza en el(los) proceso(s) de color.
 - Pintura 0911-ROJO JANITZIO BB se utiliza en el(los) proceso(s) de color.
 - Pintura 0309-CAFE MOCHA se utiliza en el(los) proceso(s) de color.
 - Pintura 0159-DURAPLUS BCO COOL ROOF 75 se utiliza en el(los) proceso(s) de color.
 - Pintura 0226-BREATHTAKING BLUE FC se utiliza en el(los) proceso(s) de color.
 - Pintura 2469-EDGEGARD QUARTZ GRAY TXT se utiliza en el(los) proceso(s) de color.
 - Pintura 0209-AZUL MILITAR SILIC se utiliza en el(los) proceso(s) de color.
 - Pintura 0966-ROJO RAL 8004 BC se utiliza en el(los) proceso(s) de color.
 - Pintura 2157-BLANCO STD SILIC se utiliza en el(los) proceso(s) de color.
 - Pintura 0995-ROSA KB LAV se utiliza en el(los) proceso(s) de color.
 - Pintura 0078-BECKRYPRIM 243 se utiliza en el(los) proceso(s) de primer.
 - Pintura 0106-BLANCO CONFAD se utiliza en el(los) proceso(s) de color.
 - Pintura 0051-80 GLOSS CLEAR BACKER. se utiliza en el(los) proceso(s) de clear.
 - Pintura 0018-HIGH BUILT PRIMER… se utiliza en el(los) proceso(s) de primer.
 - Pintura 0068-APPLIANCE 10 GLOSS CLEAR se utiliza en el(los) proceso(s) de clear.
 - Pintura 0933-ROJO RAL-8004 POLY se utiliza en el(los) proceso(s) de color.
 - Pintura 0099-CLEAR P/ESPUMA ROHS se utiliza en el(los) proceso(s) de clear.
 - Pintura 0029-CLEAR INOXIDABLE se utiliza en el(los) proceso(s) de clear.
 - Pintura 1005-COIL PRIMER ROHS se utiliza en el(los) proceso(s) de primer.
 - Pintura 2102-BLANCO RAL 9003 BC se utiliza en el(los) proceso(s) de color.
 - Pintura 0620-VERDE PINO RAL-6028 POLY se utiliza en el(los) proceso(s) de color.
 - Pintura 1009-APPLIANCE 30 GLOSS CLR OP se utiliza en el(los) proceso(s) de clear.
 - Pintura 0153-BLANCO IMPERIAL JUV SILIC se utiliza en el(los) proceso(s) de color.
 - Pintura 0054-CLEAR KRYSTAL KOTE se utiliza en el(los) proceso(s) de clear.
 - Pintura 1077-CLEAR PRINT INOX MATE se utiliza en el(los) proceso(s) de clear.
 - Pintura 0021-UNIVERSAL PRIMER se utiliza en el(los) proceso(s) de primer.
 - Pintura 1055-CLEAR TOP LAVADORAS se utiliza en el(los) proceso(s) de clear.
 - Pintura 0145-BLANCO COOL ROOF 75 se utiliza en el(los) proceso(s) de color.
 - Pintura 1418-PRINTING SILVER INOX se utiliza en el(los) proceso(s) de color.
 - Pintura 0177-WHITE 800043 LAUNDRY se utiliza en el(los) proceso(s) de color.
 - Pintura 1340-DARK BROWN RAL 8022 SILIC se utiliza en el(los) proceso(s) de color.
 - Pintura 0447-WARM DARK GRAY TEXT. se utiliza en el(los) proceso(s) de color.
 - Pintura 1013-DYNAPRIME 2821 se utiliza en el(los) proceso(s) de primer.
 - Pintura 0084-EDGEGARD PRIMER se utiliza en el(los) proceso(s) de primer.
 - Pintura 0401-RANGE GRAY se utiliza en el(los) proceso(s) de color.
 - Pintura 0025-CLEAR EPOXY (WASH COAT) se utiliza en el(los) proceso(s) de clear.

- Pintura 1070-CLEAR P/ESPUMA GRIS se utiliza en el(los) proceso(s) de clear.
- Pintura 0144-WHITE POLYESTER ROHS se utiliza en el(los) proceso(s) de color.
- Pintura 0056-50 GLOSS APPLIANCE CLEAR se utiliza en el(los) proceso(s) de clear.
- Pintura 0404-GRIS FONDO MC se utiliza en el(los) proceso(s) de color.
- Pintura 0114-BLANCO QL se utiliza en el(los) proceso(s) de color.
- Pintura 0503-NEGRO ERNA se utiliza en el(los) proceso(s) de color.
- Pintura 2153-BLANCO PURO USDA ROHS AST se utiliza en el(los) proceso(s) de color.
- Pintura 0730-ARENA STD USDA se utiliza en el(los) proceso(s) de color.
- Pintura 1492-GRIS KB LAV se utiliza en el(los) proceso(s) de color.
- Pintura 0728-SAND TEXTURED se utiliza en el(los) proceso(s) de color.
- Pintura 0104-BLANCO SUPERMATIC se utiliza en el(los) proceso(s) de color.
- Pintura 0944-LIBERTY RED PANT 2035C se utiliza en el(los) proceso(s) de color.
- Pintura 0075-CLEAR LAUNDRY BACKER ROHS se utiliza en el(los) proceso(s) de clear.
- Pintura 0422-BACKER 0507 CR se utiliza en el(los) proceso(s) de color.
- Pintura 1922-ROJO JANITZIO MONOCAPA se utiliza en el(los) proceso(s) de color.
- Pintura 0524-BLACK KRYSTAL KOTE se utiliza en el(los) proceso(s) de color.
- Pintura 0400-GRIS FONDO se utiliza en el(los) proceso(s) de color.
- Pintura 0664-AQUA KB LAV se utiliza en el(los) proceso(s) de color.
- Pintura 1431-GRIS 356 TEXT. se utiliza en el(los) proceso(s) de color.
- Pintura 0039-CR FREE POLYURETAN PRIMER se utiliza en el(los) proceso(s) de primer.
- Pintura 0158-WHITE KRYSTAL KOTE se utiliza en el(los) proceso(s) de color.
- Pintura 2460-GRIS 356 REF se utiliza en el(los) proceso(s) de color.
- Pintura 0504-BLACK QL se utiliza en el(los) proceso(s) de color.
- Pintura 0092-80 GLOSS CLEAR LAV se utiliza en el(los) proceso(s) de clear.
- Pintura 0027-80 GLOSS CLEAR BACKER II se utiliza en el(los) proceso(s) de clear.
- Pintura 0498-GRIS FONDO 70 GLOSS se utiliza en el(los) proceso(s) de color.
- Pintura 0919-ROJO TERNIUM CR se utiliza en el(los) proceso(s) de color.
- Pintura 0534-BLACK TEXT KOTE se utiliza en el(los) proceso(s) de color.
- Pintura 0204-DURAPLUS AZUL MILITAR se utiliza en el(los) proceso(s) de color.
- Pintura 0744-AMARILLO TERNIUM CR FC se utiliza en el(los) proceso(s) de clear y color.
- Pintura 1056-PRIMER LAUNDRY NCC ROHS se utiliza en el(los) proceso(s) de primer.
- Pintura 1145-BLANCO TERNIUM CR se utiliza en el(los) proceso(s) de color.
- Pintura 0529-BLACK QUANTUM SAB II se utiliza en el(los) proceso(s) de color.
- Pintura 0055-BLACK PRIMER ROHS se utiliza en el(los) proceso(s) de primer.
- Pintura 0038-PRIMER 917 se utiliza en el(los) proceso(s) de primer.
- Pintura 0063-CLEAR KYNAR UV se utiliza en el(los) proceso(s) de clear.
- Pintura 0487-GRIS TERNIUM CR FC se utiliza en el(los) proceso(s) de color.
- Pintura 0440-BALTIC GRAY HVAC se utiliza en el(los) proceso(s) de color.
- Pintura 0036-BACK PRIMER se utiliza en el(los) proceso(s) de clear.

```
- Pintura 0553-DARK SLATE METALLIC se utiliza en el(los) proceso(s) de color.
- Pintura 3139-CHARACTER WHITE ROHS se utiliza en el(los) proceso(s) de color.
```

Información sobre colores repetidos en cada tipo de proceso

[37]:
```python
# Definimos diccionarios para agrupar los identificadores de pinturas por tipo de proceso
clear_codes = {'inferior_clear_code', 'superior_clear_code'}
primer_codes = {'inferior_primer_code', 'superior_primer_code'}
color_codes = {'inferior_color_code', 'superior_color_code'}

# Creamos DataFrames vacíos para cada tipo de proceso
clear_paints = pd.DataFrame(columns=['Paint Code', 'Process Type', 'Count'])
primer_paints = pd.DataFrame(columns=['Paint Code', 'Process Type', 'Count'])
color_paints = pd.DataFrame(columns=['Paint Code', 'Process Type', 'Count'])

# Función para contar y agregar pinturas por tipo de proceso
def add_paint_count(row, process_dict, process_name, df):
    for col_name in process_dict:
        paint_code = row[col_name]
        if pd.notna(paint_code):  # Verificamos que el código de pintura no sea NaN
            if not df[df['Paint Code'] == paint_code].empty:
                df.loc[df['Paint Code'] == paint_code, 'Count'] += 1
            else:
                df = pd.concat([df, pd.DataFrame([[paint_code, process_name, 1]], columns=df.columns)], ignore_index=True)
    return df

# Recorremos el DataFrame para contar y agrupar pinturas por proceso
for index, row in rows_with_repeated_values.iterrows():
    clear_paints = add_paint_count(row, clear_codes, 'Clear', clear_paints)
    primer_paints = add_paint_count(row, primer_codes, 'Primer', primer_paints)
    color_paints = add_paint_count(row, color_codes, 'Color', color_paints)

# Unimos los resultados para tener un resumen completo
all_paints = pd.concat([clear_paints, primer_paints, color_paints]).reset_index(drop=True)

# Mostramos las pinturas existentes y su uso por proceso
print("Summary of Paint Use by Process:")
#print(all_paints.sort_values(by=['Process Type', 'Count'], ascending=[True, False]))

# Para ver cuáles pinturas son las más usadas en cada proceso
print("\nMost Used Paints by Process:")
for process_type, group_df in all_paints.groupby('Process Type'):
```

```
    most_used = group_df.loc[group_df['Count'].idxmax()]
    print(f"{process_type}: {most_used['Paint Code']} used {most_used['Count']}␣
 ↪times")

all_paints.sort_values(by=['Process Type', 'Count'], ascending=[True, False])
```

```
Summary of Paint Use by Process:

Most Used Paints by Process:
Clear: 0028-CLEAR P/ESPUMA used 2365 times
Color: 0601-VERDE PRIMSA used 3764 times
Primer: 0001-PRIMER 4457 used 3665 times
```

[37]:

	Paint Code	Process Type	Count
4	0028-CLEAR P/ESPUMA	Clear	2365
3	0054-CLEAR KRYSTAL KOTE	Clear	1069
5	0023-CLEAR EPOXICO P/ESPUMA	Clear	978
12	0024-CLEAR LAUNDRY BACKER	Clear	816
17	0097-CLEAR APPLIANCE BACKER MC	Clear	577
..	…	…	…
47	1030-PRIMER 4494	Primer	8
42	0018-HIGH BUILT PRIMER…	Primer	5
46	1043-WHITE PRIMER LAUNDRY ROHS	Primer	4
49	0064-PO PRIMER 6334A	Primer	2
40	1056-PRIMER LAUNDRY NCC ROHS	Primer	1

[170 rows x 3 columns]

1.4 4 .Valores faltantes

1.4.1 4.1 Suma de nulos por variable

4.1.1 Análisis de consumo de pintura En el resultado podemos observar cómo no existe algún valor nulo dentro de las variables que seleccionamos para esta base de datos, estos son datos que son necesarios dentro del proceso de pintado.

[38]:
```
# Calcular el total de valores nulos para cada columna
paint_null_values= paint_df.isnull().sum()

print("paint_df , Null values per column:")
paint_null_values
```

```
paint_df , Null values per column:
```

[38]:

production_line	0
month	0
month_number	0
paint	0
real_consumption	0

```
theoretical_consumption        0
consumption_difference         0
average_yield                  0
real_yield                     0
yield_difference               0
real_produced_square_meters    0
new_average_yield              0
dtype: int64
```

4.1.2 Pinturas y revestidos En el resultado podemos observar cómo no existe algún valor nulo dentro de las variables que seleccionamos para esta base de datos, estos son datos que son necesarios en la evaluación del proceso de las pinturas. En donde se identifica datos importantes de la pintura utilizada como proveedor y precio.

[39]:
```
# Calcular el total de valores nulos para cada columna
coating_null_values= coating_df.isnull().sum()

print("coating_df, Null values per column:")
coating_null_values
```

```
coating_df, Null values per column:
```

[39]:
```
production_line       0
paint_id              0
paint_name            0
monetary_value_usd    0
total_liters_used     0
production_plant      0
supplier              0
date                  0
hour                  0
price_per_liter       0
dtype: int64
```

4.1.3 Resumen de producción En el resultado podemos observar cómo aquí si existen valores nulos en algunas variables de la base de datos. Esto puede llegar a pasar porque no se realizó el proceso necesario o hasta no se guardó el dato usado.

[40]:
```
# Calcular el total de valores nulos para cada columna
production_null_values= production_df.isnull().sum()

print("production_df, Null values per column:")
production_null_values
```

```
production_df, Null values per column:
```

[40]:
```
input_material_code         0
output_material_code        0
```

```
start_date                  0
end_date                    0
client_name                 0
inferior_clear_code     10197
superior_clear_code     14167
width1_mm                   0
width2_mm                   0
width3_mm                   0
width_mm                    0
thickness1_mm               0
thickness2_mm               0
thickness3_mm               0
thickness_mm                0
input_weight_kg             0
weight_kg                   0
length_m                    0
inferior_color_code     10999
superior_color_code      5338
superior_primer_code    10297
inferior_primer_code    13059
route                       0
dtype: int64
```

4.1.4 Defectos de producción En este resultado podemos ver como la mayoría de las columnas tienen nulos, esto se debe a que las pinturas no fueron detectadas con algún defecto dentro del proceso de la revisión.

[41]:
```
# Calcular el total de valores nulos para cada columna
defects_null_values= defects_df.isnull().sum()

print("defects_df, null values per column:")
defects_null_values
```

```
defects_df, null values per column:
```

[41]:
```
defect_code          0
defect_name          0
location           313
is_containment      78
is_preventive       78
intensity         1639
face              1769
side              1575
frequency         1847
register_date        0
dtype: int64
```

1.4.2 4.2 Indentificacion de columnas con mayor porcentaje de 15% datos nulos

El código calcula primero las filas del dataframe, después calcula el porcentaje de los valores nulos por cada columna y finalmente filtra los datos con el porcentaje seleccionado (en este caso 15%). Si identifica valores mayores a 15% siendo nulos, te imprime las columnas, de no ser así imprime un mensaje comentando que no hay nulos arriba del %15.

4.2.1 Análisis de consumo de pintura

[42]:
```
total = paint_df.shape[0]

null_values_percentage= (paint_df.isnull().sum() / total) * 100

total_nulls= null_values_percentage[null_values_percentage > 15]
if not total_nulls.empty:
    print("Columns with more than 15% null data:" ,total_nulls)
else:
    print("No columns above 15 percent null data")
```

```
No columns above 15 percent null data
```

4.2.2 Pinturas y revestidos

[43]:
```
total = coating_df.shape[0]

null_values_percentage= (coating_df.isnull().sum() / total) * 100

total_nulls= null_values_percentage[null_values_percentage > 15]
if not total_nulls.empty:
    print("Columns with more than 15% null data:" ,total_nulls)
else:
    print("No columns above 15 percent null data")
```

```
No columns above 15 percent null data
```

4.2.3 Resumen de producción

[44]:
```
total = production_df.shape[0]

null_values_percentage= (production_df.isnull().sum() / total) * 100

total_nulls= null_values_percentage[null_values_percentage > 15]
if not total_nulls.empty:
    print("Columns with more than 15% null data: \n" ,total_nulls)
else:
    print("No columns above 15 percent null data ")
```

```
Columns with more than 15% null data:
 inferior_clear_code     63.627855
superior_clear_code     88.400100
inferior_color_code     68.632223
superior_color_code     33.308374
```

```
superior_primer_code    64.251841
inferior_primer_code    81.486335
dtype: float64
```

4.2.4 Defectos de producción

[45]:
```
total = defects_df.shape[0]

null_values_percentage= (defects_df.isnull().sum() / total) * 100

total_nulls= null_values_percentage[null_values_percentage > 15]
if not total_nulls.empty:
    print("Columns with more than 15% null data: \n" ,total_nulls)
else:
    print("No columns above 15 percent null data ")
```

```
Columns with more than 15% null data:
 intensity    65.324831
face          70.506178
side          62.774014
frequency     73.614986
dtype: float64
```

1.5 5 Registros duplicados

Para cada una de los dataframes se utilizó la función de duplicated para identificar las filas duplicadas en cada uno de ellos.

5.1 Resumen de producción

[46]:
```
production_df[production_df.duplicated()]
```

[46]:
```
Empty DataFrame
Columns: [input_material_code, output_material_code, start_date, end_date,
client_name, inferior_clear_code, superior_clear_code, width1_mm, width2_mm,
width3_mm, width_mm, thickness1_mm, thickness2_mm, thickness3_mm, thickness_mm,
input_weight_kg, weight_kg, length_m, inferior_color_code, superior_color_code,
superior_primer_code, inferior_primer_code, route]
Index: []

[0 rows x 23 columns]
```

5.2 Análisis de consumo de pintura

[47]:
```
paint_df[paint_df.duplicated()]
```

[47]:
```
Empty DataFrame
Columns: [production_line, month, month_number, paint, real_consumption,
theoretical_consumption, consumption_difference, average_yield, real_yield,
yield_difference, real_produced_square_meters, new_average_yield]
Index: []
```

5.3 Pinturas y revestidos

[48]:
```
defects_df[defects_df.duplicated(keep=False)]
```

[48]:

	defect_code	defect_name	location	is_containment	is_preventive	intensity	face	side	frequency	register_date
830	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-05-30 20:12:00
831	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-05-30 20:12:00
864	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-06-06 16:08:00
865	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-06-06 16:08:00
918	40.0	PESO (-)	TODO	0.0	0.0	NaN	None	None	None	2023-06-16 01:56:00
919	40.0	PESO (-)	TODO	0.0	0.0	NaN	None	None	None	2023-06-16 01:56:00
1350	647.0	CINTA DE NOCHE	None	NaN	NaN	NaN	None	None	None	2023-07-14 05:39:00
1351	647.0	CINTA DE NOCHE	None	NaN	NaN	NaN	None	None	None	2023-07-14 05:39:00
1699	674.0	MANCHAS SIN GALVANIZAR	TODO	0.0	0.0	3.0	AMBAS	AMBOS LADOS	CONTINUA	2023-08-02 19:25:00
1700	674.0	MANCHAS SIN GALVANIZAR	TODO	0.0	0.0	3.0	AMBAS	AMBOS LADOS	CONTINUA	2023-08-02 19:25:00
2228	133.0	ANCHO (+/-)	TODO	0.0	0.0	NaN	None	None	None	2023-05-30 14:13:00
2229	133.0	ANCHO (+/-)	TODO	0.0	0.0	NaN	None	None	None	2023-05-30 14:13:00
2393	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-05-30 20:12:00
2394	40.0	PESO (-)	TODO	1.0	1.0	NaN	None	None	None	2023-05-30 20:12:00

```
2229 2023-05-30 14:13:00
2393 2023-07-24 10:11:00
2394 2023-07-24 10:11:00
```

[49]:
```
coating_df[coating_df.duplicated()]
```

[49]:
```
             production_line    paint_id                    paint_name  \
508     Pintado 1 Prod. Univ  I1248_VALS        0054-CLEAR KRYSTAL KOTE
509     Pintado 1 Prod. Univ  I1248_VALS        0054-CLEAR KRYSTAL KOTE
510     Pintado 1 Prod. Univ  I1248_VALS        0054-CLEAR KRYSTAL KOTE
511     Pintado 1 Prod. Univ  I1248_VALS        0054-CLEAR KRYSTAL KOTE
531     Pintado 1 Prod. Univ  I4461_VALS  0435-APOLLO GRAY KRYSTAL KOTE
…                        …        …                             …
211307         Pintado 2 UNI  I0604_OSEL              0601-VERDE PRIMSA
211313         Pintado 1 UNI  I0604_OSEL              0601-VERDE PRIMSA
211443         Pintado 1 UNI  I0604_OSEL              0601-VERDE PRIMSA
211497         Pintado 2 UNI  I1116_AKZO             0028-CLEAR P/ESPUMA
211567         Pintado 1 UNI  I0604_OSEL              0601-VERDE PRIMSA

        monetary_value_usd  total_liters_used production_plant supplier  \
508                 1882.0              200.0              Uni     VALS
509                 1882.0              200.0              Uni     VALS
510                 1882.0              200.0              Uni     VALS
511                 1882.0              200.0              Uni     VALS
531                 1856.0              200.0              Uni     VALS
…                      …                  …                …       …
211307               438.0              100.0              Uni     OSEL
211313               438.0              100.0              Uni     OSEL
211443               350.4               80.0              Uni     OSEL
211497               307.2               60.0              Uni     AKZO
211567               219.0               50.0              Uni     OSEL

              date      hour  price_per_liter
508     24.07.2020  23:16:33             9.41
509     24.07.2020  23:16:33             9.41
510     24.07.2020  23:16:33             9.41
511     24.07.2020  23:16:33             9.41
531     28.07.2020  03:41:16             9.28
…              …         …                …
211307  04.08.2023  00:00:00             4.38
211313  30.08.2023  00:00:00             4.38
211443  21.08.2023  00:00:00             4.38
211497  03.08.2023  00:00:00             5.12
211567  17.08.2023  00:00:00             4.38

[8380 rows x 10 columns]
```

1.6 Contribución individual

Hiram Muñoz Construcción del *script* de procesamiento de datos, verificación de tipos de datos, selección y eliminación de columnas innecesarias. #### Raúl Murillo Verificación de la columna de diferencia de las columnas real y teórico y de la verificación de los metros cuadrados reales.

Andrea Garza Verificación del rendimiento real y diferencia con el rendimiento estándar, Calculo e identificación de diferencias en el promedio de Ancho y Espesor e Identificación de valores faltantes.

Erick Hernández Calculo de nuevo rendimiento std, identificación de renglones duplicados y verificación de metros cuadrados reales

David Martínez Cambio de nombres de variables y analizar información sobre pinturas repetidas en los renglones de producción (encontrados duplicados y segmentadas pinturas que aparecen por cada proceso)

Distribucion de Tareas: https://docs.google.com/spreadsheets/d/1Nx3N0PPtPqMVHnxO9pkpvJUUQX7RM

1.7 Referencias

Apache Software Foundation. (s. f.). Feather File Format — Apache Arrow v15.0.1. Recuperado 12 de marzo de 2024, de https://arrow.apache.org/docs/python/feather.html