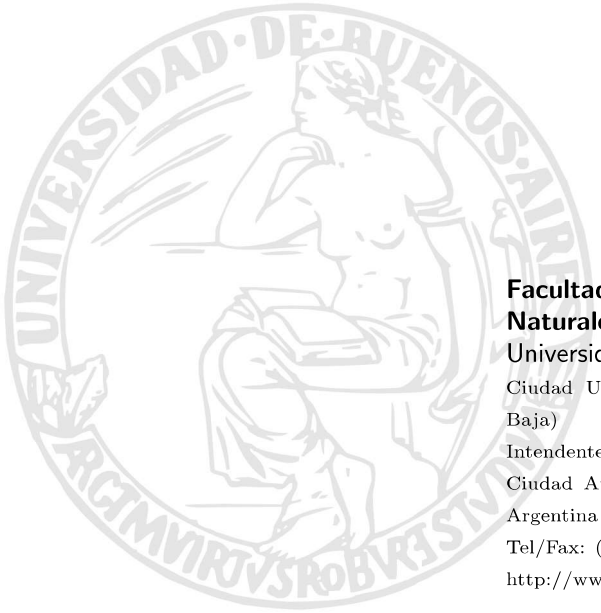
DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico 01

15 Mayo de 2022 Laboratorio de Datos


| Integrante | LU | Correo electrónico |
|---|---|---|
| Juan Pablo Aquilante | 755/18 | aquilantejp@outlook.es |
| Gastón Sanchez | 361/22 | gasanchez@dc.uba.ar |
| Mariano Papaleo | 848/21 | gagopoliscool@gmail.com |


**Facultad de Ciencias Exactas y Naturales**
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

# Resumen

Se nos proporcionó una fuente de datos públicos correspondientes al Padrón de Operadores Orgánicos Certificados de la República Argentina, y se desea saber si existe cierta relación entre el salario promedio y la actividad que se realiza en el sector privado, además de tener en cuenta si esta relación cambia dependiendo la provincia o el departamento en el que la actividad es realizada.
Se definieron dependencias funcionales para la obtención de un esquema de los datos, a partir de los esquemas se utilizó el DER como herramienta para construir un modelo conceptual de los datos.
A partir del DER se definió un MER en 3FN teniendo en cuenta PKs y FKs.
Uno de los mayores problemas del trabajo fue que cada fuente necesitaba un tratamiento en cuanto a calidad de datos bastante extenso. Pero a la hora de limpiar y corregir las fuentes de datos se tuvo en cuenta el MER, para que las tablas limpias quedarán en 3FN.
Una vez realizado el tratamiento se procedió a contestar preguntas en relación a los datos a través de consultas SQL.
Y por último antes de sacar conclusiones se hicieron gráficos por medio de herramientas de visualización.

# Introducción

Los datos del trabajo están dados dentro de 5 archivos csv.

**Padrón (DF1):** Padrón de operadores orgánicos certificados reconocidos por la Dirección de Agroalimentos. Se puede obtener de https://datos.magyp.gob.ar/dataset/padron-de-operadores-organicos-certificados

**Salarios (DF2):** Salarios asociados a cada actividad productiva, caracterizada por un atributo clae2 determinado por el INDEC, por cada departamento, año y mes. Se puede obtener de https://www.datos.gob.ar/fa_IR/dataset/produccion-salarios-por-departamentopartido-sector-actividad/archivo/produccion_515b41b2-d008-42fa-a9d7-8a1bb26d04ab

**Localidades censales (DF3):** Listado de las localidades censales según la base de datos censales del INDEC; contiene los nomencladores utilizados por AFIP para clasificar actividades con su correspondiente descripción. Se puede obtener de https://datos.gob.ar/ar/dataset/jgm-servicio-normalizacion-datos-geograficos/archivo/jgm_8.12

**Departamentos (DF4):** Diccionario de los códigos utilizados por INDEC para caracterizar los departamentos/partidos y provincias. Se puede obtener de https://datos.produccion.gob.ar/dataset/puestos-de-trabajo-por-departamento-partido-y-sector-de-actividad/archivo/125bdc76-0205-417a-bf20-76d34dbe184b

**Clases/clae2 (DF5):** Diccionario que relaciona las clae2 del INDEC a sus actividades productivas correspondientes. Se puede obtener de https://www.datos.gob.ar/fa_IR/dataset/produccion-salarios-por-departamentopartido-sector-actividad/archivo/produccion_8c7e4f21-750e-4298-93d1-55fe776ed6d4

# Decisiones tomadas

A continuación, listamos las decisiones que tomamos mientras realizamos el trabajo.

## Drops

- En **Padrón** dropeamos las columnas pais, pais_id y localidad dado que todos los operadores son de Argentina y que localidad es una columna llena de indefinidos que no utilizamos durante el trabajo
- En **Salarios** se estudió la cantidad de NaNs y se descubrió que sola eran 9156 filas del DataFrame mientras que el mismo posee más de 3 millones de filas por lo que se dropearon esas filas con NaNs.
- En **Localidades Censales** se dropeo una única fila con índice 60 dado que posee poca información y NaNs en lugares importante como departamento _id y departamento
-

## Renames

- En cada DataFrame se renombraron las columnas que se referían:
  - Al nombre de los departamentos como **departamento**
  - Las IDs de los departamentos como **departamento_id**
  - El nombre de las provincias como **provincia**
  - Las IDs de las provincias como **provincia_id**

- Se renombraron las provincias que referían a Tierra del fuego como **TIERRA DEL FUEGO** y las que referían a CABA como **CIUDAD AUTONOMA BUENOS AIRES**.

- Debido a que la columna **funcion** en **Localidades Censales** estaba llena de NaNs se decidió renombrar los NaNs de la columna como **SIN FUNCIÓN**.

- En cuanto a los valores NaNs, sin definir, sin especificar o con algún valor que indique que están indefinidos, se decidió tomar “sin_definir” como valor estándar. Donde sin_definir="INDEFINIDO".

## Ortografía

- Se eliminaron los acentos de la mayoría de columnas que se utilizaron para poder relacionarlas más fácilmente utilizando la biblioteca **unidecode**
- Se le aplicó a muchas columnas con STRINGS la función **.str.upper()** para que los mismos queden en mayúsculas y puedan relacionarse con otros DataFrame a los cuales también se les aplicó este proceso.

## Adiciones

- Se agregó **Padrón** una nueva columna llamada **id_operador** la cual se creó en base al índice del DataFrame + 1 (Para que comiencen los id en 1, en vez de en 0).
  Esto se hizo para que cada Operador, es decir, cada fila del DataFrame tuviera un ID único que lo identifica y que implique los atributos que ese Operador posee.

# Procesamiento de datos

## Primera Forma Normal (1FN)

El DF1 no estaba en 1FN debido a que la columna productos contiene varios productos que produce/elabora/vende el operador correspondiente separados por comas. Para solventar esto se creó un df1_productos en el cual se incluyeron **id_operador** y **producto**.
A continuación se atomizó, por medio de funciones y código en Python, la columna productos. Luego de esto, la nueva tabla quedó con tuplas únicas compuestas por **id_operador** y **producto** que nos indica para cada operador con qué producto se relaciona, y si se relaciona con más de un producto tiene varias filas en donde su **id_operador** se repite para un producto distinto. No hay problema en que **id_operador** se repita, ya que el mismo es una FK (Foreign Key) en esa tabla, mientras que en su tabla principal es un id único.

Todos los demás dataframes están en 1FN.

## Dependencias Funcionales

**Dataframe1:**

id_operador -> certificadora_id
id_operador -> categoria_id
id_operador -> provincia_id
id_operador -> departamento
id_operador -> rubro
id_operador -> establecimiento
id_operador -> producto
provincia_id -> provincia
certificadora_id -> certificadora_deno
categoria_id -> categoria_desc

**Paso 1 (pasar a forma canónica):** ya están en forma canónica.
**Paso 2 (remover atributos extraños):** del lado izquierdo solo aparece 1 atributo en cada caso, entonces no hay atributos extraños.
**Paso 3 (remover redundancia por transitividad):** los únicos atributos “intermediarios” entre otros 2 atributos son provincia_id, certificadora_id y categoria_id; id no tiene DF con provincia, certificadora_deno o categoria_desc, entonces no hay DFs redundantes.

**Dataframe2:**

{fecha, codigo_departamento_indec,clae2} -> w_median
codigo_departamento_indec -> id_provincia_indec

**Paso 1 (pasar a forma canónica):** ya están en forma canónica.
**Paso 2 (remover atributos extraños):** no hay atributos extraños porque no se pueden derivar otras DFs que tengan un solo atributo izquierdo de la primer DF.
**Paso 3 (remover redundancia por transitividad):** no hay DFs redundantes posibles.

**Dataframe3:**

id -> nombre
~~id -> provincia_id~~ *(eliminada por paso 3)*
id -> departamento_id
id -> centroide_lon
id -> centroide_lat
id -> funcion
id -> categoria
provincia_id -> provincia_nombre
departamento_id -> departamento_nombre
departamento_id -> provincia_id

-Se omite la columna fuente porque es INDEC para todos los casos y por ende no aporta nada.
-Se omiten las dependencias funcionales que involucran a municipio y municipio_id, porque no nos son relevantes a la hora de responder las preguntas de este trabajo, y tampoco es un atributo que aparezca en otros dataframes. A pesar de esto, si se fueran a considerar esas columnas se pueden observar las siguientes DF:

id -> municipio_id
municipio_id -> departamento_id
municipio_id -> municipio_nombre

**Paso 1 (pasar a forma canónica):** ya están en forma canónica.
**Paso 2 (remover atributos extraños):** no hay atributos extraños porque el lado derecho solo tiene un atributo en todos los casos.
**Paso 3 (remover redundancia por transitividad):** el único atributo intermediario que induce redundancia es departamento_id, entre id y provincia_id, que es redundante con la DF id->provincia_id; se decide eliminar esta última (aparece tachada en las DFs de arriba).

**Dataframe4:**

codigo_departamento_indec -> nombre_departamento_indec
codigo_departamento_indec -> id_provincia_indec
id_provincia_indec -> nombre_provincia_indec

**Paso 1 (pasar a forma canónica):** ya están en forma canónica.

**Paso 2 (remover atributos extraños):** no hay atributos extraños porque el lado derecho solo tiene un atributo en todos los casos.
**Paso 3 (remover redundancia por transitividad):** no hay DFs redundantes posibles.

**Dataframe5:**

clae2 -> clae2_desc
clae2 -> letra
letra -> letra_desc

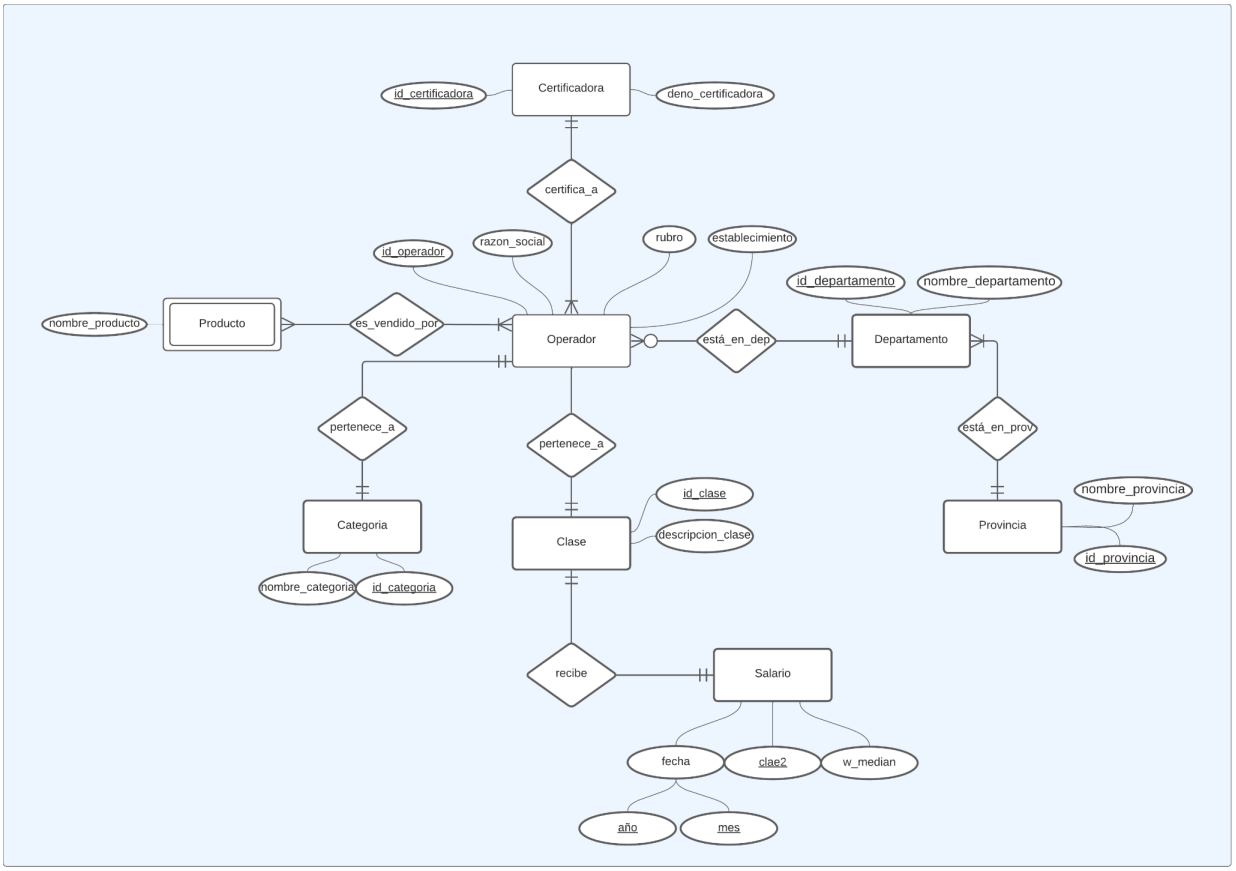
**Paso 1 (pasar a forma canónica):** ya están en forma canónica.
**Paso 2 (remover atributos extraños):** no hay atributos extraños porque el lado derecho solo tiene un atributo en todos los casos.
**Paso 3 (remover redundancia por transitividad):** no hay DFs redundantes posibles.

# Modelado de datos

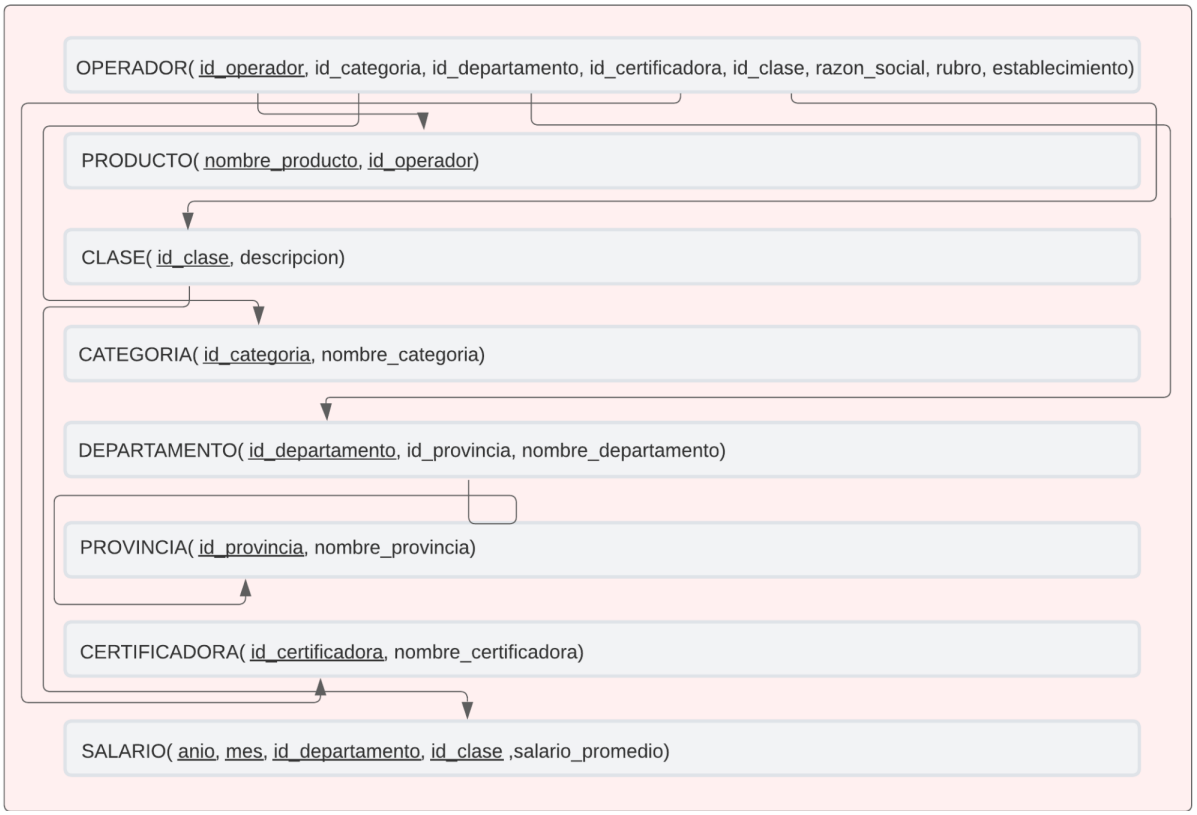
Una vez que se descompusieron los esquemas originales tal que estén en 3FN, se identificaron las siguientes entidades:

-Certificadora
-Operador
-Departamento
-Provincia
-Producto
-Categoría
-Clase
-Salario

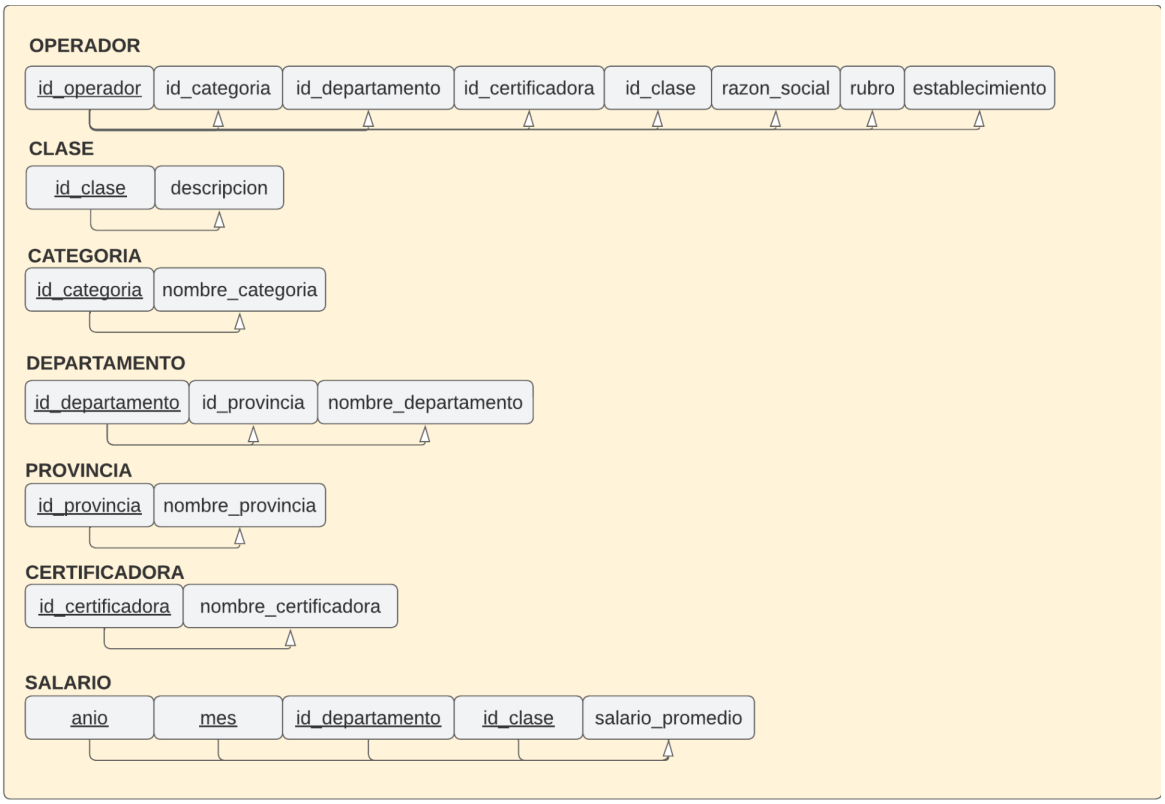
Se consideró agregar las entidades Municipio y Localidad, pero debido a que no son relevantes a la hora de responder las preguntas de este trabajo, se ignoraron. El DER resultante se puede observar en la figura inferior.




El modelo relacional derivado a partir del mismo se puede ver en la siguiente figura; las PK son los atributos subrayados, y las FK entre esquemas se relacionan con flechas. **Algo a tener en cuenta es que, en base a los atributos rubro y productos de un operador, a los mismos se les asignó manualmente una actividad productiva reconocida por el INDEC, a través del atributo clae2 del DF5, tal que se puedan relacionar las actividades productivas de cada operador con los salarios.**




Finalmente, las dependencias funcionales resultantes en este esquema se pueden observar en la siguiente figura; para facilitar la visualización, las DFs canónicas obtenidas que tienen un mismo atributo izquierdo se juntaron en una sola.



-En las DFs resultantes, por virtud de haber ignorado la localidad como entidad, se perdieron algunas DFs asociadas al DF3 (localidades censales).

-El atributo fecha se separó en atributos anio y mes (el día siempre era 01), lo que modificó las DFs del DF2 (salarios); no obstante, fecha era parte de la PK y los atributos resultantes también siguen siendo parte de la PK, así que a niveles prácticos no hubieron tantos cambios.

-Otros atributos que aparecían en varios dataframes fueron juntados en uno solo (codigo_departamento_indec del DF4, departamento_id del DF3 por ejemplo), entonces las DFs que previamente los involucraban desaparecieron; no obstante, siguen siendo representadas dentro de los nuevos esquemas mediante el atributo unificado correspondiente.

-En los esquemas resultantes, en todos los casos los atributos no primos dependen de manera total de la PK, y no hay dependencias transitivas, entonces los esquemas se encuentran en 3FN. También, las FK son PK de los esquemas que provienen, entonces se garantiza que no haya generación de tuplas espúreas.

-Nótese que no aparece el esquema de productos; debido a que es una entidad débil que sólo contiene el nombre de producto como atributo propio y una FK proveniente de operador, ambos de esos atributos forman la PK que identifica a un producto, y ningún atributo tiene una dependencia funcional del otro, por lo cual no tiene DFs asociadas.

# Limpieza/calidad de datos

## Departamentos (DF4)

De las primeras cosas que se hicieron fue crear un diccionario de departamentos con ayuda de **Departamentos**, ya que era necesario para tener vinculados cada departamento con un departamento_id **único**. Además, luego de haber experimentado con JOINS entre **Departamentos** y **Padrón** on = "departamento" (los nombres de cada departamento), se descubrieron 2 cosas:

1. Algunos departamentos en **Padrón** no son nombres de departamentos sino de localidades.
2. Algunos departamentos en **Padrón** no existen en el diccionario de departamento.

Por (2) se vio necesario crear IDs nuevos de departamento_id y que cada uno tuviera en su columna departamento = "departamento_desconocido" representando así los departamentos que no tienen ID para cada provincia que existe, es decir, se crearon 24 nuevas IDs con los números 1002-1096 en base a las IDs de provincia_id que van del 02 hasta el 96. Estás IDs se agregaron a el diccionario de departamentos

Por (1) se vio necesario vincular **Padrón** con **Localidades Censales** para encontrar aún más departamentos_id dado que en **Localidades Censales** hay nombres de localidades

asociadas a IDs de departamentos, y **Padrón** posee algunos nombres de localidades en la columna departamento erróneamente.

## Localidades censadas (DF3)

Luego de analizar el DF 4 nos damos cuenta que podemos obtener id de departamentos para **Padrón** de **Localidades Censales**.
Por lo tanto se decidió hacer varios LEFT JOIN (con la función pd.merge).

- Primero, se hizo merge de las columnas "departamento_id", "provincia_id", "departamento" de **Localidades Censales,** ON= "departamento", "provincia_id" de **Padrón**.

- Segundo, se hizo otro merge con las columnas "departamento_id", "provincia_id" y "nombre" (nombre de localidad) de **Localidades Censales,** ON= "provincia_id", "nombre" de **Padrón** ( este nombre es df1[“nombre”] = df1[“departamento”], es decir, es una copia de la columna departamentos para poder obtener IDS, luego se elimina).

- Por último se hizo un merge de las columnas "departamento_id", "provincia_id", "departamento" de **Departamentos,** ON= "departamento", "provincia_id" de **Padrón.**

Cada uno de estos merge genera una columna de departamento_id y luego con la función **combine.first** podemos tomar el primer departamento_id y complementar los NaNs que tenga con el segundo departamento_id y con otro combine.first lo mismo con el tercer departamento_id.
Gracias a este proceso se agregaron la mayoría de los departamento_id correspondientes a cada operador en **Padrón**, columna la cual original ni existía y ahora solo 124 operadores de **Padrón** no tienen un id para su departamento cuando esta fuente de datos tiene más de 1300 de operadores.
Lo último que falta es que para cada **departamento** al cual no le encontramos ID en ninguno de los tres JOINs que realizamos, asignarle la ID de “departamento_desconocido” que creamos en la columna departamento_id teniendo en cuenta a qué provincia pertenece cada operador.
Luego de esto, cada operador tiene asociado un departamento_id.

## Padrón (DF1)

En **Padrón** se logró agregar la columna **departamento_id** y se garantizó que cada operador posea una. Además de que, cada operador está asociado a un único **id_operador** que lo identifica y distingue de los demás operadores. Esa misma ID es la que se utiliza para las dependencias funcionales y en general cumple un rol principal como PK de la fuente de datos.

## Clases (DF5)

En **Clases** se agregó la letra Z para que acompañe cuando la clase es 999, es decir, que la clae

# Análisis de datos

Primero, se calculó la cantidad de operadores por provincia; en el esquema resultante, se observó que, más allá de que los operadores están distribuidos mayormente entre Buenos Aires, Córdoba y Mendoza, **no hay ninguna provincia que no tenga operadores certificados**. Se puede visualizar la distribución de los operadores por provincia en el siguiente histograma.

Luego, se calculó la cantidad de operadores por departamento, y se encontraron 307 departamentos sin operadores