CLEANING DATA

Para ver fechas fuera del rango posible → ver fecha de hoy → `datetime.date.today()` → Filtramos fuera de esto

Valores duplicados

Para encontrar duplicados → `dup = df.duplicated()` (serie booleana)
- subset = [..] → lista de columnas donde buscar duplicidad
- keep = → que valor quedarse 'first', 'last' ...

Para borrar duplicados → `df.drop_duplicates(inplace=True)` → Así solo borramos duplicados completos → Tiene también subset y keep

Datos categoricos

Para buscar datos con categorias que no existan → `malos = set(df['col_cat']).difference(categorias posibles)`

Consistencia → Mayusculas, Espacios

Para agrupar columnas numericas en categorias → `df['col_cat'] = pd.qcut(df['col'], q=3, labels=['..', '...'])` (q → Cuantos grupos)

Si definimos nosotros los bloques → bloques [0, 100, 150, ..] ... `pd.cut(df['col'], bins=bloques, labels=...)`

Uniformidad en los datos

Tienen que estar en las mismas unidades y formato (en caso de fechas)

Para buscar en una columna de datetime algo en concreto → `df['col'].dt.strftime('%Y')`

Cross-field Validation

Consiste en usar varias columnas para comprobar la integridad de los datos → p.ej `col_1 + col_2 + col_3 = col_total`

Para ello podemos sumar a axis=1

CLEANING DATA

Datos NULL

Ver su distribución →

```
import missingno as msno
msno.matrix(df)
plt.show()
```

Comparando strings

Para calcular la distancia entre strings →

```
from thefuzz import process
matches = process.extract('...', df['col'], limit=df.shape[0])
```

Generando pares

Para generarlos por una columna →

```
import recordlinkage
indexer = recordlinkage.Index()
indexer.block('col')
pairs = indexer.index(df_a, df_b)
```

```
comp = recordlinkage.Compare()
comp.exact('col_df_a', 'col_df_b')
comp.string('...', '...', threshold=0.85)
potential_matches = comp.compute(pairs, df_a, df_b)
```

Para ver los matches →

```
matches = pot... [... .sum(axis=1) >= N]
dup = matches.index.get_level_values(1)
df_2 = df_2[~df_2.index.isin(dup)]
df = df_a.append(df_b)
```