DOCUMENTACIÓN CUBOS DE DATOS

Descripción de limpieza y transformaciones de datos

Versión 2

(Julio 2021)

SE Secretaría de Economía | Data México

Datawheel

TABLA DE CONTENIDO

1. Censo Económico

1.1. inegi_economic_census

Descripción general y ejecución

El pipeline del censo económico (*pipeline_economic_census*) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del censo económico entregados por INEGI (Instituto Nacional de Estadística y Geografía). Estos datos incluyen cantidades asociadas a las actividades económicas, tales como; unidades económicas (UE), personal, inversión total, entre otros. Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de dos formas:

(Python) `~/data-etl/etl/inegi_economic_census/$ python pipeline_economic_census.py`

(bamboo-cli) `~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry python pipeline_economic_census`

Descarga de datos

Una vez obtenidos los datos desde el sitio de INEGI y almacenados en un compartimiento privado de GCP storage, se establece una conexión privada validada con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Lectura de datos

Una vez descargados los datos, estos se leen y almacenan en un *DataFrame* de la librería pandas. Al momento en el que se elaboró esta documentación, el *DataFrame* inicial contenía 437.990 filas y 192 columnas. La siguiente figura muestra las primeras filas y algunas columnas del *DataFrame* inicial obtenido.

	Año Censal	Entidad	Municipio	Actividad Económica	UE Unidades económicas	...	P030C Variaciï¿½n de inventarios de productos en proceso (millones de pesos)1	Q000B Depreciaciï¿½n total de activos fijos (millones de pesos)1	Q010A Acervo total de maquinaria y equipo de produccïï¿½n (millones de pesos)	Q400A Acervo total de equipo de cï¿½mputo y perifï¿½ricos (millones de pesos)
437985	2004.0	08 Chihuahua	045 Meoqui	311520 Elaboración de helados y paletas	5.0	...	NaN	NaN	NaN	NaN
437986	2004.0	08 Chihuahua	048 Namiquipa	3115CC Clases agrupadas por el principio de c...	3.0	...	NaN	NaN	NaN	NaN
437987	2004.0	08 Chihuahua	050 Nuevo Casas Grandes	3115CC Clases agrupadas por el principio de c...	11.0	...	NaN	NaN	NaN	NaN

Figura X. Formato original datos del Censo Económico

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se eliminan del conjunto de datos las filas y columnas que corresponden a valores totales y las que se encuentran vacías. Cabe destacar que los valores totales no son necesarios porque pueden ser calculados agregando los valores individuales. Luego, se renombran las columnas que tienen código de medida, dejando sólo el código de medida como nombre, por ejemplo; la columna "Q000A Acervo total de activos fijos (millones de pesos)" pasa a ser "Q000A".

Posteriormente, se crean números identificadores (ID) para cada entidad federativa y municipio y se extrae el ID de cada actividad económica para luego incluirlos en el *DataFrame* como un identificador de cada uno de ellos, además, se guardan en tablas diferentes en la base de datos cada columna ID generada junto con su columna original, por ejemplo, Municipio con *mun_id* irán en una tabla. Finalmente, se eliminan del *DataFrame* las columnas duplicadas y las columnas que ya no son necesarias, por ejemplo; Municipio ya no es necesaria, porque ahora existe *mun_id* que corresponde al ID de cada municipio, el cual está adecuadamente indexado y guardado en otra tabla de dimensiones de la base de datos.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 426.563 filas y 101 columnas. La estructura final de la tabla se presenta a continuación:

	year	mun_id	national_industry_id	ue	h001a	h000a	h010a	...	p030a	p030b	q010a	q020a	q030a	q400a	q900a
426558	2004	8045	311520	5.0	13	13.0	1	...	0.000	0.000	0.126	1.080	0.060	0.000	0.157
426559	2004	8048	3115CC	3.0	89	83.0	80	...	0.006	0.003	7.231	2.503	2.654	0.397	0.356
426560	2004	8050	3115CC	11.0	28	28.0	6	...	0.006	0.000	0.847	1.200	0.285	0.000	0.101
426561	2004	8052	311520	5.0	29	28.0	20	...	0.000	0.000	0.515	2.750	0.200	0.020	0.042
426562	2004	8060	311520	3.0	9	9.0	0	...	0.000	0.000	0.110	0.080	0.000	0.000	0.026

Figura X. Formato final tabla de datos del Censo Económico

1.2. Inegi_economic_census_additional

Descripción general y ejecución

El pipeline *economic_census_2019_additional.py* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con el Censo Económico de 2019. En específico, contiene información adicional sobre indicadores a nivel de entidad y municipio de distintas actividades económicas. Los datos son descargados desde el sitio del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para ejecutar su posterior procesamiento.

El procesamiento consiste en un proceso de limpieza y transformación, donde luego son ingestados en una base de datos de Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de dos formas:

(Python) `~/data-etl/etl/inegi_economic_census/$ python economic_census_2019_additional.py`

(bamboo-cli) `~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry economic_census_2019_additional`

Descarga de datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv, y son 32 los archivos que se descargan para obtener la totalidad de los datos.

Una vez descargados los archivos a la carpeta temporal, mediante un *for loop*, se procede a leer con la función *read_csv* de Pandas cada uno de los archivos. Posterior a haber renombrado las columnas, y eliminada la variable ENTIDAD, se concatena cada archivo a un *DataFrame* llamado t.

Luego de haber concatenado los 32 archivos, se crea el *DataFrame* df como una copia de t.

Finalmente, se crea la columna *year*, la cual almacena el periodo en estudio. En este caso, corresponde a 2019.

Transformación - Entidades por Sector

Esta etapa procede como sigue:

- Se crea una copia del conjunto de datos total en un *DataFrame* llamado *base*.
- Selección de filas donde la variable *CODIGO* tenga largo igual a 2, sea igual a 31-33, o sea igual a 48-49.
- Selección de datos donde las variables *ID_ESTRATO*, *MUNICIPIO*, y *CODIGO*, no posean valores *NaN*.
- Creación de la variable *sector_id*, al eliminar espacios vacíos de la variable *CODIGO*.
- Creación de la variable *ent_id*, cuyos valores corresponden a los de la variable *ENTIDAD* con *dtype Integer*.
- Se eliminan variables que no serán utilizadas: *ID_ESTRATO, CODIGO, MUNICIPIO, y ENTIDAD*.
- Se crean variables que no existan en el conjunto de datos con el valor 0. Entre las variables que deben existir, se encuentran: *ent_id, mun_id, sector_id, subsector_id, y rama_id*.

- Creación de la variable *level*. Se asigna el valor único de 1.

Posterior a ello, se crea una copia del *DataFrame* procesado, llamándolo *df_ent_sec*. Dicho *DataFrame* junto al *DataFrame* base, continúan a la siguiente etapa de transformación.

Transformación - Entidades por Sub Sector

En esta etapa se ejecutan las siguientes acciones:

- Se crea un nuevo *DataFrame* llamado df. Dicho df es simplemente una copia de *base*.
- Selección de filas donde la variable *CODIGO* tenga longitud igual a 3.
- Selección de datos donde las variables *ID_ESTRATO*, *MUNICIPIO*, y *CODIGO*, no posean valores *NaN*.
- Creación de la variable *subsector_id*, cuyos valores corresponden a los de la variable *CODIGO* con *dtype Integer*.
- Creación de la variable *ent_id*, cuyos valores corresponden a los de la variable ENTIDAD con *dtype Integer*.
- Se eliminan variables que no serán utilizadas: *ID_ESTRATO*, *CODIGO*, *MUNICIPIO* y *ENTIDAD*.
- Se crean variables que no existan en el conjunto de datos con el valor 0. Entre las variables que deben existir, se encuentran: *ent_id, mun_id, sector_id, subsector_id,* y *rama_id*.
- Creación de la variable *level*. Se asigna el valor único de 2.

Posterior a ello, se crea una copia del *DataFrame* procesado, llamándolo *df_ent_sub*. Dicho *DataFrame* junto al *DataFrame* base y *df_ent_sec*, continúan a la siguiente etapa de transformación.

Transformación - Entidades por Rama

Esta etapa procede como sigue:

- Se crea un nuevo *DataFrame* llamado df. Dicho *DataFrame* es simplemente una copia de *base*.

- Selección de filas donde la variable *CODIGO* tenga longitud igual a 4.
- Selección de datos donde las variables *ID_ESTRATO, MUNICIPIO* y *CODIGO*, no posean valores *NaN*.
- Creación de la variable *rama_id*, cuyos valores corresponden a los de la variable *CODIGO* con *dtype Integer*.
- Creación de la variable *ent_id*, cuyos valores corresponden a los de la variable *ENTIDAD* con *dtype Integer*.
- Se eliminan variables que no serán utilizadas: *ID_ESTRATO, CODIGO, MUNICIPIO* y *ENTIDAD*.
- Se crean variables que no existan en el conjunto de datos con el valor 0. Entre las variables que deben existir, se encuentran: *ent_id, mun_id, sector_id, subsector_id* y *rama_id*.
- Creación de la variable level. Se asigna el valor único de 3.

Posterior a ello, se crea una copia del *DataFrame* procesado, llamándolo *df_ent_ram*. Dicho *DataFrame* junto al *DataFrame base*, *df_ent_sec*, y *df_ent_sub*, continúan a la siguiente etapa de transformación.

Transformación - Municipio por Sector

En esta etapa se ejecutan las siguientes acciones:

- Se crea un nuevo *DataFrame* llamado df. Dicho df es simplemente una copia de *base*.
- Selección de filas donde la variable *CODIGO* tenga largo igual a 2, sea igual a 31-33, o sea igual a 48-49.
- Selección de datos donde las variables *ID_ESTRATO, MUNICIPIO* y *CODIGO*, no posean valores *NaN*.
- Creación de la variable *sector_id*, al eliminar espacios vacíos de la variable *CODIGO*.
- Creación de la variable *mun_id*, cuyos valores corresponden a la agregación de las variables *ENTIDAD* y *MUNICIPIO*, transformados a *dtype Integer*.

- Se eliminan variables que no serán utilizadas: *ID_ESTRATO, CODIGO, MUNICIPIO* y *ENTIDAD*.
- Se crean variables que no existan en el conjunto de datos con el valor 0. Entre las variables que deben existir, se encuentran: *ent_id, mun_id, sector_id, subsector_id* y *rama_id.*
- Creación de la variable *level*. Se asigna el valor único de 4.

Posterior a ello, se crea una copia del *DataFrame* procesado, llamándolo *df_mun_sec*. Dicho *DataFrame* junto al *DataFrame base*, *df_ent_sec, df_ent_sub*, y *df_ent_ram*, continúan a la siguiente etapa de transformación.

Concatenación

En esta etapa, ingresan los conjuntos de datos: *base, df_ent_sec, df_ent_sub, df_ent_ram* y *df_mun_sec*. Dado su estructura similar, la idea es agrupar todos los conjuntos de datos en uno solo. Dado ello, se llevan a cabo las siguientes etapas:

- Mediante un *for loop* y la función *append* de Pandas, se concatenan todos los conjuntos de datos en uno llamado *df*.
- Transformación de *dtypes* a conveniencia.
- Creación de la variable *year*. Dicha variable almacena el periodo en estudio.
- Transformación de nombres de columnas a minúsculas. Se utiliza la función *str.lower()* de Pandas.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 42.507 filas y 189 columnas. A continuación puede ser visualizada su estructura de salida:

	ue	a111a	a121a	a131a	a211a	...	ent_id	mun_id	subsector_id	rama_id	level
5	37	188.191	150.755	37.436	-2.869	...	1.0	0.0	0.0	0.0	1
39	19	8651.804	3349.414	5302.390	322.953	...	1.0	0.0	0.0	0.0	1
78	13	NaN	NaN	NaN	NaN	...	1.0	0.0	0.0	0.0	1
96	408	7843.008	5432.334	2410.674	71.039	...	1.0	0.0	0.0	0.0	1
244	5588	326901.167	228336.310	98564.857	5802.321	...	1.0	0.0	0.0	0.0	1

Figura X. Formato final tabla de datos del Censo Económico 2019

1.3. Inegi_economic_census_sex

Descripción general y ejecución

El pipeline del censo económico por sexo (*economic_census_sex_pipeline*) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del censo económico entregados por INEGI (Instituto Nacional de Estadística y Geografía) desagregando los datos según el sexo de los trabajadores. Incluye características asociadas al personal en las diferentes industrias.

Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos *Clickhouse* siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales la librería pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de dos formas:

(Python) `~/data-etl/etl/inegi_economic_census/$ python economic_census_sex_pipeline.py`

(bamboo-cli) `~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry python economic_census_sex_pipeline`

Descarga de datos

Una vez obtenidos los datos desde el sitio de INEGI y almacenados en un compartimiento privado de GCP storage, se establece una conexión privada validada

con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Lectura de datos

Una vez descargados los datos, estos se leen y almacenan en un *DataFrame* de la librería pandas. Al momento en el que se elaboró esta documentación, el *DataFrame* inicial contenía 437.990 filas y 192 columnas. La siguiente figura muestra las primeras filas y algunas columnas del *DataFrame* inicial obtenido.

	Año Censal	Entidad	Municipio	Actividad Económica	UE Unidades económicas	...	P030C Variaciï¿½n de inventarios de productos en proceso (millones de pesos)1	Q000B Depreciaciï¿½n total de activos fijos (millones de pesos)1	Q010A Acervo total de maquinaria y equipo de producciï¿½n (millones de pesos)	Q400A Acervo total de equipo de cï¿½mputo y perifï¿½ricos (millones de pesos)
437985	2004.0	08 Chihuahua	045 Meoqui	311520 Elaboración de helados y paletas	5.0	...	NaN	NaN	NaN	NaN
437986	2004.0	08 Chihuahua	048 Namiquipa	3115CC Clases agrupadas por el principio de c...	3.0	...	NaN	NaN	NaN	NaN
437987	2004.0	08 Chihuahua	050 Nuevo Casas Grandes	3115CC Clases agrupadas por el principio de c...	11.0	...	NaN	NaN	NaN	NaN

Figura X. Formato original datos del Censo Económico por sexo

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se eliminan del *DataFrame* las filas y columnas que no serán utilizadas, estas corresponden a valores agregados, valores totales, filas vacías y filas y columnas repetidas. Cabe destacar que los valores totales no son necesarios porque pueden ser calculados agregando los valores individuales.

Posteriormente, se crean números identificadores (ID) para cada entidad y municipio y se extrae el ID de cada actividad económica para luego incluirlos en el *DataFrame* como un identificador de cada uno de ellos, además, se guardan en tablas diferentes en la base de datos cada columna ID generada junto con su columna original, por ejemplo, Municipio con *mun_id* irán en una tabla.

Luego, se divide el *DataFrame* en dos, uno para cada sexo, a cada uno de ellos se les renombran sus columnas, pasando de códigos a nombres más específicos, por ejemplo; *"H101B: production_personnel_sales_and_service"*, pasa a ser *"production_personnel_sales_and_service"*. Además a cada *DataFrame* se le agrega una columna identificadora del sexo, donde hombres corresponde a 1 y mujeres a 2. Finalmente se concatenan nuevamente los *DataFrame*, esta vez ordenados por sexo.

Finalmente, se obtienen agregaciones de los datos agrupando el *DataFrame* por año, municipio, sexo e industria.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 853.126 filas y 13 columnas. La estructura final de la tabla se presenta a continuación:

year	mun_id	national_industry_id	sex	employed_workers	...	administrative_accounting_and_managerial_staff	owners_family_and_other_unpaid_workers
2004	1001	114119	1	18	...	0	7
2004	1001	114119	2	0	...	0	0
2004	1001	21CCCC	1	394	...	10	1
2004	1001	21CCCC	2	7	...	7	0
2004	1001	23611C	1	5332	...	301	32
...	...	...	...	...	...	...	...
2014	32058	8114CC	2	6	...	0	3
2014	32058	81CCCC	1	1	...	0	1
2014	32058	81CCCC	2	7	...	0	5
2014	32058	SCCCCC	1	6	...	0	1
2014	32058	SCCCCC	2	2	...	1	1

Figura X. Formato original datos del Censo Económico por sexo

1.4. Inegi_economic_census_2014_ent

Descripción general y ejecución

El pipeline del censo económico 2014 por entidad federativa (*economic_census_2014_ent*) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del censo económico 2014 a nivel estatal entregados por INEGI (Instituto Nacional de Estadística y Geografía). Estos datos incluyen métricas asociadas a las actividades económicas por entidad federativa, tales como unidades económicas, producción bruta total, depreciación total de activos fijos, salarios de los empleados, entre otros.

Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Al existir 32 entidades federativas, este pipeline realiza el proceso de ETL de manera iterativa sobre cada una de ellas. En otras palabras, los mismos cuatro pasos del pipeline se ejecutan para cada estado.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de dos formas:

(Python) `~/data-etl/etl/inegi_economic_census/$ python economic_census_2014_ent.py`

(bamboo-cli) `~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry economic_census_2014_ent`

Descarga de datos

Una vez obtenidos los datos desde el sitio de INEGI y almacenados en un compartimiento privado de GCP storage, se establece una conexión privada validada con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Cabe destacar que existe un archivo .csv por cada entidad federativa, por lo que al terminar de iterar sobre todas ellas, se habrán descargado un total de 32 archivos.

Lectura de datos

Una vez descargados los datos para la entidad federativa de turno, estos se leen y almacenan en un *DataFrame* de la librería pandas. A modo de ejemplo, para la entidad federativa 28 (Tamaulipas), al momento en el que se elaboró esta documentación, el *DataFrame* inicial tenía 36.667 filas y 105 columnas. La siguiente imagen muestra las primeras filas y algunas columnas del *DataFrame* inicial obtenido.

	clave_entidad	entidad_federativa	clave_municipio	municipio	clave_actividad_economica	...	Q010A	Q020A	Q030A	Q400A	Q900A
36662	28	Tamaulipas	43.0	Xicoténcatl	SCCC	...	99.279	2.8	0.938	0.168	0.37
36663	28	Tamaulipas	43.0	Xicoténcatl	SCCCC	...	99.279	2.8	0.938	0.168	0.37
36664	28	Tamaulipas	43.0	Xicoténcatl	SCCCC	...	99.279	2.8	0.938	0.168	0.37
36665	28	Tamaulipas	43.0	Xicoténcatl	SCCCCC	...	99.279	2.8	0.938	0.168	0.37
36666	28	Tamaulipas	43.0	Xicoténcatl	SCCCCC	...	99.279	2.8	0.938	0.168	0.37

Figura X. Formato original datos del Censo Económico 2014

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se filtran y reemplazan todos los valores nulos, dejando solo los valores no nulos dentro del *DataFrame*, además, se filtran las claves de actividad económica dejando solo las que contienen 6 dígitos.

Finalmente, se eliminan del *DataFrame* todas las columnas que ya no son necesarias, dado que mantenerlas solo genera redundancia en los datos. También se agrega una última columna correspondiente al año de realización del censo, que en este caso corresponde a 2014.

Al momento en el que se elaboró esta documentación, para la entidad federativa 28 (Tamaulipas), el *DataFrame* final contenía 1.960 filas y 101 columnas. La estructura final de la tabla se presenta a continuación:

	ent_id	national_industry_id	ue	a111a	a121a	a131a	a211a	...	q000d	q010a	q020a	q030a	q400a	q900a	year
4605	28	813230	25	6.978	4.372	2.606	0.633	...	0.0	0.666	2.649	1.581	0.312	0.672	2014
4606	28	813230	26	19.140	5.646	13.494	0.033	...	0.0	0.076	3.406	0.635	0.586	1.485	2014
4607	28	813230	105	39.247	17.349	21.898	1.245	...	0.0	9.581	83.068	1.994	0.503	1.389	2014
4616	28	SCCCCC	24	33615.827	2941.778	30674.049	3096.148	...	0.0	19049.194	3738.023	158.497	0.784	294.842	2014
4617	28	SCCCCC	24	33615.827	2941.778	30674.049	3096.148	...	0.0	19049.194	3738.023	158.497	0.784	294.842	2014

Figura X. Formato final tabla de datos del Censo Económico 2014

1.5 inegi_economic_census_2014_mun

Descripción general y ejecución

El pipeline del censo económico 2014 por municipio (*economic_census_2014_mun*) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del censo económico 2014 a nivel municipal entregados por INEGI (Instituto Nacional de Estadística y Geografía). Estos datos incluyen métricas asociadas a las actividades económicas por municipio, tales como unidades económicas, producción bruta total, depreciación total de activos fijos, salarios de los empleados, entre otros.

Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Este pipeline es casi idéntico al pipeline del censo económico 2014 por entidad federativa, la diferencia es que el presente pipeline procesa los datos a nivel municipal.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de dos formas:

(Python)

```
~/data-etl/etl/inegi_economic_census/$ python economic_census_2014_mun.py
```

(bamboo-cli)

```
~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry economic_census_2014_mun
```

Al existir 32 entidades federativas, este pipeline realiza el proceso de ETL de manera iterativa sobre cada una de ellas. En otras palabras, los mismos cuatro pasos del pipeline se ejecutan para cada estado. En este pipeline en específico, se desagregan los datos por cada municipio.

La razón para tener un pipeline para nivel estatal y otro a nivel municipal radica en la anonimización de los datos, donde el acumulado de los municipios no entrega el valor correcto a nivel estatal.

Descarga de datos

Una vez obtenidos los datos desde el sitio de INEGI y almacenados en un compartimiento privado de GCP storage, se establece una conexión privada validada con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Cabe destacar que existe un archivo .csv por cada entidad federativa, por lo que al terminar de iterar sobre todas ellas, se habrán descargado un total de 32 archivos.

Lectura de datos

Una vez descargados los datos para la entidad federativa de turno, estos se leen y almacenan en un *DataFrame* de la librería pandas. A modo de ejemplo, para la entidad 28 (Tamaulipas), al momento en el que se elaboró esta documentación, el *DataFrame* inicial tenía 36.667 filas y 105 columnas. La siguiente imagen muestra las primeras filas y algunas columnas del *DataFrame* inicial obtenido.

	clave_entidad	entidad_federativa	clave_municipio	municipio	clave_actividad_economica	...	Q010A	Q020A	Q030A	Q400A	Q900A
36662	28	Tamaulipas	43.0	Xicoténcatl	SCCC	...	99.279	2.8	0.938	0.168	0.37
36663	28	Tamaulipas	43.0	Xicoténcatl	SCCCC	...	99.279	2.8	0.938	0.168	0.37
36664	28	Tamaulipas	43.0	Xicoténcatl	SCCCC	...	99.279	2.8	0.938	0.168	0.37
36665	28	Tamaulipas	43.0	Xicoténcatl	SCCCCC	...	99.279	2.8	0.938	0.168	0.37
36666	28	Tamaulipas	43.0	Xicoténcatl	SCCCCC	...	99.279	2.8	0.938	0.168	0.37

Figura X. Formato original datos del Censo Económico 2014 a nivel municipal

Transformación y limpieza

Luego de la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se filtran y reemplazan todos los valores nulos, dejando solo los valores no nulos dentro del *DataFrame*, además, se filtran las claves de actividad económica dejando solo las que contienen 6 dígitos.

Posterior al filtrado, se genera un número identificador (ID) para cada municipio, el cual será el identificador del municipio dentro del cubo final. De forma paralela se guardará el ID del municipio junto al nombre del municipio en una tabla diferente en la base de datos.

Finalmente, se eliminan del *DataFrame* todas las columnas que ya no son necesarias, dado que mantenerlas solo genera redundancia en los datos. También se agrega una última columna correspondiente al año de realización del censo, que en este caso corresponde a 2014.

Al momento en el que se elaboró esta documentación, para la entidad federativa 28 (Tamaulipas), el *DataFrame* final contenía 7.124 filas y 101 columnas. La estructura final de la tabla se presenta a continuación:

	national_industry_id	ue	a111a	a121a	a131a	a211a	a221a	...	q010a	q020a	q030a	q400a	q900a	mun_id	year
36636	812110	7	0.546	0.226	0.320	0.002	0.002	...	0.152	0.295	0.045	0.000	0.168	28043	2014
36642	812CCC	3	0.474	0.262	0.212	0.017	0.017	...	0.414	0.800	0.030	0.000	0.049	28043	2014
36650	8131CC	6	20.968	9.665	11.303	1.190	1.190	...	27.000	1.890	5.165	0.226	0.049	28043	2014
36656	813230	3	0.013	0.018	-0.005	0.000	0.000	...	0.000	0.070	0.000	0.000	0.011	28043	2014
36666	SCCCCC	7	128.471	63.474	64.997	20.971	20.569	...	99.279	2.800	0.938	0.168	0.370	28043	2014

Figura X. Formato final tabla de datos del Censo Económico 2014 a nivel municipal

1.6. Indicators_economic_census

Descripción general y ejecución

El pipeline *indicators_economic_census_2019.py* es un compilado de diferentes indicadores de la industria mexicana provenientes del Censo Económico 2019.

Dentro de los indicadores incluidos se encuentran aquellos asociados al financiamiento, tecnologías de información, características del personal ocupado y manejo del negocio, compras y ventas por internet, medio ambiente y ciencia y tecnología.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

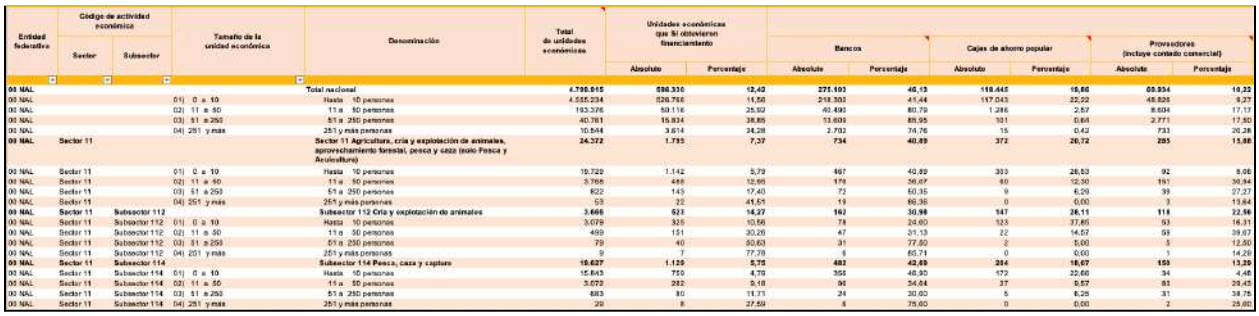
Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli) `~/data-etl/etl/inegi_economic_census/$ bamboo-cli --folder . --entry indicators_economic_census_2019`

Lectura de datos

El pipeline utiliza 29 conjuntos de datos extraídos de los tabulados del Censo Económico 2019. Cada archivo tiene su estructura propia dependiendo del indicador en cuestión y, por ende, también se realiza un procesamiento diferente a cada archivo de datos para finalizar concatenándolos en una única tabla de datos.

A modo de ejemplo, a continuación se visualiza un extracto de la estructura inicial del conjunto de datos que contiene el indicador asociado a fuentes de financiamiento por tamaño de la unidad económica. La estructura inicial se encuentra en formato .xlsx, posee varias filas vacías y otras combinadas, pero en términos generales cuenta con 14.386 filas y 24 columnas:



Entidad federativa	Código de actividad económica		Tamaño de la unidad económica	Denominación	Total de unidades económicas	Unidades económicas que SI obtuvieron financiamiento		Bancos		Cajas de ahorro popular		Proveedores (incluye comercio comercial)	
	Sector	Subsector				Absoluto	Porcentaje	Absoluto	Porcentaje	Absoluto	Porcentaje	Absoluto	Porcentaje
00 NAL				Total nacional	4.798.845	596.330	12,42	275.103	46,13	118.445	19,86	69.934	10,23
00 NAL			01) 0 a 10	Hasta 10 personas	4.555.234	526.766	11,56	218.303	41,44	117.043	22,22	46.826	8,27
00 NAL			02) 11 a 50	11 a 50 personas	193.376	50.116	25,92	40.493	80,79	1.286	2,57	8.604	17,17
00 NAL			03) 51 a 250	51 a 250 personas	40.781	15.834	38,85	13.606	85,95	101	0,64	2.771	17,50
00 NAL			04) 251 y más	251 y más personas	10.544	3.614	34,28	2.702	74,76	15	0,42	733	20,28
00 NAL	Sector 11			Sector 11 Agricultura, cría y explotación de animales, aprovechamiento forestal, pesca y caza (solo Pesca y Acuicultura)	24.372	1.795	7,37	734	40,89	372	20,72	285	15,88
00 NAL	Sector 11		01) 0 a 10	Hasta 10 personas	19.722	1.142	5,79	467	40,89	393	26,53	92	8,06
00 NAL	Sector 11		02) 11 a 50	11 a 50 personas	3.766	466	12,96	179	38,87	60	12,90	191	39,94
00 NAL	Sector 11		03) 51 a 250	51 a 250 personas	822	143	17,40	72	50,35	9	6,29	39	27,27
00 NAL	Sector 11		04) 251 y más	251 y más personas	53	22	41,51	19	86,36	0	0,00	3	13,64
00 NAL	Sector 11	Subsector 112		Subsector 112 Cría y explotación de animales	3.666	623	14,27	162	30,98	147	28,11	118	22,56
00 NAL	Sector 11	Subsector 112	01) 0 a 10	Hasta 10 personas	3.079	326	10,56	78	24,60	123	37,85	53	16,31
00 NAL	Sector 11	Subsector 112	02) 11 a 50	11 a 50 personas	499	151	30,26	47	31,13	22	14,57	59	39,07
00 NAL	Sector 11	Subsector 112	03) 51 a 250	51 a 250 personas	79	40	50,63	31	77,50	2	5,00	5	12,50
00 NAL	Sector 11	Subsector 112	04) 251 y más	251 y más personas	9	7	77,78	6	85,71	0	0,00	1	14,29
00 NAL	Sector 11	Subsector 114		Subsector 114 Pesca, caza y captura	19.627	1.129	5,75	483	42,69	284	16,07	153	13,29
00 NAL	Sector 11	Subsector 114	01) 0 a 10	Hasta 10 personas	15.843	759	4,79	355	46,90	172	22,66	34	4,48
00 NAL	Sector 11	Subsector 114	02) 11 a 50	11 a 50 personas	3.072	282	9,18	96	34,04	37	9,57	83	29,43
00 NAL	Sector 11	Subsector 114	03) 51 a 250	51 a 250 personas	683	80	11,71	24	30,00	5	6,25	31	38,75
00 NAL	Sector 11	Subsector 114	04) 251 y más	251 y más personas	29	8	27,59	6	75,00	0	0,00	2	25,00

Figura X. Formato original indicadores del Censo Económico 2019 - Financiamiento

Es importante notar que cada archivo posee diferentes cantidades de columnas y filas, por ello, el pipeline es extenso dado que debe hacer un pre-procesamiento de cada archivo en forma separada.

Limpieza y Transformación

El proceso de limpieza y transformación inicia con la lectura de los 29 archivos de datos y otros dos archivos de dimensiones que contienen el nombre de los indicadores y categorías en español e inglés junto a sus *id's*.

Las primeras líneas crean los diccionarios para categorías e indicadores usando los 2 archivos leídos previamente. Estos diccionarios serán utilizados al finalizar el proceso para estandarizar los datos.

Luego se procesan los archivos de datos agrupándolos por indicadores:

Indicadores de financiamiento

Financiamiento por tamaño y edad de las unidades económicas:

- Lectura de los archivos *crednce19_03*, *crednce19_04*, *crednce19_05*, *crednce19_06*.

- Definición de variables que permiten procesar cada archivo según características propias de su estructura.
- Eliminación de filas que contienen las palabras *"Todos"*, *"Sector"*, *"Subsector"*. Esto con el objetivo de eliminar filas con totales que son redundantes cuando se trabaja en formato *tidy*.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de las variables de interés: *indicator*, *value_no_financing*, *pct_no_financing*
- Creación de dos *DataFrames*: *df_crednce19_03_B1* y *df_crednce19_03_B2* con las columnas de interés.

Fuentes de financiamiento:

- Aplicación de la función *categories_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline. En este caso, se aplica al archivo *crednce19_03* que detalla las fuentes de financiamiento utilizadas por las unidades económicas

Se concatenan los *DataFrames* que ya tienen la estructura deseada generando el *DataFrame* *df_financing1*.

Acceso a financiamiento, cuenta bancaria y crédito bancario a nivel nacional y estatal:

- Lectura de los archivos *crednce19_03, crednce19_07, crednce19_08*.
- Selección de columnas de interés.
- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Unión de los archivos con la función *append* de pandas para obtener el *DataFrame* *df_financing2*.

Razones para no acceder a crédito o cuenta bancaria:

- Lectura de los archivos *crednce19_07* y *crednce19_08*.
- Definición del nombre de columnas de interés.
- Aplicación de las funciones *general_format*, *yes_no_format* y *categories_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Unión de los archivos con la función *append* de pandas para obtener el *DataFrame* *df_financing3*.

Finalmente se concatenan los *DataFrames* *df_financing1*, *df_financing2* y *df_financing3* para obtener una única tabla con los indicadores de financiamiento.

Indicadores de acceso a internet

Acceso a internet según cantidad de unidades económicas:

- Lectura de los archivos *ecomnce19_05* y *ecomnce19_06*.
- Definición de variables que permiten procesar cada archivo según características propias de su estructura.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Selección de columnas de interés.
- Aplicación de la función *melt* de pandas para lograr una tabla en formato *tidy*.

Acceso a internet según porcentaje de unidades económicas:

- Lectura de los archivos *ecomnce19_03* y *ecomnce19_03*.
- Se aplican los mismos pasos anteriores para conseguir una tabla con los valores porcentuales

Se concatenan los *DataFrames* que contienen valores y porcentajes generando *df_internet1*.

Acceso a internet según nivel geográfico:

- Lectura de los archivos *ecomnce19_05* y *ecomnce19_06*.
- Definición de variables que permiten procesar cada archivo según características propias de su estructura.
- Selección de columnas de interés.
- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna indicator.
- Unión de los archivos con la función *append* de pandas para obtener el *DataFrame df_internet2*.

Finalmente se concatenan los *DataFrames df_internet1* y *df_internet2* para obtener una única tabla con los indicadores de internet.

Indicadores de manejo del negocio

Problemáticas que enfrentan las unidades económicas por sector económico:

- Lectura del archivo *probnce19_03*.
- Definición del nombre de las columnas.
- Aplicación de las funciones *general_format* y *categories_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.

Problemáticas que enfrentan las unidades económicas por nivel geográfico:

- Eliminación de filas innecesarias del archivo *probnce19_03* y renombre de columnas.
- Selección de filas que contienen la palabra "personas" en la columna *category* y que contienen nulos en la columna *sector_id*.
- Reemplazo de valores nulos por ceros.
- Creación de las columnas *ent_id* y *nation_id*.

- Aplicación de la función *categories_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.

Se concatenan las tablas generadas para las problemáticas a nivel geográfico y sector económico en el *DataFrame* *df_business1*.

Cuentan con sistema contable:

- Lectura del archivo *contnce19_02*.
- Definición del nombre de las columnas.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *subsector_id*.
- Aplicación de las funciones *yes_no_format* y *categories_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Creación de un nuevo *DataFrame* a partir del archivo *contnce19_02* que contendrá los valores por nivel geográfico.
- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Concatenación de archivos para conseguir la tabla *df_business2*.

Transacciones monetarias:

- Lectura del archivo **pagonce19_02**.
- Definición del nombre de las columnas.
- Aplicación de las funciones *general_format* y *categories_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Creación de un nuevo *DataFrame* a partir del archivo *pagonce19_02* que contendrá los valores por nivel geográfico.

- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Aplicación de la función *melt* de pandas para lograr una tabla en formato *tidy*.
- Creación de las columnas *percentage* e *indicator*.
- Concatenación de archivos para conseguir la tabla *df_business3*.

Finalmente se concatenan los *DataFrames* *df_business1*, *df_business2* y *df_business3* para obtener una única tabla con los indicadores de manejo del negocio.

Indicadores de personal

Rotación y permanencia del personal:

- Lectura del archivo *capance19_02*.
- Definición del nombre de las columnas.
- Aplicación de las funciones *general_format* y *yes_no_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Con esto se logra el *DataFrame* *df_staff1*.

Capacitación del personal:

- Lectura del archivo *edadnce19_02*.
- Definición del nombre de las columnas.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Selección de variables de interés.
- Creación de la columna *category* que utiliza un diccionario creado previamente.
- Creación de dos *DataFrames*, uno contiene los datos de capacitación según edad del personal y el otro según nivel de estudios del personal.

- En cada uno de los *DataFrames* se renombra la columna *category* por *indicator*, se aplica la función *melt* de pandas para conseguir una tabla en formato *tidy* y se crean las columnas *percentage* e *indicator*.
- Concatenación de los *DataFrames* para conseguir la tabla *df_staff2*.

Rotación y permanencia del según edad y nivel educacional:

- Lectura del archivo *edadnce19_03*.
- Definición del nombre de las columnas.
- Aplicación de las funciones *general_format* y *yes_no_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Mediante un ciclo *for* se trabajan en forma separada los *DataFrames* con edad y educación y luego se unen con la función *append* de pandas. En el ciclo se seleccionan las columnas de interés, se aplica la función *melt* de pandas para conseguir una tabla en formato *tidy*, se crean las columnas *percentage* e *indicator*, se renombran las columnas *category* por *indicator* y *category_temp* por *category* y se elimina la columna *total_staff*. Al finalizar el ciclo se obtiene el *DataFrame* *df_staff3*.

Finalmente se concatenan los *DataFrames* *df_staff1*, *df_staff2* y *df_staff3* para obtener una única tabla con los indicadores de personal.

Indicadores de tecnologías de información

Servicios de cómputo e internet:

- Lectura de los archivos *ticsnce19_01* y *ticsnce19_02*.
- Definición de variables que permiten procesar cada archivo según características propias de su estructura y definición del nombre de columnas.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.

- Aplicación de la función *yes_no_format* dos veces consecutivas para generar dos *DataFrames*, uno asociado a servicios de internet y otro asociado a servicios de cómputo.
- Creación de la columna *indicator*.
- Concatenación de los *DataFrames* para conseguir la tabla *df_tic1*.

Usos de internet:

- Lectura de los archivos *ticsnce19_03* y *ticsnce19_04*.
- Definición de variables que permiten procesar cada archivo según características propias de su estructura y definición del nombre de columnas.
- Aplicación de las funciones *general_format* y *categories_format* alojadas en la carpeta *util* y desarrolladas para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Concatenación de los *DataFrames* para conseguir la tabla *df_tic2*.

Usos de internet por nivel geográfico:

- Lectura del archivo *ticsnce19_03*.
- Selección de columnas de interés.
- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *indicator*.
- Con esto se logra el *DataFrame df_tic3*.

Finalmente se concatenan los *DataFrames df_tic1*, *df_tic2* y *df_tic3* para obtener una única tabla con los indicadores de tecnologías de información.

Indicadores de innovación

Innovación por nivel geográfico:

- Lectura del archivo *innonce19_01*.
- Definición de columnas de interés.
- Aplicación de la función *geo_data* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Renombre de columnas.
- Creación de las columnas *percentage* e *indicator*.
- Eliminación de la columna *total_ue*.
- Con esto se logra el *DataFrame df_innovation1*.

Actividades de innovación en 2018:

- Lectura del archivo *innonce19_04*.
- Definición de columnas de interés.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *category*.
- Selección de las columnas de interés
- Creación de la columna *pct_**.
- Renombre de columnas.
- Aplicación de la función *yes_no_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Con esto se logra el *DataFrame df_innovation2*.

Personal en actividades de innovación en 2018:

- Se utiliza el archivo *innonce19_04* leído previamente.
- Selección de las columnas de interés
- Creación de la columna *pct_**. Se debe tener cuidado con las divisiones por cero. Por ello se condiciona la división sólo si el denominador es diferente de cero y luego llenan las columnas vacías con ceros.

- Renombre de columnas.
- Aplicación de la función *categories_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Con esto se logra el *DataFrame df_innovation3*.

Innovación por año:

- Lectura del archivo *innonce19_06*.
- Definición de columnas de interés.
- Aplicación de la función *general_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de la columna *category*.
- Selección de las columnas de interés.
- Aplicación de la función *melt* de pandas para lograr una tabla en formato *tidy*.
- Creación de la columna *percentage*.
- Con esto se logra el *DataFrame df_innovation4*.

Finalmente se concatenan los *DataFrames df_innovation1*, *df_innovation2*, *df_innovation3* y *df_innovation4* para obtener una única tabla con los indicadores de innovación.

Indicadores de medio ambiente

Conocimiento de las normas ambientales:

- Lectura del archivo *medance19_01*.
- Eliminación de filas innecesarias del archivo
- Definición y renombre de columnas de interés.
- Selección de filas no vacías en la columna *sector_id*.
- Modificación de las columnas *ent_id*, *nation_id* y *sector_id* para dejarlas estandarizadas.

- Se crean dos *DataFrames*, uno para cantidad de empresas y otro para valores porcentuales. A ambos se le aplica la función *melt* de pandas para llevar las tablas a formato *tidy*.
- Aplicación de la función *merge* de pandas para unir ambas tablas.
- Creación de las columnas *subsector_id* e *indicator*.
- Con esto se logra el *DataFrame df_enviromental1*.

Personal involucrado en actividades de protección ambiental:

- Lectura del archivo *medance19_05*.
- Eliminación de filas innecesarias del archivo
- Definición y renombre de columnas de interés.
- Selección de filas no vacías en la columna *sector_id*.
- Modificación o creación de las columnas *ent_id*, *nation_id*, *sector_id subsector_id* y *category* para dejarlas estandarizadas.
- Aplicación de la función *yes_no_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Con esto se logra el *DataFrame df_enviromental2*.

Separación de residuos:

- Lectura del archivo *medance19_03*.
- Definición del nombre de columnas de interés.
- Aplicación de las funciones environmental, *yes_no_format* y *categories_format* alojadas en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Eliminación de la columna *category*.
- Renombre de la columna *indicator* por *category*.

- Creación de la columna *indicator*.
- Concatenación de tablas para generar el *DataFrame* *df_enviromental3*.

Gastos e inversión en actividades de protección del medio ambiente / Usos del agua:

- Lectura de los archivos *medance19_07*, *medance19_08* y *medance19_09*.
- Definición de variables y nombres de columnas que permiten procesar cada archivo según características propias de su estructura.
- Aplicación de la funciones *environmental* y *yes_no_format* alojadas en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Creación de un nuevo *DataFrame* donde se calculan los valores porcentuales de los indicadores.
- Modificación en el nombre de columnas que contienen el *string* "value _".
- Aplicación de la función *categories_format* alojada en la carpeta *util* y desarrollada para evitar pasos repetitivos en el pipeline.
- Eliminación de la columna *category*.
- Renombre de la columna *indicator* por *category*.
- Creación de la columna *indicator*.
- Concatenación de tablas para generar el *DataFrame* *df_enviromental4*.

Finalmente se concatenan los *DataFrames* *df_enviromental1*, *df_enviromental2*, *df_enviromental3* y *df_enviromental4* para obtener una única tabla con los indicadores de medio ambiente.

Como último paso y ya habiendo procesado todos los archivos, se concatenan los *DataFrames* finales de cada etapa para generar una única tabla con todos los indicadores, se usan los diccionarios definidos al inicio para estandarizar las columnas *indicator* y *category*, las columnas correspondientes se pasan a formato entero o flotante, según corresponda, y se reemplaza el *id* de nación por *mex*.

El *DataFrame* que se obtiene tras el proceso de ETL agrupa todos los indicadores en una tabla con solo 8 columnas y varios miles de filas. Un extracto de la tabla final se muestra a continuación:

nation_id	ent_id	sector_id	subsector_id	value	percentage	indicator	category
1	0	11	0	1142	5.79	115	1
1	0	11	0	488	12.95	115	2
1	0	11	0	143	17.4	115	3
1	0	11	0	22	41.51	115	4
1	0	0	112	325	10.56	115	1
...	...	...	...	...	...	...	...
1	0	0	812	0	0	12	4
1	0	0	813	550	3.61	12	1
1	0	0	813	151	4.49	12	2
1	0	0	813	4	2.58	12	3
1	0	0	813	0	0	12	4

Figura X. Formato final tabla de indicadores del Censo Económico 2019 - Financiamiento

Funciones adicionales

Durante la explicación del proceso de ETL se mencionaron algunas funciones desarrolladas para evitar código repetitivo al momento de procesar cada archivo. A continuación se entrega una breve descripción de ellas.

1) general_format

A la función se le pasan 6 parámetros que se utilizan en diferentes partes del código: *df*, *a*, *b*, *column_name*, *fix_subsector*, *rows_drop*.

- El primer parámetro corresponde al *DataFrame*.
- El segundo y tercer parámetro son útiles para eliminar filas vacías de los archivos.
- El cuarto parámetro es una variable que contiene el nombre de las columnas del *DataFrame*.

- El quinto parámetro permite reemplazar una palabra en la columna *subsector_id*.
- El último parámetro permite eliminar de la columna *category* todas las filas que contengan los *string* indicados en el parámetro.
- Adicionalmente se aplican otras transformaciones para estandarizar el *DataFrame*, entre ellas, modificaciones en las filas *category*, *sector_id* y *subsector_id*; la columna *ent_id* contendrá sólo ceros y la columna *nation_id* contendrá sólo unos.

2) yes_no_format

A la función se le pasan 3 parámetros que se utilizan en diferentes partes del código: *df*, *a* y *b*.

- Mediante un ciclo *for* se crean dos *DataFrames* con un grupo de columnas en común y otras columnas dadas por los parámetros *a* y *b*.
- Creación de la columna *indicator* usando los parámetros *a* y *b*.
- Renombre de columnas
- Concatenación de ambos *DataFrames*.

3) categories_format

A la función se le pasan 2 parámetros que se utilizan en diferentes partes del código: *df* y *names*.

- Creación de un *DataFrame* que contiene un grupo de columnas fijas y un listado de columnas dadas por el parámetro *names* que contienen el *string* "value".
- Se aplica la función *melt* de pandas para llevar el *DataFrame* a un formato *tidy*.
- Se repite el proceso creando un segundo *DataFrame* que ahora contendrá las columnas dadas por el parámetro *names* que contienen el *string* "pct".
- Se utiliza la función *merge* de pandas para unir ambos *DataFrames* donde corresponda.

4) geo_data

A la función se le pasan 4 parámetros que se utilizan en diferentes partes del código: *df*, *selected_columns*, *cut1* y *cut2*.

- Los parámetros *cut1* y *cut2* se utilizan para eliminar filas innecesarias de los archivos.
- El parámetro *selected_columns* se utiliza para crear el *DataFrame* con las columnas de interés contenidas en el parámetro.
- Se eliminan filas que contengan los *strings* “personas”, “Todos” y “Rama” en la columna *category*.
- Se modifican las columnas *sector_id*, *subsector_id*, *ent_id* y *nation_id* para estandarizar los valores.

5) environmental

A la función se le pasan 4 parámetros que se utilizan en diferentes partes del código: *df*, *a*, *b* y *column_name*.

- Los parámetros *a* y *b* se utilizan para eliminar filas innecesarias de los archivos.
- El parámetro *column_name* se utiliza para renombrar las columnas del *DataFrame*.
- Se seleccionan las filas de interés: aquellas que presentan "00 NAL" en la columna *ent_id* y aquellas donde *sector_id* es nulo.
- Se modifican las columnas *sector_id*, *subsector_id*, *ent_id* y *nation_id* para estandarizar los valores.

2. Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH)

2.1. Inegi_enigh_household_income

Descripción General y Ejecución

El pipeline *enigh_household_income_pipeline* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con la Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH). En ello, el pipeline se refiere a los datos de ingreso del hogar. Los datos son descargados desde la plataforma del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/enigh/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli) `~/data_etl/etl/enigh/$ bamboo-cli --folder . --entry enigh_household_income_pipeline --year=<year_value>`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y existe un archivo por año. Para el año 2018 y luego de seleccionar las

columnas de interés, se identifican 5 columnas y 74.647 filas. A continuación, se visualiza la estructura inicial de los datos.

	folioviv	ubica_geo	factor	tot_integ	trabajo
0	100013601	1001	175	3	53114.74
1	100013602	1001	175	5	0.00
2	100013603	1001	175	2	141885.21
3	100013604	1001	175	2	0.00
4	100013606	1001	175	4	8852.45

Figura X. Formato original datos ingreso de los hogares (ENIGH)

Extracción y Transformación

En este etapa se procede de la siguiente forma:

- Lectura del archivo. Se incorpora el parámetro *usecols* para seleccionar sólo las columnas requeridas en el procesamiento de los datos.
- Creación de la columna *year*, con referencia al año del archivo.
- Creación de la columna *trabajo_viv_mes* que corresponde al ingreso mensual. Esta columna se calcula dividiendo por tres la columna trabajo, que hace referencia al ingreso trimestral.
- Eliminación de las columnas *trabajo* y *folioviv.*
- Completar los valores en la columna *ubica_geo* con 5 *caracteres* para estandarizar los datos.
- Renombre de las columnas.
- Definición de la función *to_interval*, la cual será de utilidad para crear rangos salariales.
- Lectura del archivo *income* que contiene los intervalos de ingreso que serán utilizados en la función *to_interval*.
- Reemplazo del ingreso mensual por su id con la función mencionada previamente.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 5 columnas y 74.647 filas. La estructura final de la tabla se presenta a continuación:

	mun_id	households	n_people_home	year	monthly_wage
0	1001.0	175.0	3.0	2018.0	18.0
1	1001.0	175.0	5.0	2018.0	1.0
2	1001.0	175.0	2.0	2018.0	41.0
3	1001.0	175.0	2.0	2018.0	1.0
4	1001.0	175.0	4.0	2018.0	3.0

Figura X. Formato final tabla de datos ingreso de los hogares (ENIGH)

2.2. Inegi_enigh_jobs

Descripción General y Ejecución

El pipeline *enigh_jobs_pipeline* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con la Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH). En ello, el pipeline se refiere a los datos de empleo. Los datos son descargados desde la plataforma del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de Clickhouse, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/enigh/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli)

```
~/data_etl/etl/enigh/$ bamboo-cli --folder . --entry enigh_jobs_pipeline --year=<year_value>
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y en archivos anuales, además, se utilizan tres archivos: Empleo, Vivienda, y Población. Con respecto a su estructura y considerando los archivos del año 2018, se identifican en el archivo de Empleo 139.933 filas y 13 columnas; en el de

Vivienda 73.405 filas y 4 columnas; y en que se refiere a Población 269.206 filas y 5 columnas. A continuación, se visualiza la estructura inicial de los datos.

	folioviv	foliohog	numren	id_trabajo	trapais	pago	contrato	tipocontr	htrab	sinco	scian	clas_emp	tam_emp
0	100013601	1	1	1	1	1	1	2	48	2614	1121	1	2
1	100013601	1	3	1	1	1	2		36	8341	4840	1	1
2	100013602	1	1	1	1				48	2135	5411		1
3	100013602	1	3	1	1				1	2716	6141		1
4	100013603	1	1	1	1	1	1	2	40	2271	5411	2	10

Figura X. Formato original datos de empleo ENIGH

	folioviv	ubica_geo	est_socio	factor
0	100013601	1001	3	175
1	100013602	1001	3	175
2	100013603	1001	3	175
3	100013604	1001	3	175
4	100013606	1001	3	175

Figura X. Formato original datos de vivienda ENIGH

	folioviv	foliohog	numren	sexo	edad
0	100013601	1	1	1	74
1	100013601	1	2	2	70
2	100013601	1	3	1	38
3	100013602	1	1	1	48
4	100013602	1	2	2	47

Figura X. Formato original datos de población ENIGH

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de los archivos de Empleo y Vivienda.
- Reemplazo de valores faltantes con el valor 99.
- Uso de la función *merge* de Pandas para agrupar ambos archivos (ahora *DataFrames*) en un mismo *DataFrame*. El nuevo *DataFrame* será definido en adelante como df.

- Creación de nueva columna *mun_id*, la cual toma relación con el id del municipio. Dicha columna se crea con los primeros 4 dígitos de la columna *ubica_geo*.
- Lectura del archivo de Población.
- Creación de una nueva columna en los *DataFrames* relacionada con la Población. Se asigna como nombre *coding* a la nueva variable, y su valor es la agregación de las columnas: *folioviv*, *foliohog*, y *numren*.
- Uso de la función *merge* de Pandas para unir ambos *DataFrames* en uno solo. El *merge* se hace bajo la nueva columna *coding*. En adelante, el único *DataFrame* en transformación toma como nombre df.
- Reemplazar los valores de diferentes columnas haciendo uso de un *loop*. Dicho *loop* obtiene la data desde un Google Spreadsheet y genera diccionarios para identificar los valores a reemplazar.
- Renombre de las columnas al inglés.
- Agrupación de datos. Dicha agrupación se lleva a cabo considerando las columnas en la lista *group_list*.
- Reemplazo de valores 99 por *np.nan*. Esto se realiza con motivo de reemplazar, posteriormente, todos los valores *np.nan* por 999999. Lo anterior con motivo de estandarizar el reemplazo de data faltante.
- Creación de nueva columna *year*, considerando el año de estudio.
- Cambio de *dtype* para columnas específicas.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 139.717 filas y 16 columnas. La estructura final de la tabla se presenta a continuación:

	sex	age	job_id	national_job	sinco_id	scian_id	eco_stratum	business_size	mun_id	pay_mode	contract	contract_type	business_type	worked_hou
0	1	101	1	1	6121	1121	2	2	19022	NaN	NaN	NaN	NaN	7
1	1	12	1	1	2152	5110	3	3	17011	1.0	2.0	NaN	2.0	40
2	1	12	1	1	2172	7111	1	3	13062	1.0	2.0	NaN	1.0	24
3	1	12	1	1	2511	4611	1	2	31047	1.0	2.0	NaN	1.0	12
4	1	12	1	1	2531	5414	3	2	12035	2.0	NaN	NaN	1.0	10

Figura X. Formato final tabla de datos empleo ENIGH

2.3. Inegi_enigh_population

Descripción General y Ejecución

El pipeline *enigh_population_pipeline* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con la Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH). En ello, el pipeline se refiere a los datos de Población. Los datos son descargados desde la plataforma del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de Clickhouse, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/enigh/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli)

```
~/data_etl/etl/enigh/$ bamboo-cli --folder . --entry enigh_population_pipeline
--year=<year_value>
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y en archivos anuales. Serán utilizados dos archivos con referencia a Población y Hogar. Con respecto a su estructura y considerando los archivos del año 2018, se identifican en el archivo de Población 269.206 filas y 178 columnas; y en el de

Hogar 74.647 filas y 5 columnas. A continuación, se visualiza la estructura inicial de los datos:

	folioviv	foliohog	numren	parentesco	sexo	edad	madre_hog	madre_id	padre_hog	padre_id	...	segvol_6	segvol_7	hijos_viv	hijos_mue	hijo
0	100013601	1	1	101	1	74	2		2		...	6				
1	100013601	1	2	201	2	70	2		2		...	6		6	0	
2	100013601	1	3	301	1	38	1	&	1	&	...	6				
3	100013602	1	1	101	1	48	2		2		...	6				
4	100013602	1	2	201	2	47	2		2		...	6		3	0	

Figura X. Formato original datos de población ENIGH

	folioviv	foliohog	ubica_geo	est_socio	factor
0	100013601	1	1001	3	175
1	100013602	1	1001	3	175
2	100013603	1	1001	3	175
3	100013604	1	1001	3	175
4	100013606	1	1001	3	175

Figura X. Formato original datos de hogar ENIGH

Lectura y Transformación

Esta etapa sigue el proceso definido a continuación:

- Lectura de los archivos de Población y Hogar. En ello, se agrega el parámetro *usecols* a la función *read_csv* de Pandas para seleccionar las columnas necesarias en el desarrollo del ETL.
- Creación de una nueva columna llamada *code*. Dicha variable se construye a partir de la agregación de las columnas: *folioviv* y *foliohog*. Dicha columna se construye en ambos archivos.
- Utilizando la función *merge* de Pandas se procede a unir ambos *DataFrames*.
- Creación de una nueva columna que define el municipio asociado. La nueva variable es llamada *mun_id*. Esta variable se construye a partir de los primeros cuatro caracteres de los valores en la columna *ubica_geo*.
- Reemplazo de valores faltantes por 0.
- Creación de la columna *months_social_security*. Dicha columna se crea a partir de la agregación de las variables *ss_aay ss_mm*.

- Reemplazo de valores por *pd.np.nan* para las personas que son menores de edad o no se han adherido al *Social Security*.
- Reemplazo de valores faltantes por 999999.
- Reemplazar los valores de diferentes columnas haciendo uso de un *loop*. Dicho *loop* obtiene la data desde un Google Spreadsheet y genera diccionarios para identificar los valores a reemplazar.
- Renombre de columnas al inglés.
- Agrupación de datos. Dicha agrupación se lleva a cabo considerando las columnas en la lista *group_list*.
- Reemplazo de valores 999999 por *pd.np.nan*.
- Creación de nueva columna *year*, considerando el año de estudio.
- Cambio de *dtype* para columnas específicas.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 86.348 filas y 35 columnas. La estructura final de la tabla se presenta a continuación:

	mun_id	sex	age	speaks_native	ethnicity	academic_degree	previous_entity_5_years	social_security	months_social_security	near_support_money	...
0	1001	1	16	2	2	3	1	1	4	4.0	...
1	1001	1	17	2	2	3	1	1	4	4.0	...
2	1001	1	17	2	2	4	1	1	1	3.0	...
3	1001	1	18	2	1	3	1	1	2	3.0	...
4	1001	1	18	2	1	3	1	1	6	3.0	...

Figura X. Formato final tabla de datos de población ENIGH

2.4. inegi_enigh_income

Descripción General y Ejecución

El pipeline *enigh_decile_income* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con la Encuesta Nacional de Ingresos y Gastos de los Hogares (ENIGH). En ello, el pipeline se refiere a los deciles de ingreso. Los datos son descargados desde la plataforma del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de Clickhouse, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/enigh/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli)

```
~/data_etl/etl/enigh/$ bamboo-cli --folder . --entry enigh_decile_income --year=<year_value>
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx; y se identifican 12 columnas y 44 filas. A continuación, se visualiza la estructura inicial de los datos.

	Unnamed: 0	I	II	III	IV	V	VI	VII	VIII	IX	X
0	Estados Unidos Mexicanos	9113.0	16100.0	21428.0	26696.0	32318.0	38957.0	47264.0	58885.0	78591.0	166750.0
1	Aguascalientes	12902.0	21645.0	28059.0	34001.0	41215.0	48847.0	57987.0	70360.0	92628.0	185797.0
2	Baja California	14265.0	22711.0	28532.0	34732.0	41413.0	48917.0	58997.0	73533.0	98477.0	170193.0
3	Baja California Sur	15578.0	26001.0	33234.0	40973.0	48877.0	57578.0	69223.0	84042.0	104247.0	208003.0
4	Campeche	8686.0	14720.0	19538.0	24371.0	29640.0	35722.0	44455.0	57599.0	77733.0	164497.0
5	Coahuila de Zaragoza	12288.0	20587.0	26802.0	32449.0	38573.0	45343.0	54510.0	66729.0	89461.0	172504.0

Figura X. Formato original datos deciles de ingresos ENIGH

Extracción y Transformación

Este paso del pipeline sigue el siguiente proceso:

- Transformar el archivo .xlsx a un *DataFrame* de Pandas.
- Renombre de la columna *Unnamed: 0* a *ent_id*.
- Lectura de un archivo que contiene los estados, para reemplazar el nombre del estado en la variable ent_id con el id del estado.
- Transformación de la estructura a un formato *tidy*. Para ello, se utiliza la función *melt* de Pandas. La idea de ello es tener una mejor visualización de los deciles de Ingreso.
- Creación de la columna *year* que toma relación al año de estudio.

Posterior al proceso de transformación, obtenemos un *DataFrame* con 4 columnas y 330 filas. La estructura final de la tabla se presenta a continuación:

	ent_id	decile	value	year
0	33	I	9113	2018
1	1	I	12902	2018
2	2	I	14265	2018
3	3	I	15578	2018
4	4	I	8686	2018
...	...	...	...	...

Figura X. Formato final tabla de datos con deciles de ingreso

3. Encuesta Nacional de Ocupación y Empleo (ENOE)

3.1. Inegi_enoe

Descripción general y ejecución

El pipeline *enoe_pipeline.py* de ENOE (Encuesta Nacional de Ocupación y Empleo) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos de dicha encuesta entregados por INEGI (Instituto Nacional de Estadística y Geografía). Estos datos ofrecen información mensual y trimestral de la fuerza de trabajo, la ocupación, la informalidad laboral, la subocupación y la desocupación.

Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli)

```
~/data-etl/etl/enoe/$ bamboo-cli --folder . --entry enoe_pipeline
```

Descarga de datos

Desde el sitio de INEGI se descargan cuatro archivos para ingestar, estos son *"coe1t", "coe2t", "sdemt" y "vivt"*, todos en formato CSV, cada uno corresponde a una parte de

la ENOE y a un año y trimestre específico. Una vez descargados estos archivos son almacenados en un compartimiento privado de GCP storage, luego, se establece una conexión privada validada con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Lectura de datos

Una vez descargados los datos, estos se leen y almacenan en un *DataFrame* de la librería pandas, cada uno de los archivos son leídos por separados y guardados en *DataFrames* separados, a continuación se presentan las primeras filas y columnas de estos archivos.

	cd_a	ent	con	v_sel	n_hog	h_mud	n_ren	eda	p1b	p2_1	...	p2_9	p2a_anio	p2b	\
0	01	09	00501	01	1	0	01	35			...		NaN		
1	01	09	00501	01	1	0	02	33	2		...		2009	1	
2	01	09	00501	02	1	0	01	32	1		...		NaN		
3	01	09	00501	02	1	0	02	29	1		...		NaN		
4	01	09	00501	03	1	0	01	54	2		...		NaN		

	p3	p4a	p5b_thrs	p5b_tdia	fac	code	population_monthly
0	6270	5510	032	4	1060	0900501011001	NaN
1	NaN	NaN	NaN		1060	0900501011002	NaN
2	6260	2210	NaN		1060	0900501021001	NaN
3	1330	6112	NaN		1060	0900501021002	NaN
4	NaN	NaN	NaN		1060	0900501031001	NaN

Figura X. Formato original datos del archivo coeIt

Al momento de la elaboración de esta documentación, el *DataFrame* inicial proveniente del archivo *"coe1t"* contenía 312.167 filas por 23 columnas.

	ent	con	v_sel	n_hog	h_mud	n_ren	p6b1	p6b2	p6c	p6d	p7	p7a	p7c	\
0	09	00501	01	1	0	01	7	NaN	6	2	7	NaN	NaN	
1	09	00501	01	1	0	02		NaN				NaN	NaN	
2	09	00501	02	1	0	01	8	NaN	5	1	7	NaN	NaN	
3	09	00501	02	1	0	02	8	NaN	5	3	7	NaN	NaN	
4	09	00501	03	1	0	01		NaN				NaN	NaN	

	code
0	0900501011001
1	0900501011002
2	0900501021001
3	0900501021002
4	0900501031001

Figura X. Formato original datos del archivo coe2t

El *DataFrame* proveniente del archivo *"coe2t*" contenía 312.167 filas por 14 columnas

	ent	con	v_sel	n_hog	h_mud	n_ren	sex	cs_p13_1	cs_p13_2	clase1	...	\
0	09	00506	05	1	0	02	2	03	3	1	...	
1	09	00517	03	1	0	03	2	07	4	1	...	
2	09	00525	05	1	0	04	1	NaN		0	...	
3	09	00532	03	1	0	03	1	03	2	1	...	
4	09	00532	04	1	0	03	2	03	2	1	...	

	dur_des	sub_o	s_clasifi	hij5c	anios_esc	ingocup	cp_anoc	ma48me1sm	emp_ppal	\
0	0	1	5	2	9	6000	0	0	1	
1	0	0	0	1	16	0	0	0	1	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	8	3870	0	0	2	
4	0	0	0	1	8	0	0	0	1	

	code
0	0900506051002
1	0900517031003
2	0900525051004
3	0900532031003
4	0900532041003

Figura X. Formato original datos del archivo sdmet

El *DataFrame* proveniente del archivo "sdmet" contenía 406.797 filas por 24 columnas.

	mun	ent	con	v_sel	code
0	001	01	00001	01	010000101
1	001	01	00001	02	010000102
2	001	01	00001	03	010000103
3	001	01	00001	04	010000104
4	001	01	00001	05	010000105

Figura X. Formato original datos del archivo vivt

El *DataFrame* proveniente de *"vivt"* contiene 120.427 filas por 5 columnas.

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se estandarizan los nombres de las columnas de los dos primeros *DataFrames*, quitando símbolos no deseados y dejando todo en letras minúsculas. Continuando con la estandarización, se reindexan ambos *DataFrames* basándose en valores únicos individuales de sus columnas. Finalmente, se juntan ambos *DataFrames* en uno.

Posteriormente, se cargan los siguientes dos *DataFrames*, los relacionados a datos sociodemográficos y a vivienda, los cuales son sometidos a estandarización de sus datos tal como los anteriores. Luego se juntan estos *DataFrames* con los dos anteriores, generando un solo *DataFrame* que contiene todos los datos.

Se generan los números identificadores geográficos, basados en los municipios y entidades federativas. Luego se reemplazan los valores nulos por valores numéricos para poder realizar agregaciones de los datos. Se generan números identificadores para cada columna y este reemplaza al valor original en el *DataFrame*, por ejemplo; "Posee un trabajo o negocio" tendrá el número 1 si posee trabajo o negocio y 2 si no posee. Esta referencia será guardada por separado en otra tabla, de la forma; "posee trabajo o negocio" = 1; "No" = 2.

Posteriormente creados todos los ID's, se procede a agrupar el *DataFrame* por cada fila única. Luego se integran en el *DataFrame* los valores de ingreso para cada una de las filas.

Finalmente, se estandariza el *DataFrame* por última vez, quitando los valores repetidos e innecesarios, también validando el tipo de dato que es cada columna, dejándolas en un tipo de dato estándar para que sea leído apropiadamente por la base de datos.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 288.614 filas y 23 columnas. La estructura final de la tabla se presenta a continuación.

	code	mun_id	population	population_monthly	mensual_wage	\
0	1T20101	9002	1060	NaN	0.0	
1	2T20101	9002	1060	NaN	0.0	
2	3T20101	9002	1060	NaN	0.0	
3	4T20101	9002	1060	NaN	0.0	
4	5T20101	9002	1060	NaN	0.0	

	has_job_or_business	second_activity	eap	occ_unocc_pop	eap_comp	...	\
0	NaN	7.0	1.0	1.0	1.0	...	
1	2.0	NaN	1.0	2.0	6.0	...	
2	1.0	7.0	1.0	1.0	3.0	...	
3	1.0	7.0	1.0	1.0	3.0	...	
4	2.0	NaN	2.0	3.0	0.0	...	

	classification_formal_informal_jobs_first_activity	age	\
0	2.0	35	
1	0.0	33	
2	2.0	32	
3	2.0	29	
4	0.0	54	

	actual_job_industry_group_id	sex	actual_job_position	\
0	5510	1.0	6270.0	
1	0	2.0	NaN	
2	2210	1.0	6260.0	
3	6112	2.0	1330.0	
4	0	1.0	NaN	

	actual_job_hrs_worked_lastweek	actual_job_days_worked_lastweek	\
0	32.0	4.0	
1	NaN	NaN	
2	NaN	NaN	
3	NaN	NaN	
4	NaN	NaN	

	workforce_is_wage	workforce_is_wage_monthly	quarter_id
0	1060	NaN	20101
1	1060	NaN	20101
2	1060	NaN	20101
3	1060	NaN	20101
4	1060	NaN	20101

Figura X. Formato final tabla de datos ENOE

3.2. inegi_etoe

Descripción general y ejecución

El pipeline *etoe_pipeline.py* de ETOE (Encuesta Telefónica de Ocupación y Empleo) es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos de dicha encuesta entregados por INEGI (Instituto Nacional de Estadística y Geografía). Estos datos ofrecen información relevante para monitorear la situación de la ocupación y empleo en el periodo de contingencia del COVID-19.

Los datos son descargados desde el repositorio de datos de INEGI y almacenados en un compartimiento de Google Storage privado, pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli)

```
~/data-etl/etl/etoe/$ bamboo-cli --folder . --entry etoe_pipeline
```

Descarga de datos

Desde el sitio de INEGI se descargan cuatro archivos para ingestar, estos son *"coe1t", "coe2t", "sdemt" y "vivt"*, todos en formato DBF, cada uno corresponde a una parte de la encuesta ETOE y a un año y trimestre específico. Una vez descargados estos archivos son almacenados en un compartimiento privado de GCP storage, luego, se

establece una conexión privada validada con credenciales fuertemente encriptadas, para ejecutar una descarga segura hasta el servidor de procesamiento de los datos.

Lectura de datos

Una vez descargados los datos, estos se leen y almacenan en un *DataFrame* de la librería pandas, cada uno de los archivos son leídos por separados y guardados en *DataFrames* separados, a continuación se presentan las primeras filas y columnas de estos archivos.

	ent	cd_a	con	v_sel	n_hog	h_mud	n_ren	eda	p1b	p2_1	...	p2_4	p2_9	\
0	01	14	40265	05	1	0	01	55	NaN	NaN	...	NaN	NaN	
1	01	14	40265	05	1	0	02	45	2	NaN	...	4	NaN	
2	01	14	40299	03	1	0	01	64	2	NaN	...	4	NaN	
3	01	14	40299	03	1	0	02	33	1	NaN	...	NaN	NaN	
4	01	14	40299	03	1	0	03	35	NaN	NaN	...	NaN	NaN	

	p2a_anio	p2b	p3	p4a	p5b_thrs	p5b_tdia	fac	code
0	NaN	NaN	8341	4840	060	6	2715	0140265051001
1	NaN	NaN	NaN	NaN	NaN	NaN	2715	0140265051002
2	NaN	NaN	NaN	NaN	NaN	NaN	3396	0140299031001
3	NaN	NaN	5312	9313	NaN	NaN	3396	0140299031002
4	NaN	NaN	2512	4681	046	6	3396	0140299031003

Figura X. Formato original datos del archivo coeIt

Al momento de la elaboración de esta documentación, el *DataFrame* inicial proveniente del archivo *"coe1t"* contenía 23.893 filas por 22 columnas.

	ent	con	v_sel	n_hog	h_mud	n_ren	p6b1	p6b2	p6c	p6d	p7	p7a	p7c	\
0	01	40265	05	1	0	01	7	NaN	4	1	7	NaN	NaN	
1	01	40265	05	1	0	02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	01	40299	03	1	0	01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	01	40299	03	1	0	02	8	NaN	9	1	7	NaN	NaN	
4	01	40299	03	1	0	03	7	NaN	9	1	7	NaN	NaN	

	code
0	0140265051001
1	0140265051002
2	0140299031001
3	0140299031002
4	0140299031003

Figura X. Formato original datos del archivo coe2t

El *DataFrame* proveniente del archivo *"coe2t"* contenía 23.893 filas por 14 columnas

	ent	con	v_sel	n_hog	h_mud	n_ren	sex	clase1	clase2	clase3	...	\
0	09	40015	01	1	0	01	1	1	1	1	...	
1	09	40015	01	1	0	02	2	2	4	0	...	
2	09	40015	01	1	0	03	1	2	4	0	...	
3	09	40015	01	1	0	04	1	1	2	6	...	
4	09	40015	01	1	0	05	2	2	4	0	...	

	cs_p13_2	ingocup	d_ant_lab	d_cexp_est	dur_des	sub_o	s_clasifi	cp_anoc	\
0	5	40000	0	0	0	0	0	0	
1	3	0	0	0	0	0	0	0	
2	2	0	0	0	0	0	0	0	
3	3	0	1	2	3	0	0	0	
4	3	0	0	0	0	0	0	0	

	emp_ppal	code
0	2	0940015011001
1	0	0940015011002
2	0	0940015011003
3	0	0940015011004
4	0	0940015011005

Figura X. Formato original datos del archivo sdmet

El *DataFrame* proveniente del archivo *"sdmet"* contenía 29.704 filas por 24 columnas

	ent	con	v_sel	mun	code
0	01	40007	05	001	014000705
1	01	40011	01	001	014001101
2	01	40011	03	001	014001103
3	01	40015	03	001	014001503
4	01	40016	05	001	014001605

Figura X. Formato original datos del archivo vivt

El *DataFrame* proveniente de *"vivt"* contenía 14.185 filas por 5 columnas

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se estandarizan los nombres de las columnas de los dos primeros *DataFrames*, quitando símbolos no deseados y dejando todo en letras minúsculas. Continuando con la estandarización, se reindexan ambos *DataFrames* basándose en valores únicos individuales de sus columnas. Finalmente, se juntan ambos *DataFrames* en uno.

Posteriormente, se cargan los siguientes dos *DataFrames*, aquellos relacionados a datos sociodemográficos y a vivienda, los cuales son sometidos a estandarización de sus datos tal como los anteriores. Luego se juntan estos *DataFrames* con los dos anteriores, generando un solo *DataFrame* que contiene todos los datos.

Se generan los números identificadores geográficos, basados en los municipios y entidades federativas. Luego se reemplazan los valores nulos por valores numéricos para poder realizar agregaciones de los datos. Se generan números identificadores para cada columna y este reemplaza al valor original en el *DataFrame*, por ejemplo; "Posee un trabajo o negocio" tendrá el número 1 si posee trabajo o negocio y 2 si no posee. Esta referencia será guardada por separado en otra tabla, de la forma; "posee trabajo o negocio" = 1; "No" = 2.

Posteriormente creados todos los ID's, se procede a agrupar el *DataFrame* por cada fila única. Luego se integran en el *DataFrame* los valores de ingreso para cada una de las filas.

Finalmente, se estandariza el *DataFrame* por última vez, quitando los valores repetidos e innecesarios, también validando el tipo de dato que es cada columna, dejándolas en un tipo de dato estándar para que sea leído apropiadamente por la base de datos.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 21.315 filas y 42 columnas. La estructura final de la tabla se presenta a continuación:

	mun_id	represented_city	age	has_job_or_business	search_job_overseas	search_job_mexico	search_start_business	search_no_search	search_no_knowledge	search_job_year	...	work_history	unoccupied_condition	classification_duration_unemployment	underemployed_population	underemployed_classification	classification_self_employed_unqualified_activities	classification_formal_informal_jobs_first_activity	population	income_id	month_id
0	01001	14.0	15.0	2.0	NaN	NaN	NaN	1.0	NaN	NaN	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2534	NaN	202004
1	01001	14.0	15.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.0	0.0	0.0	1.0	3.0	0.0	1.0	4362	4.0	202004
2	01001	14.0	16.0	2.0	NaN	NaN	NaN	1.0	NaN	NaN	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3389	NaN	202004
3	01001	14.0	16.0	2.0	NaN	NaN	NaN	1.0	NaN	NaN	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4352	NaN	202004
4	01001	14.0	16.0	2.0	NaN	NaN	NaN	1.0	NaN	NaN	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3492	NaN	202004

Figura X. Formato final tabla de datos ETOE

4. Salud

4.1. Dgis_pregnancy_mortality

Descripción General y Ejecución

El pipeline *pregnancy_mortality_pipeline* procesa los datos facilitados por la Dirección General de Información en Salud (DGIS) que toman relación con mortalidad materna. Los datos fueron descargados desde la plataforma de la Dirección General de Información en Salud, y son almacenados en Google Storage para su posterior procesamiento. Ofrece información de las muertes maternas, indicando la causa del fallecimiento según CIE10, seguro social, grado académico, edad y municipio de residencia de las fallecidas.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de Clickhouse, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/healthcare/maternal_death/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

```
~/data_etl/etl/healthcare/maternal_death/$ source pregnancy_mortality_ingest.sh
```

(bamboo-cli)

```
~/data_etl/etl/healthcare/maternal_death/$ bamboo-cli --folder . --entry pregnancy_mortality_pipeline
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y se identifican 19.058 filas y 56 columnas. A continuación, se visualizan algunas columnas y filas de la estructura inicial de los datos

AÑO	EDAD	ENTIDAD	DESCRIPCIÓN I	MUNICIPIO	DESCRIPCIÓN	LOCALIDAD	DESCRIPCIÓN I	OCUPACIÓN	DESCRIPCIÓN DE	ESCOLARIDAD	DESCRIPCIÓN DE LA ESCOLARIDAD	CAUSA CIE	DESCRIPCIÓN DE LA CAUSA CIE
0	25	10	DURANGO	35	035 TEPEHUAN	38	100350038 EL D	0	NO ESPECIFICADO	3	PRIMARIA COMPLETA	O759	O759 COMPLICACIÓN NO ESPECIFI
1965	37	30	VERACRUZ DE I	87	087 XALAPA	1	300870001 XALA	2	NO OCUPADOS	2	PRIMARIA INCOMPLETA	O622	O622 OTRAS INERCIAS UTERINAS
0	33	10	DURANGO	12	012 LERDO	137	100120137 LA G	0	NO ESPECIFICADO	0	NO ESPECIFICADA	O759	O759 COMPLICACIÓN NO ESPECIFI
1939	35	12	GUERRERO	47	047 PEDRO ASC	35	120470035 RINC	2	NO OCUPADOS	3	PRIMARIA COMPLETA	O998	O998 OTRAS ENFERMEDADES ESPE
1945	30	10	DURANGO	4	004 CUENCAMI	21	100040021 12 DI	2	NO OCUPADOS	2	PRIMARIA INCOMPLETA	O441	O441 PLACENTA PREVIA CON HEMO
1958	23	15	MÉXICO	105	105 TLATLAYA	74	151050074 SAN	2	NO OCUPADOS	5	SECUNDARIA COMPLETA	O721	O721 OTRAS HEMORRAGIAS POSTP
1944	36	30	VERACRUZ DE I	83	083 IXHUATLAN	68	300830068 SANT	2	NO OCUPADOS	1	NINGUNA	O064	O064 ABORTO NO ESPECIFICADO IN

Figura X. Formato original datos de mortalidad materna

Lectura y Transformación

El procesamiento de los datos sigue el proceso detallado a continuación:

- Lectura de los datos desde Google Storage. En ello, se utiliza el parámetro *chunksize* de la función *read_csv* de Pandas para iterar sobre el conjunto de datos y leer por sub conjuntos. Esto se realiza cuando existen grandes volúmenes de datos y no se quiere colapsar la memoria. Además, se agrega el parámetro *usecols* para seleccionar las columnas relevantes en este proceso de ETL. En términos generales, se omiten las columnas cuyo título contiene la palabra “descripción”.
- Renombre de las columnas.
- Creación de la columna *mun_residence_id*. Dicha columna se calcula mediante la agregación de las columnas: Entidad de residencia y Municipio de residencia. Esta variable es formulada para hacer referencia al lugar donde vivía la persona fallecida.
- Creación de la columna *mun_happening_id*. Dicha columna se calcula mediante la agregación de las columnas: Entidad de ocurrencia y Municipio de ocurrencia. Esta variable es formulada para hacer referencia al lugar donde ocurre la muerte de la embarazada.
- Eliminación de variables no relevantes.
- Traducción de columnas al inglés. Dicha traducción se ejecuta con la lectura de un archivo Google Spreadsheet que contiene una tabla con los renombres a ejecutar.

- Creación de la columna *count* con el valor único igual a 1. Esto se realiza con motivo de realizar agrupaciones posteriormente.
- Agrupación de datos a conveniencia. La agrupación se realiza por las columnas contenidas en la lista *group_list*.
- Reemplazo de los valores en las columnas *academic_degree* y *social_security* por sus id's según archivo de dimensiones leído anteriormente.
- Transformación de columnas a valores enteros.

Posterior al proceso de transformación, obtenemos un *DataFrame* con 19054 filas y 12 columnas. La estructura final de la tabla se presenta a continuación:

age	marital_status	occupation	academic_degree	social_security	medical_center	year_decease	cie10	year_of_register	mun_residence_id	mun_happening_id
11	8	98	2	8	2	2009	O450	2009	7112	7078
12	0	2	3	8	1	2005	C58X	2005	30141	30193
12	1	2	2	1	1	2011	O150	2011	30204	30039
12	1	71	2	8	1	2005	O873	2005	15037	15057
12	1	11	3	1	1	2014	O021	2014	27004	27004

Figura X. Formato original datos del Censo Económico

4.2. Gobmx_covid

Descripción general y ejecución

El pipeline de datos de la situación mexicana frente al COVID-19, *covid_pipeline.py*, es una herramienta de ETL (Extracción, Transformación y Carga de datos). Estos datos ofrecen información respecto al último balance de los casos de contagios reportados en México, considerando datos geográficos y características de los contagiados, fallecidos y casos sospechosos (edad, sexo y comorbilidades).

Los datos son descargados desde el repositorio de datos del gobierno de México y pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli)

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_pipeline
```

Si el archivo de datos ya se encuentra descargado, entonces se puede pasar como parámetro del pipeline el directorio donde está localizado el archivo.

(bamboo-cli)

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_pipeline
--file_path='/path/to/file'
```

Descarga de datos

Para efectuar la descarga de datos, el pipeline se conecta directamente a la página fuente, usando la clase *DownloadStep* de *Bamboo*. El archivo viene comprimido en formato ZIP, por lo que es descomprimido y puesto a disposición del siguiente paso del pipeline.

Lectura de datos

Una vez descargados los datos, estos se leen y almacenan en un *DataFrame* de la librería pandas.

Al momento de la elaboración de esta documentación, el *DataFrame* inicial contenía 7.228.942 filas por 40 columnas, sin embargo este valor crece día a día ya que el pipeline es ejecutado a diario para mantener actualizados los datos. La siguiente imagen muestra las primeras filas y algunas columnas del *DataFrame* inicial.

	FECHA_ACTUALIZACION	ID_REGISTRO	ORIGEN	SECTOR	ENTIDAD_UM	SEXO	ENTIDAD_NAC	ENTIDAD_RES	MUNICIPIO_RES	TIPO_PACIENTE	...
0	2021-06-13	z482b8	1	12	9	2	9	9	12	1	...
1	2021-06-13	z49a69	1	12	23	1	23	23	4	2	...
2	2021-06-13	z23d9d	1	12	22	2	24	22	9	1	...
3	2021-06-13	z24953	1	12	9	1	9	9	10	1	...
4	2021-06-13	zz8e77	1	12	9	2	9	9	2	1	...
...	...	...	...	...	...	...	...	...	...	...	...

Figura X. Formato original reporte diario COVID-19

Transformación y limpieza

Posterior a la lectura de los datos, estos son sometidos al proceso de transformación y limpieza para poder obtenerlos en el formato deseado. Primero, se estandarizan los nombres de las columnas del *DataFrame*, quitando símbolos no deseados de estos, renombrando las columnas con nombres en inglés fáciles de identificar y dejando en letras minúsculas todas las palabras. Continuando con la estandarización, se estandarizan los formatos de las columnas relacionadas a entidad y municipio haciendo que sus valores sean del tipo *string* (*caracteres*).

Las columnas que informan si el paciente falleció, fecha de fallecimiento, país de origen y nacionalidad son estandarizadas en el siguiente paso, quitando sus valores nulos y reemplazándolos por valores numéricos donde corresponde. Por ejemplo, en la columna *is_dead*, donde había un valor NaN, ahora habrá un 0.

Finalmente, se reemplazan los valores no conocidos de los números identificadores de municipalidades por un valor estándar y se estandarizan los ID de *covid_positive* donde 1 indica que el paciente dio positivo al covid, 2 si es negativo y 3 si no se ha obtenido resultado (caso sospechoso).

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 7.228.942 filas y 39 columnas. La estructura final de la tabla se presenta a continuación:

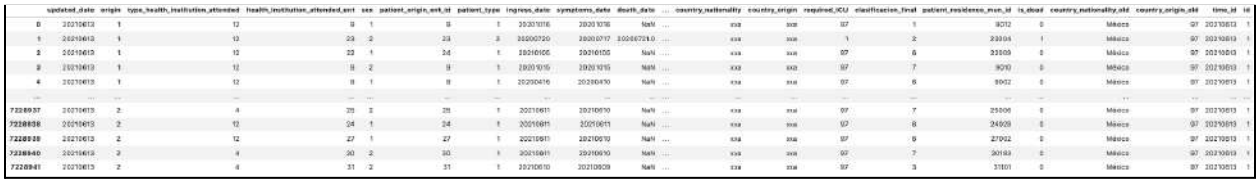


Figura X. Formato final tabla de datos COVID-19

4.3. Gobmx_covid_stats_nation

Descripción general y ejecución

El pipeline de estadísticas de la situación mexicana frente al COVID-19, *covid_stats_nation*.py, es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del cubo *gobmx_covid* para obtener diferentes métricas asociadas a la evolución de la pandemia a nivel nacional.

Estos datos ofrecen información respecto de la evolución de los casos de positivos, fallecidos, sospechosos y hospitalizados, informando los valores diarios, acumulados, promedios móviles a 7 días y tasas de contagios sobre el total de la población.

Los datos son obtenidos desde la API de tesseract construida por Datawheel para el sitio DataMéxico. También se usan los datos del último reporte COVID-19 entregado por el gobierno mexicano en su repositorio. Pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Cabe destacar que este pipeline es necesario para obtener las métricas correctas a diferentes niveles geográficos, en este caso, a nivel nacional. Al momento de ingestar los datos, cuando hay diferentes niveles geográficos estos se agrupan desde los niveles más pequeños a los niveles superiores en sumas o promedios simples. En el caso de las estadísticas del COVID-19 se calcula, por ejemplo, el promedio móvil a 7 días, medida que al ser agregada a diferentes niveles geográficos no representaría el valor real del indicador.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli)

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_nation
```

Si el archivo de datos ya se encuentra descargado, entonces se puede pasar como parámetro del pipeline el directorio donde está localizado el archivo.

(bamboo-cli)

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_nation
--file_path='/path/to/file'
```

Lectura de datos

Los datos son extraídos desde la API usando la librería *requests* de python, la que permite extraer directamente los datos desde la API y recibirlos en formato JSON, estos se leen y almacenan en un *DataFrame* de la librería pandas. Son 4 las llamadas a la API que se realizan para obtener el *DataFrame* inicial. Una vez recibida cada llamada, se guarda en un *DataFrame* y se junta con los datos anteriores. Cabe destacar que las llamadas obtienen datos que toman relación con: casos, muertes, hospitalizados, y sospechosos.

Al momento de la elaboración de esta documentación, el *DataFrame* inicial contenía 1 fila y 9 columnas. La siguiente imagen muestra un extracto del *DataFrame* inicial. Cabe destacar que posee sólo una fila ya que son los agregados a nivel nacional para la última fecha disponible.

date	accum_cases_report	new_cases_report	accum_deaths_report	new_deaths_report	accum_hospitalized_report	new_hospitalized_report	accum_suspect_report	new_suspect_report
20210613	2454176	0	230150	0	459470	0	434331	0

Figura X. Formato original datos COVID-19 a nivel nacional, última fecha disponible

En cuanto al último reporte de COVID-19 este es descargado por el pipeline *covid_pipeline* y, al momento de la elaboración de esta documentación, el

DataFrame inicial contenía 7.228.942 filas por 40 columnas. Para mayor información ver sección 4.2.

Transformación y limpieza

Primero, se estandarizan los ID de *covid_positive* donde 1 indica si el paciente dio positivo al COVID-19, 2 si es negativo, y 3 si no se ha obtenido resultado. Luego se agregan los datos, agrupando por casos positivos, negativos y no reportados, obteniendo la suma de casos sospechosos, casos hospitalizados y casos totales, esto en *DataFrames* separados. Posteriormente se vuelve a juntar toda la información en un solo *DataFrame*.

Luego se procede a reemplazar los datos vacíos o nulos en las columnas de tiempo y municipio, para agregar consistencia y validez a los datos.

Finalmente, se juntan los datos del último reporte del gobierno mexicano con los datos ya pre procesados obtenidos desde la API, para luego obtener agregaciones de este *DataFrame*, tales como los casos promedio por día, acumulación total de casos, porcentaje de nuevos casos, entre otros.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contiene 473 filas y 45 columnas. La estructura final de la tabla se presenta a continuación:

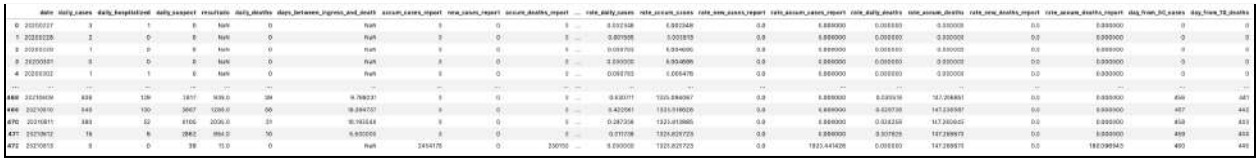


Figura X. Formato final tabla de datos y estadísticas COVID-19 a nivel nacional

4.4. Gobmx_covid_stats_state

Descripción general y ejecución

El pipeline de estadísticas de la situación mexicana frente al COVID-19, *covid_stats_state.py*, es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del cubo *gobmx_covid* para obtener diferentes métricas asociadas a la evolución de la pandemia a nivel estatal.

Estos datos ofrecen información respecto de la evolución de los casos de positivos, fallecidos, sospechosos y hospitalizados, informando los valores diarios, acumulados, promedios móviles a 7 días y tasas de contagios sobre el total de la población.

Los datos son obtenidos desde la API de tesseract construida por Datawheel para el sitio DataMéxico. También se usan los datos del último reporte COVID-19 entregado por el gobierno mexicano en su repositorio. Pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Cabe destacar que este pipeline es necesario para obtener las métricas correctas a diferentes niveles geográficos, en este caso, para cada entidad federativa. Al momento de ingestar los datos, cuando hay diferentes niveles geográficos estos se agrupan desde los niveles más pequeños a los niveles superiores en sumas o promedios simples. En el caso de las estadísticas del COVID-19 se calcula, por ejemplo, el promedio móvil a 7 días, medida que al ser agregada a diferentes niveles geográficos no representaría el valor real de indicador.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline a través de *bamboo-cli* de la siguiente forma:

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_state
```

Si el archivo de datos ya se encuentra descargado, entonces se puede pasar como parámetro del pipeline el directorio donde está localizado el archivo.

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_state
--file_path='/path/to/file'
```

Lectura de datos

Los datos son extraídos desde la API usando la librería *requests* de python, la que permite extraer directamente los datos desde la API y recibirlos en formato JSON, estos se leen y almacenan en un *DataFrame* de la librería pandas. Son 4 las llamadas a la API que se realizan para obtener el *DataFrame* inicial. Una vez recibida cada llamada, se guarda en un *DataFrame* y se junta con los datos anteriores.

Al momento de la elaboración de esta documentación, el *DataFrame* inicial contiene 32 filas por 10 columnas. La siguiente imagen muestra las primeras filas y algunas columnas del *DataFrame* inicial.

	date	ent_id	accum_cases_report	new_cases_report	accum_deaths_report	new_deaths_report	accum_hospitalized_report	new_hospitalized_report	accum_suspect_report	new_suspect_report
0	20210613	1	26649	0	2446	0	5403	0	10310	0
1	20210613	2	49997	0	8617	0	15239	0	33664	0
2	20210613	3	34180	0	1461	0	2960	0	3036	0
3	20210613	4	10674	0	1243	0	2549	0	4639	0
4	20210613	5	69220	0	6365	0	11967	0	14004	0
5	20210613	6	12115	0	1204	0	2636	0	2120	0
6	20210613	7	11827	0	1647	0	3542	0	38841	0
7	20210613	8	57224	0	7450	0	13919	0	5121	0

Figura X. Formato original datos COVID-19 a nivel estatal, última fecha disponible

En cuanto al último reporte de COVID-19 este es descargado por el pipeline *covid_pipeline* y, al momento de la elaboración de esta documentación, el *DataFrame* inicial contenía 7.228.942 filas por 40 columnas. Para mayor información ver sección 4.2.

Transformación y limpieza

Primero, se estandarizan los ID de *covid_positive* donde 1 indica si el paciente dio positivo al COVID-19, 2 si es negativo, y 3 si no se ha obtenido resultado. Luego se agregan los datos, agrupando por casos positivos, negativos y no reportados, obteniendo la suma de casos sospechosos, casos hospitalizados y casos totales, esto en *DataFrames* separados. Posteriormente se vuelve a juntar toda la información en un solo *DataFrame*.

Luego se procede a reemplazar los datos vacíos o nulos en las columnas de tiempo y entidad federativa, para agregar consistencia y validez a los datos.

Finalmente, se juntan los datos del último reporte del gobierno mexicano con los datos ya pre procesados obtenidos desde la API, para luego obtener agregaciones de este *DataFrame*, tales como los casos promedio por día, acumulación total de casos, porcentaje de nuevos casos, entre otros.

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 14.713 filas y 46 columnas. La estructura final de la tabla se presenta a continuación:

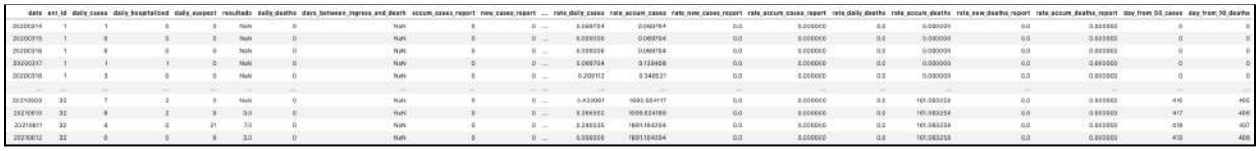


Figura X. Formato final tabla de datos y estadísticas COVID-19 a nivel estatal

4.5. Gobmx_covid_stats_metroarea

Descripción general y ejecución

El pipeline de estadísticas de la situación mexicana frente al COVID-19, *covid_stats_metroarea.py*, es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del cubo *gobmx_covid* para obtener diferentes métricas asociadas a la evolución de la pandemia a nivel de área metropolitana..

Estos datos ofrecen información respecto de la evolución de los casos de positivos, fallecidos, sospechosos y hospitalizados, informando los valores diarios, acumulados, promedios móviles a 7 días y tasas de contagios sobre el total de la población.

Los datos son obtenidos del último reporte COVID-19 entregado por el gobierno mexicano en su repositorio. Pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Cabe destacar que este pipeline es necesario para obtener las métricas correctas a diferentes niveles geográficos, en este caso, para cada área metropolitana. Al momento de ingestar los datos, cuando hay diferentes niveles geográficos estos se agrupan desde los niveles más pequeños a los niveles superiores en sumas o promedios simples. En el caso de las estadísticas del COVID-19 se calcula, por ejemplo, el promedio móvil a 7 días, medida que al ser agregada a diferentes niveles geográficos no representaría el valor real del indicador.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

(bamboo-cli)

```
~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_metroarea
```

Si el archivo de datos ya se encuentra descargado, entonces se puede pasar como parámetro del pipeline el directorio donde está localizado el archivo.

```
(bamboo-cli) ~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_metroarea --file_path='/path/to/file'
```

Lectura de datos

En el presente ETL son utilizados tres conjuntos de datos extraídos desde diferentes fuentes. Primero que todo, se importan datos de población a nivel de área metropolitana, desde el cubo de datos *population_projection*, y son almacenados un *DataFrame*. Segundo, se lee un conjunto de datos que posee los id para las áreas metropolitanas. Tercero, se agregan los datos del último reporte de COVID-19, los cuales son descargados a través del pipeline *covid_pipeline*, y son puestos a disposición para ejecutar el procesamiento de los datos.

A continuación se visualiza la estructura inicial de los tres conjuntos de datos mencionados anteriormente. Los datos de población a nivel de área metropolitana poseen una estructura inicial de 74 filas y 3 columnas.

	Metro Area ID	Metro Area	Projected Population
0	99101	Aguascalientes	1143729
1	99201	Ensenada	536143
2	99202	Mexicali	1087478
3	99203	Tijuana	2011247
4	99301	La Paz	301961
...	...	...	...
69	993006	Poza Rica	570057
70	993007	Veracruz	956463
71	993008	Xalapa	817501
72	993101	Mérida	1237697
73	993201	Zacatecas-Guadalupe	399055

Figura X. Formato original datos proyección de población por zona metropolitana

Los datos de id para áreas metropolitanas, poseen una estructura inicial de 417 filas y 3 columnas.

	zm_id	NOM_ZM	mun_id
0	99101	Aguascalientes	1001
1	99101	Aguascalientes	1005
2	99101	Aguascalientes	1011
3	99201	Ensenada	2001
4	99202	Mexicali	2002
...	...	...	...
412	993201	Zacatecas-Guadalupe	32017
413	993201	Zacatecas-Guadalupe	32032
414	993201	Zacatecas-Guadalupe	32050
415	993201	Zacatecas-Guadalupe	32056
416	993201	Zacatecas-Guadalupe	32057

Figura X. Formato original datos municipios pertenecientes a cada zona metropolitana

En cuanto al último reporte de COVID-19 este es descargado por el pipeline *covid_pipeline* y, al momento de la elaboración de esta documentación, el *DataFrame* inicial contenía 7.228.942 filas por 40 columnas. Para mayor información ver sección 4.2.

Limpieza y Transformación

Primero, se estandarizan los ID de *covid_positive* donde 1 indica si el paciente dio positivo al COVID-19, 2 si es negativo, y 3 si no se ha obtenido resultado. Luego se agregan los datos, agrupando por casos positivos, negativos y no reportados, obteniendo la suma de casos sospechosos, casos hospitalizados y casos totales, esto en *DataFrames* separados. Posteriormente se vuelve a juntar toda la información en un solo *DataFrame*.

Segundo, se procede a reemplazar los datos vacíos o nulos en las columnas de tiempo y área metropolitana, para agregar consistencia y validez a los datos.

Finalmente, se obtienen las estadísticas tales como casos promedio por día, acumulación total de casos, porcentaje de nuevos casos, entre otros..

Al momento en el que se elaboró esta documentación, el *DataFrame* final contiene 33.257 filas y 20 columnas. La estructura final de la tabla se presenta a continuación:

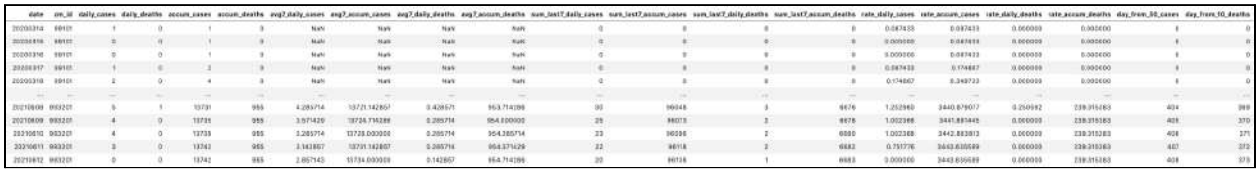


Figura X. Formato final tabla de datos y estadísticas COVID-19 a nivel de zona metropolitana

4.6. Gobmx_covid_stats_mun

Descripción general y ejecución

El pipeline de estadísticas de la situación mexicana frente al COVID-19, *covid_stats_mun.py*, es una herramienta de ETL (Extracción, Transformación y Carga de datos) que procesa los datos del cubo *gobmx_covid* para obtener diferentes métricas asociadas a la evolución de la pandemia a nivel municipal.

Estos datos ofrecen información respecto de la evolución de los casos positivos, fallecidos, sospechosos y hospitalizados, informando los valores diarios, acumulados, promedios móviles a 7 días y tasas de contagios sobre el total de la población.

Los datos son obtenidos desde la API de tesseract construida por Datawheel para el sitio DataMéxico. También se usan los datos del último reporte COVID-19 entregado por el gobierno mexicano en su repositorio. Pasan por un proceso de limpieza y transformación, para luego ser ingestados en una base de datos Clickhouse siguiendo un modelo relacional.

Cabe destacar que este pipeline es necesario para obtener las métricas correctas a diferentes niveles geográficos, en este caso, para cada municipio. Al momento de ingestar los datos, cuando hay diferentes niveles geográficos estos se agrupan desde los niveles más pequeños a los niveles superiores en sumas o promedios simples. En el caso de las estadísticas del COVID-19 se calcula, por ejemplo, el promedio móvil a 7 días, medida que al ser agregada a diferentes niveles geográficos no representaría el valor real del indicador.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías necesarias:

```
~/data-etl/$ source venv/bin/activate
```

Luego, se puede ejecutar el pipeline de la siguiente forma:

```
(bamboo-cli) ~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_mun
```

Si el archivo de datos ya se encuentra descargado, entonces se puede pasar como parámetro del pipeline el directorio donde está localizado el archivo.

```
(bamboo-cli) ~/data-etl/etl/covid/$ bamboo-cli --folder . --entry covid_stats_mun --file_path='/path/to/file'
```

Lectura de datos

Los datos son extraídos desde la API usando la librería *requests* de python, la que permite extraer directamente los datos desde la API y recibirlos en formato JSON, estos se leen y almacenan en un *DataFrame* de la librería pandas. Son 4 las llamadas a la API que se realizan para obtener el *DataFrame* inicial. Una vez recibida cada llamada, se guarda en un *DataFrame* y se junta con los datos anteriores. Cabe destacar que las llamadas obtienen datos que toman relación con: casos positivos, fallecidos, hospitalizados, y sospechosos.

Al momento de la elaboración de esta documentación, el *DataFrame* inicial contenía 2.400 filas por 10 columnas. La siguiente imagen muestra algunas filas y columnas del *DataFrame* inicial.

date	mun_id	accum_cases_report	new_cases_report	accum_deaths_report	new_deaths_report	accum_hospitalized_report	new_hospitalized_report	accum_suspect_report	new_suspect_report
20210613	1001	22037.0	0.0	2133.0	0.0	4577.0	0.0	9162.0	0.0
20210613	1002	464.0	0.0	36.0	0.0	94.0	0.0	38.0	0.0
20210613	1003	982.0	0.0	26.0	0.0	95.0	0.0	54.0	0.0
20210613	1004	129.0	0.0	12.0	0.0	31.0	0.0	19.0	0.0
20210613	1005	778.0	0.0	62.0	0.0	155.0	0.0	364.0	0.0
...	...	...	...	...	...	...	...	...	...
20210613	20436	NaN	NaN	NaN	NaN	NaN	NaN	1.0	0.0
20210613	20480	NaN	NaN	NaN	NaN	NaN	NaN	2.0	0.0
20210613	20491	NaN	NaN	NaN	NaN	NaN	NaN	1.0	0.0
20210613	20514	NaN	NaN	NaN	NaN	NaN	NaN	1.0	0.0
20210613	20541	NaN	NaN	NaN	NaN	NaN	NaN	1.0	0.0

Figura X. Formato original datos COVID-19 a nivel municipal, última fecha disponible

En cuanto al último reporte de COVID-19 este es descargado por el pipeline *covid_pipeline* y, al momento de la elaboración de esta documentación, el

DataFrame inicial contenía 7.228.942 filas por 40 columnas. Para mayor información ver sección 4.2.

2. Transformación y limpieza

Primero, se estandarizan los ID de *covid_positive* donde 1 indica si el paciente dio positivo al COVID-19, 2 si es negativo, y 3 si no se ha obtenido resultado. Luego se agregan los datos, agrupando por casos positivos, negativos y no reportados, obteniendo la suma de casos sospechosos, casos hospitalizados y casos totales, esto en *DataFrames* separados. Posteriormente se vuelve a juntar toda la información en un solo *DataFrame*.

Luego se procede a reemplazar los datos vacíos o nulos en las columnas de tiempo y municipio, para agregar consistencia y validez a los datos.

Finalmente, se juntan los datos del último reporte del gobierno mexicano con los datos ya pre procesados obtenidos desde la API, para luego obtener agregaciones de este *DataFrame*, tales como los casos promedio por día, acumulación total de casos, porcentaje de nuevos casos, entre otros..

Al momento en el que se elaboró esta documentación, el *DataFrame* final contenía 909.735 filas y 46 columnas. La estructura final de la tabla se presenta a continuación:

Figura X. Formato final tabla de datos y estadísticas COVID-19 a nivel municipal

4.7. imss

Descripción general y ejecución

El pipeline *imss_pipeline.py* procesa los datos facilitados por el Instituto Mexicano del Seguro Social (IMSS) que toman relación con los reportes de Salud. Los datos son descargados desde la plataforma del Instituto Mexicano del Seguro Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/imss/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/imss/$ python imss_pipeline.py`

(Bamboo CLI) `~/data_etl/etl/imss/$ bamboo-cli --folder . --entry imss_pipeline`

Lectura de datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y se identifican 4.656.548 filas y 29 columnas. A continuación, se visualiza la estructura inicial de los datos:

cve_delegacion	cve_subdelegacion	cve_entidad	cve_municipio	sector_economico_1	sector_economico_2	sector_economico_4	tamaño_patron	sexo	rango_eda
1	1	1	A01	NaN	NaN	NaN	NaN	1	E
1	1	1	A01	NaN	NaN	NaN	NaN	1	E
1	1	1	A01	NaN	NaN	NaN	NaN	1	E
1	1	1	A01	NaN	NaN	NaN	NaN	1	E
1	1	1	A01	NaN	NaN	NaN	NaN	1	E

Figura X. Formato original datos imss

Transformación y limpieza

El procesamiento de los datos sigue el proceso detallado a continuación:

- Se utiliza el parámetro *chunksize* de la función *read_csv* de Pandas para leer los datos en subconjuntos, debido al gran espacio requerido por el conjunto de datos.
- Modificación del nombre de las columnas.
- Eliminación de columnas que no serán utilizadas en el proceso de ETL.
- Renombre de columnas.
- Relleno de valores *nan* con el valor 0.
- Creación de la columna *count* con el valor 1 en toda su extensión. Dicha técnica es utilizada para generar, posteriormente, agrupaciones en los datos.
- Reemplazo de *caracteres* por espacios en blanco en las columnas: *age_range, uma_range, pattern_size* y *salary_range*. Esto se realiza con el motivo de dejar solamente valores numéricos.
- Agrupación de datos para tener la data mejor organizada.
- Creación de la columna *salary*. Dicha columna se crea a partir de la división de la variable *masa_sal_ta* por la columna *count*. De esta forma se obtiene el salario individual.

Para continuar con el procesamiento de datos, fue necesaria la lectura de dos nuevos archivos. Con respecto al primero:

- Lectura de un archivo Google Spreadsheet que contiene los municipios y sus *id's* otorgados por el IMSS.
- Renombre de columnas y reemplazo de nombres para definiciones de mayor claridad.

Con respecto al segundo:

- Lectura de archivo que contiene los municipios y sus *id*. Ahora bien, dichos id difieren a los anteriores ya que son *id's* de mayor claridad otorgados por Datawheel.

Posterior a ello, se procede a reemplazar el *id* original por los otorgados por Datawheel.

- Creación de la columna *month_id*. Dicha columna se crea mediante la agregación del año y el mes en estudio.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 4.272.221 filas y 18 columnas . A continuación puede ser visualizada su estructura de salida:

	age_range	mun_id	salary_range	level_4_id	sex	pattern_size	uma_range	asegurados	non_workers	ta	teu	tec	tpu	tpc	masa_sal_ta	count	s
0	1	9999	0	0	1	0	0	2008	2008	0	0	0	0	0	0.00	10	
1	1	9999	0	0	2	0	0	1836	1836	0	0	0	0	0	0.00	10	
2	1	9999	2	3708	1	3	2	1	0	1	0	0	1	0	112.00	1	1
3	1	9999	2	4101	1	3	2	2	0	2	2	0	0	0	298.62	1	2
4	1	9999	2	4102	1	4	2	1	0	1	0	0	1	0	126.16	1	1

Figura X. Formato final tabla de datos imss

5. Créditos

5.1. Imss_credits

Descripción General y Ejecución

Este pipeline *imss_credits.py* procesa los datos a nivel estatal y municipal de los créditos otorgados por el Instituto Mexicano de Seguro Social (IMSS) como ayuda económica a raíz de la pandemia causada por el COVID-19. Los datos son proporcionados de manera interna por la Secretaría de Economía de México a Datawheel, y estos son almacenados de forma segura en Google Storage. Desde Google Storage se realiza la descarga de datos, para posteriormente ser sometidos a un proceso de limpieza y transformación antes de ser ingestados en una base de datos de Clickhouse

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/credits/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/credits/$ python imss_credits.py`

(bamboo-cli) `~/data_etl/etl/credits/$ bamboo-cli --folder . --entry imss_credits`

Descarga de Datos

En este paso se obtienen los archivos en formato .csv de los créditos otorgados a nivel estatal y municipal.

clave	ENTIDAD	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
1	Aguascalientes	H	Física	De 1 a 10 empleados	19 años o menos	3
1	Aguascalientes	H	Física	De 1 a 10 empleados	20 a 29 años	87
1	Aguascalientes	H	Física	De 1 a 10 empleados	30 a 39 años	291
1	Aguascalientes	H	Física	De 1 a 10 empleados	40 a 49 años	336
1	Aguascalientes	H	Física	De 1 a 10 empleados	50 a 59 años	272
...	...	...	...	...	...	...
32	Zacatecas	C	Moral	De 1 a 10 empleados	C	152
32	Zacatecas	C	Moral	De 1 a 10 empleados	C	1
32	Zacatecas	C	Moral	De 11 a 20 empleados	C	26
32	Zacatecas	C	Moral	De 21 a 50 empleados	C	9
32	Zacatecas	C	Moral	Más de 50 empleados	C	2

Figura X. Formato original datos a nivel estatal de créditos modalidad IMSS-Patrón

clave	ENTIDAD	MUNICIPIO	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
1001	Aguascalientes	Aguascalientes	H	Física	De 1 a 10 empleados	19 años o menos	2
1001	Aguascalientes	Aguascalientes	H	Física	De 1 a 10 empleados	20 a 29 años	73
1001	Aguascalientes	Aguascalientes	H	Física	De 11 a 20 empleados	20 a 29 años	2
1001	Aguascalientes	Aguascalientes	H	Física	De 1 a 10 empleados	30 a 39 años	259
1001	Aguascalientes	Aguascalientes	H	Física	De 11 a 20 empleados	30 a 39 años	8
...	...	...	...	...	...	...	...
20072	Oaxaca	San José del Progreso	C	Moral	De 1 a 10 empleados	C	1
20343	Oaxaca	San Sebastián Abasolo	C	Moral	De 1 a 10 empleados	C	1
20405	Oaxaca	Villa de Chilapa de Díaz	C	Moral	De 1 a 10 empleados	C	1
31065	Yucatán	San Felipe	C	Moral	De 1 a 10 empleados	C	2
31072	Yucatán	Suma	C	Moral	De 1 a 10 empleados	C	1

Figura X. Formato original datos a nivel municipal de créditos modalidad IMSS-Patrón

Cabe destacar que se entregan archivos por separado para entidades y municipios debido a la confidencialidad de los datos, ya que la suma de los valores municipales no necesariamente corresponde a la Entidad Federativa que las agrupa.

Lectura

En este paso se lee el archivo con datos municipales y el archivo con datos estatales, luego se modifica el nombre de las columnas en el *DataFrame* y se añade la columna *Level* que indica el nivel geográfico de los datos: Estatal o Municipal.

Transformación

En el paso de transformación se realizan los siguientes cambios:

- Filtrar los valores confidenciales, omitiendo las entradas que marcan C en la columna *count*.
- Las columnas *sex, person_type, company_size y age_range* son mapeadas a valores numéricos.
- Posteriormente se llena la información de municipalidades específicas limpiando los datos faltantes. Se realiza un proceso similar con el nivel estatal.

Al finalizar los pasos de transformación se obtiene la estructura final de la tabla que se presenta a continuación:

	ent_id	mun_id	level	sex	person_type	company_size	age_range	count
0	0	1001	Municipality	1	1	1	5	87
1	0	1001	Municipality	1	1	1	6	265
2	0	1001	Municipality	1	1	1	7	272
3	0	1001	Municipality	1	1	1	8	230
4	0	1001	Municipality	1	1	1	9	177
...	...	...	...	...	...	...	...	...
4672	32	32999	State	2	1	1	7	176
4673	32	32999	State	2	1	1	8	159
4674	32	32999	State	2	1	1	9	163
4675	32	32999	State	2	2	1	12	13
4676	32	32999	State	2	2	1	13	21

Figura X. Formato final tabla de datos créditos modalidad IMSS-Patrón

5.2. Household_credits

Descripción General y Ejecución

El pipeline *households.py* procesa datos a nivel estatal y municipal de los créditos otorgados por el Instituto Mexicano de Seguro Social (IMSS) a los trabajadores del hogar. Los datos son proporcionados de manera interna por la Secretaría de Economía de México a Datawheel, y estos son almacenados de forma segura en Google Storage. Desde Google Storage se realiza la descarga de datos, para posteriormente ser sometidos a un proceso de limpieza y transformación antes de ser ingestados en una base de datos de Clickhouse.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/credits/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/credits/$ python households.py`

(Bamboo CLI) `~/data_etl/etl/credits/$ bamboo-cli --folder . --entry households`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y en archivos separados para Entidades y Municipios. El archivo de Entidades contaba con 311 filas y 7 columnas, y su formato inicial se muestra a continuación:

clave	ENTIDAD	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
1	Aguascalientes	H	Física	C	20 a 29 años	6
1	Aguascalientes	H	Física	C	30 a 39 años	18
1	Aguascalientes	H	Física	C	40 a 49 años	51
1	Aguascalientes	H	Física	C	50 a 59 años	64
1	Aguascalientes	H	Física	C	60 años o más	92
...	...	...	...	...	...	...
32	Zacatecas	M	Física	C	20 a 29 años	2
32	Zacatecas	M	Física	C	30 a 39 años	2
32	Zacatecas	M	Física	C	40 a 49 años	5
32	Zacatecas	M	Física	C	50 a 59 años	8
32	Zacatecas	M	Física	C	60 años o más	9

Figura X. Formato original datos a nivel estatal de créditos modalidad Trabajadores del Hogar

Al momento de escribir esta documentación, el archivo con valores por municipios contaba con 2.334 filas y 8 columnas. Su formato inicial se muestra a continuación:

clave	ENTIDAD	MUNICIPIO	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
1001	Aguascalientes	Aguascalientes	H	Física	c	20 a 29 años	5
1001	Aguascalientes	Aguascalientes	H	Física	c	30 a 39 años	12
1001	Aguascalientes	Aguascalientes	H	Física	c	40 a 49 años	41
1001	Aguascalientes	Aguascalientes	H	Física	c	50 a 59 años	54
1001	Aguascalientes	Aguascalientes	H	Física	c	60 años o más	74
...	...	...	...	...	...	...	...
32056	Zacatecas	Zacatecas	M	Física	c	20 a 29 años	1
32056	Zacatecas	Zacatecas	M	Física	c	30 a 39 años	1
32056	Zacatecas	Zacatecas	M	Física	c	40 a 49 años	2
32056	Zacatecas	Zacatecas	M	Física	c	50 a 59 años	5
32056	Zacatecas	Zacatecas	M	Física	c	60 años o más	4

Figura X. Formato original datos a nivel municipal de créditos modalidad Trabajadores del Hogar

Cabe destacar que se entregan archivos por separado para entidades y municipios debido a la confidencialidad de los datos, ya que la suma de los valores municipales no necesariamente corresponde a la Entidad Federativa que las agrupa.

Lectura

En esta etapa se procede inicialmente a modificar el nombres de las columnas en el *DataFrame* y se añade la columna *Level* que indica el nivel geográfico de los datos: Estatal o Municipal.

Transformación

En esta etapa se sigue el siguiente proceso:

- Filtrar los valores confidenciales, omitiendo las entradas que marcan c en la columna *count*.
- Se reemplazan los valores en las columnas *sex, person_type, y age_range* por valores numéricos. Esto se realiza con la ayuda de diccionarios que son definidos en el script *shared.py*.
- Se completan los valores faltantes en las columnas que describen el nivel estatal y municipal.
- Se reorganizan las columnas en el *DataFrame*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 2.645 filas y 7 columnas. La estructura final de la tabla se presenta a continuación:

	ent_id	mun_id	level	sex	person_type	age_range	count
0	0	1001	Municipality	1	1	5	5
1	0	1001	Municipality	1	1	6	12
2	0	1001	Municipality	1	1	7	41
3	0	1001	Municipality	1	1	8	54
4	0	1001	Municipality	1	1	9	74
...	...	...	...	...	...	...	...

Figura X. Formato final tabla de datos de créditos modalidad Trabajadores del Hogar

5.3. Wellness_credits

Descripción General y Ejecución

El pipeline *wellness.py* procesa datos a nivel estatal y municipal de los créditos otorgados por el Instituto Mexicano de Seguro Social (IMSS) para el financiamiento de micronegocios. Dichos créditos pertenecen a la modalidad de Bienestar.

Los datos son proporcionados de manera interna por la Secretaría de Economía de México a Datawheel, y estos son almacenados de forma segura en Google Storage. Desde Google Storage se realiza la descarga de datos, para posteriormente ser sometidos a un proceso de limpieza y transformación antes de ser ingestados en una base de datos de Clickhouse.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/credits/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/credits/$ python wellness.py`

(Bamboo CLI) `~/data_etl/etl/credits/$ bamboo-cli --folder . --entry wellness`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y al momento de escribir esta documentación el archivo con datos a nivel estatal contaba con 380 filas y 7 columnas. El formato inicial se muestra a continuación:

CLAVE	ENTIDAD	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
1	Aguascalientes	H	Física	C	19 años o menos	2
1	Aguascalientes	H	Física	C	20 a 29 años	508
1	Aguascalientes	H	Física	C	30 a 39 años	916
1	Aguascalientes	H	Física	C	40 a 49 años	1142
1	Aguascalientes	H	Física	C	50 a 59 años	1034
...	...	...	...	...	...	...
32	Zacatecas	M	Física	C	20 a 29 años	606
32	Zacatecas	M	Física	C	30 a 39 años	1450
32	Zacatecas	M	Física	C	40 a 49 años	1646
32	Zacatecas	M	Física	C	50 a 59 años	1312
32	Zacatecas	M	Física	C	60 años o más	655

Figura X. Formato original datos a nivel estatal de créditos modalidad Bienestar

Por su parte, el archivo con datos a nivel municipal contaba 12.855 filas y 7 columnas. Su formato se observa en la figura siguiente:

Clave	entidad	municipio	sexo	tipo_persona	rango_edad_antigüedad	conteo_anonimizado
1001	Aguascalientes	Aguascalientes	H	Física	19 años o menos	2
1001	Aguascalientes	Aguascalientes	H	Física	20 a 29 años	452
1001	Aguascalientes	Aguascalientes	H	Física	30 a 39 años	819
1001	Aguascalientes	Aguascalientes	H	Física	40 a 49 años	1052
1001	Aguascalientes	Aguascalientes	H	Física	50 a 59 años	960
...	...	...	...	...	...	...
32035	Zacatecas	Noria de Ángeles	H	Física	20 a 29 años	2
20404	Oaxaca	Santa María Chachoápam	M	Física	40 a 49 años	1
21198	Puebla	Xicotlán	M	Física	30 a 39 años	1
21201	Puebla	Xochiltepec	M	Física	30 a 39 años	1
30060	Veracruz de Ignacio de la Llave	Chinampa de Gorostiza	M	Física	60 años o más	1

Figura X. Formato original datos a nivel municipal de créditos modalidad Bienestar

Cabe destacar que se entregan archivos por separado para entidades y municipios debido a la confidencialidad de los datos, ya que la suma de los valores municipales no necesariamente corresponde a la Entidad Federativa que las agrupa.

Lectura

En esta etapa se procede inicialmente a modificar el nombres de las columnas en el *DataFrame* y se añade la columna *Level* que indica el nivel geográfico de los datos: Estatal o Municipal.

Transformación

Este proceso puede ser definido por las siguientes etapas:

- Filtrar valores confidenciales. Para ello, se omiten las entradas que marcan **c** en la columna *count*, y también, se reemplazan las entradas que marcan **c** en las columnas *sex y age_range* por el valor 0.
- Se reemplazan las entradas en las columnas *person_type, sex, y age_range* por valores numéricos. Dicho reemplazo se hace con la ayuda de diccionarios definidos en el script *shared.py*.
- Se completan los valores faltantes en las columnas que describen el nivel estatal y municipal. Además, se cambian los nombres de estados y municipios por sus id's correspondientes.
- Finalmente, se reorganizan las columnas en el *DataFrame*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 13.489 filas y 7 columnas. La estructura final de la tabla se presenta a continuación:

	ent_id	mun_id	sex	person_type	age_range	count	level
0	1	1001	1	1	4	2	Municipality
1	1	1001	1	1	5	448	Municipality
2	1	1001	1	1	6	811	Municipality
3	1	1001	1	1	7	1035	Municipality
4	1	1001	1	1	8	952	Municipality

Figura X. Formato final tabla de datos de créditos modalidad Bienestar

5.4. Female_credits

Descripción General y Ejecución

El pipeline *females.py* procesa datos a nivel municipal de los créditos otorgados por la Secretaría de Economía de México a empresarias de pequeñas empresas. Dicho programa se denomina Mujeres Solidarias. Los datos son proporcionados de manera interna por la Secretaría de Economía de México a Datawheel, y estos son almacenados en Google Storage. Desde Google Storage se realiza la descarga de datos, para posteriormente ser sometidos a un proceso de limpieza y transformación antes de ser ingestados en una base de datos de *Clickhouse*.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/credits/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/credits/$ python females.py`

(Bamboo CLI) `~/data_etl/etl/credits/$ bamboo-cli --folder . --entry females`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, en cuanto a su estructura, posee 2.296 filas y 7 columnas. A continuación se presenta su estructura inicial:

	Clave municipal	Nom Ent	Nom mun	sexo	tipo_persona	rango_edad_antigüedad	conteo_anonimizado
0	1001	Aguascalientes	Aguascalientes	M	Física	19 años o menos	1
1	1001	Aguascalientes	Aguascalientes	M	Física	20 a 29 años	39
2	1001	Aguascalientes	Aguascalientes	M	Física	30 a 39 años	50
3	1001	Aguascalientes	Aguascalientes	M	Física	40 a 49 años	41
4	1001	Aguascalientes	Aguascalientes	M	Física	50 a 59 años	18

Figura X. Formato original datos a nivel municipal de créditos modalidad Mujeres Solidarias

Lectura y Transformación

En esta etapa se sigue el siguiente proceso:

- Lectura de datos. Se utiliza la librería *bamboo-lib*.
- Transformación de nombres de columnas a minúsculas, renombrando además sus nombres al inglés.
- Se reemplazan los valores en las columnas *sex, person_type* y *age_range* por sus *id's*. Esto se realiza con la ayuda de diccionarios que son definidos en el *script shared.py*.
- Se reorganizan las columnas en el *DataFrame*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 2.296 filas y 7 columnas. A continuación puede ser visualizada su estructura:

	mun_id	sex	person_type	age_range	count
0	1001	2	1	4	1
1	1001	2	1	5	39
2	1001	2	1	6	50
3	1001	2	1	7	41
4	1001	2	1	8	18

Figura X. Formato final tabla de datos de créditos modalidad Mujeres Solidarias

5.5. accomplished_companies

Descripción General y Ejecución

El pipeline *accomplished_companies.py* procesa datos a nivel municipal de los créditos otorgados por la Secretaría de Economía de México a empresas que han demostrado ser solidarias con sus trabajadores. Dicho programa se denomina Empresas Cumplidas. Los datos son proporcionados de manera interna por la Secretaría de Economía de México a Datawheel, y estos son almacenados en Google Storage. Desde Google Storage se realiza la descarga de datos, para posteriormente ser sometidos a un proceso de limpieza y transformación antes de ser ingestados en una base de datos de *Clickhouse*.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/credits/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

(Python) `~/data_etl/etl/credits/$ python accomplished_companies.py`

(Bamboo-cli) `~/data_etl/etl/credits/$ bamboo-cli --folder . --entry accomplished_companies`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, en cuanto a su estructura, posee 4.222 filas y 7 columnas. A continuación se presenta su estructura inicial:

	Clave municipal	Nom Ent	Nom mun	sexo	tipo_persona	rango_edad_antigüedad	conteo_anonimizado
0	1001	Aguascalientes	Aguascalientes	H	Física	20 a 29 años	36
1	1001	Aguascalientes	Aguascalientes	H	Física	30 a 39 años	52
2	1001	Aguascalientes	Aguascalientes	H	Física	40 a 49 años	31
3	1001	Aguascalientes	Aguascalientes	H	Física	50 a 59 años	15
4	1001	Aguascalientes	Aguascalientes	H	Física	60 años o más	11

Figura X. Formato original datos de créditos modalidad Empresas Cumplidas

Lectura y Transformación

En esta etapa se sigue el siguiente proceso:

- Lectura de datos. Se utiliza la librería *bamboo-lib*.
- Transformación de nombres de columnas a minúsculas, renombrando además sus nombres al inglés.
- Reemplazo de valores en las variables *sex*, *person_type*, y *age_range* por sus *id's*. Esto se realiza con la ayuda de diccionarios que son definidos en el *script shared.py*.
- Eliminación de variables no utilizadas: *nom_ent* y *nom_mun*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 4.222 filas y 5 columnas. A continuación puede ser visualizada su estructura:

	mun_id	sex	person_type	age_range	count
0	1001	1	1	5	36
1	1001	1	1	6	52
2	1001	1	1	7	31
3	1001	1	1	8	15
4	1001	1	1	9	11

Figura X. Formato final tabla de datos de créditos modalidad Empresas Cumplidas

6. Comercio Exterior

6.1. economy_foreign_trade_nat/state/mun

Descripción General y Ejecución

El pipeline *foreign_trade_pipeline* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México relacionados con el Comercio Exterior. Los datos son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Cabe destacar que los datos poseen diferentes niveles de agregación. Entre ellos, están diferentes periodos de tiempo y profundidad en la clasificación bajo el Sistema Armonizado (HS).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_trade/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_trade/$ python run.py
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y los datos, entregados por la Secretaría de Economía, están distribuidos en múltiples archivos que proporcionan información para diferentes periodos de tiempo, niveles geográficos y códigos de productos: los archivos se encuentran organizados por nivel nacional, estatal y municipal; cada uno de estos niveles

contiene los valores organizados por código de productos a dos, cuatro y seis dígitos y estos, a su vez, se encuentran organizados en archivos anuales y mensuales. Por ejemplo, a continuación se muestra un extracto de la data inicial para un archivo de marzo 2020, a nivel municipal y desagregado por productos a 2 dígitos, donde se identifican 15.676 filas y 7 columnas. A continuación, se visualiza la estructura inicial de los datos:

Date	Product_2D	Trade_flow	Foreign_Destination_Origin	Value	Firms	unanonymized_value
2020	33	1	AFG	546.8700000000001	2611	546.87
2020	33	1	AIA	9.64	2611	9.64
2020	33	1	ALA	20.16	2611	20.16
2020	33	1	ALB	6191.88	2611	6191.88
2020	33	1	AND	47.45	2611	47.45
...	...	...	...	...	...	...
2020	97	2	PRI	9536.42	94	9536.42
2020	97	2	SWE	112.13	94	112.13
2020	97	2	TWN	11978.38	94	11978.38
2020	97	2	USA	2157933.1799999997	94	2157933.18
2020	97	2	VEN	31.5	94	31.50

Figura X. Formato original datos Comercio Exterior 2020, nacional, HS2

Limpieza y Transformación

El proceso inicia con la lectura de cada archivo y cada uno sigue el siguiente procesamiento:

- Renombre de las columnas.
- Reemplazo de valores confidenciales (C) en la columna *value* por *np.nan*.
- Localización de datos cuyo valor en la columna value es mayor a cero.
- Códigos de países a minúsculas.
- Creación de columnas con clasificación bajo el Sistema Armonizado que no existan en el set de datos. Dichas clasificación pueden ser al nivel: HS2, HS4, y HS6.
- Creación de columnas *month_id* y *year*, las cuales definen el periodo en estudio.

- Agregación de capítulos para todos los niveles de profundidad del Sistema Armonizado. Dicho proceso es realizado con la ayuda de la función *hs6_converter* definida en el script *util.py*.
- Creación de la columna *product_level* que indica el nivel de agregación de los datos según el archivo que contiene los valores por HS2, HS4 o HS6.
- Creación de la columna url, la cual posee la dirección de localización del archivo en Google Storage.
- Homogeneización de algunos códigos de países.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente (), se obtiene un *DataFrame* con 13.753 filas y 13 columnas. La estructura final de la tabla se presenta a continuación:

hs2_id	flow_id	partner_country	value	firms	unanonymized_value	hs6_id	hs4_id	month_id	year	nat_id	product_level	url
633	1	afg	547	2611	547	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
633	1	aia	10	2611	10	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
633	1	ala	20	2611	20	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
633	1	alb	6192	2611	6192	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
633	1	and	47	2611	47	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
...	...	...	...	...	...	...	...	...	...	...	...	...
2197	2	pri	9536	94	9536	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
2197	2	swe	112	94	112	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
2197	2	twn	11978	94	11978	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
2197	2	usa	2157933	94	2157933	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20
2197	2	ven	32	94	32	0	0	0	2020	mex	2	National/HS_2D/Annual/2D_nat_annual20

Figura X. Formato final tabla de datos Comercio Exterior 2020, nacional, HS2

Al finalizar el proceso, todas las tablas con datos nacionales se unen para generar el cubo *economy_foreign_trade_nat*, las tablas con datos estatales generan el cubo *economy_foreign_trade_ent* y, finalmente, las tablas con datos municipales generan el cubo *economy_foreign_trade_mun*.

7. Inversión Extranjera Directa (IED)

7.1. fdi_10_year_country/country_investment/investment

Descripción General y Ejecución

El pipeline *fdi_10.py* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos con relación a la inversión total anual por país de origen, inversión total anual por país de origen y tipo de inversión e inversión total anual por tipo de inversión. Al respecto, son tres los cubos generados a partir del pipeline. Por otro lado, los datos fuente son almacenados en Google Storage.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python fdi_10.py
```

Descarga de datos

Utilizando el módulo de descarga que facilita la librería bamboo-lib se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y su estructura inicial varía de acuerdo al cubo procesado. A continuación, se visualiza la estructura inicial de los datos. Al momento de escribir esta documentación, para el caso de la inversión total anual por país de origen el archivo contenía 1.152 filas y 5 columnas.

Año	País	Monto	Recuento	Monto C
1999	Alemania	685,711369	266	685,7
1999	Argentina	3,638073	91	3,6
1999	Australia	6,945795	21	6,9
...	...	...	...	...
2021	Suecia	-31,64759634	13	-31,6
2021	Suiza	95,06947608	30	95,1
2021	Taiwán	0,61283825	7	0,6

Figura X. Formato original datos IED por año y país

El archivo de inversión total anual por país de origen y tipo de inversión contaba con 2.752 filas y 6 columnas.

Año	País	Inversión	Monto	Recuento	Monto C
1999	Alemania	Cuentas entre compañías	160,518939	70	160,5
1999	Alemania	Nuevas inversiones	308,403768	198	308,4
1999	Alemania	Reinversión de utilidades	216,788662	43	216,8
...	...	...	...	...	...
2021	Suiza	Cuentas entre compañías	41,7244782	20	41,7
2021	Suiza	Nuevas inversiones	22,0718243	9	22,1
2021	Suiza	Reinversión de utilidades	31,2731736	9	31,3

Figura X. Formato original datos IED por año, país y tipo de inversión

Por último, el archivo con datos de inversión total anual por tipo de inversión contaba con 24 filas y 10 columnas.

Año de materialización	Monto cuentas entre compañías	Monto nuevas inversiones	Monto reinversión de utilidades	Recuento cuentas entre compañías	Recuento nuevas inversiones	Recuento reinversión de utilidades	Monto C cuentas entre compañías	Monto C nuevas inversiones	Monto C reinversión de utilidades
1999	4986,21495	6606,924368	2342,739387	2548	6570	586	4.986,2	6.606,9	2.342,7
2000	5637,606776	8704,326873	3905,431695	2551	6332	533	5.637,6	8.704,3	3.905,4
2001	3046,299236	23110,76694	3899,172799	2400	6357	446	3.046,3	23.110,8	3.899,2
...	...	...	...	...	...	...	...	...	...
2019	3016,147825	13228,76851	17998,99319	2336	6229	1007	3.016,1	13.228,8	17.999,0
2020	5098,840477	6622,010814	16064,86588	2441	4159	786	5.098,8	6.622,0	16.064,9
2021	2629,401193	2209,418013	7025,211257	1546	1102	423	2.629,4	2.209,4	7.025,2
Total general	144636,4303	268530,6185	205484,5198	10081	114783	4050	144.636,4	268.530,6	205.484,5

Figura X. Formato original datos IED por año y tipo de inversión

Limpieza y Transformación

Como se ha mencionado anteriormente, dado que existen diferentes fuentes y salidas de datos, los procesos de ETL difieren entre sí. A continuación se presenta, para cada archivo, el proceso de ETL implementado que genera cada cubo de datos.

a) Inversión total anual por país de origen

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas.
- Selección de filas que no corresponden a agregados totales.
- Eliminación de valores confidenciales.
- Reemplazo en columna *country* por el código iso3 de cada país.
- Transformación del *dtype* en las columnas *year, count,* y *value_c*, a enteros.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, obtenemos un *DataFrame* con 1.074 filas y 4 columnas. La estructura final de la tabla se presenta a continuación:

	year	country	count	value_c
0	1999.0	deu	266.0	685.711369
1	1999.0	arg	91.0	3.638073
2	1999.0	aus	21.0	6.945795
3	1999.0	aut	23.0	12.524428
4	1999.0	bel	31.0	-49.370033
...	...	...	...	...
1143	2021.0	cze	3.0	0.015951
1146	2021.0	swe	13.0	-31.647596
1147	2021.0	che	30.0	95.069476
1148	2021.0	twn	7.0	0.612838
1150	2021.0	ven	17.0	1.146261

Figura X. Formato final tabla de datos IED por año y país

b) Inversión total anual por país de origen y tipo de inversión

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas.
- Selección de filas que no corresponden a agregados totales.

- Eliminación de valores confidenciales.
- Reemplazo en columna *country* por el código iso3 de cada país.
- Transformación del *dtype* en las columnas *year, count, y value_c*, a enteros.
- Reemplazo en columna *investment_type* for el id de cada tipo de inversión.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, obtenemos un *DataFrame* con 2.159 filas y 5 columnas. A continuación puede ser visualizada su estructura de salida:

	year	country	investment_type	count	value_c
0	1999.0	deu	1	70.0	160.518939
1	1999.0	deu	2	198.0	308.403768
2	1999.0	deu	3	43.0	216.788662
3	1999.0	arg	1	4.0	-2.594482
4	1999.0	arg	2	87.0	6.211129
...	...	...	...	...	...
2744	2021.0	che	1	20.0	41.724478
2745	2021.0	che	2	9.0	22.071824
2746	2021.0	che	3	9.0	31.273174
2747	2021.0	twn	1	7.0	0.612838

Figura X. Formato final tabla de datos IED por año, país y tipo de inversión

c) Inversión total anual por tipo de inversión

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas por medio de una lista.
- Selección de filas que no corresponden a agregados totales.
- Eliminación de columnas que no serán utilizadas: *value_between_companies*, *value_new_investments*, y *value_re_investments*.
- Transformación del *DataFrame* a formato tidy. Para ello, se crea una nueva columna denominada *investment_type*, la cual posee valores enteros para referirse a los tipos de inversiones (*between_companies*, *new_investments*, y *re_investments*) las cuales anteriormente estaban separadas en columnas individuales.

- Reemplazo en columna *investment_type* por sus id's.

En cuanto a la salida del proceso de ETL, se obtiene un *DataFrame* con 69 filas y 4 columnas. La estructura final de la tabla se presenta a continuación:

	year	count	value_c	investment_type
0	1999.0	2548.0	4986.214950	1
1	2000.0	2551.0	5637.606776	1
2	2001.0	2400.0	3046.299236	1
3	2002.0	2314.0	5868.759710	1
4	2003.0	2285.0	7095.183763	1
...	...	...	...	...
18	2017.0	696.0	11951.846494	3
19	2018.0	820.0	13245.576041	3
20	2019.0	1007.0	17998.993194	3
21	2020.0	786.0	16064.865878	3
22	2021.0	423.0	7025.211257	3

Figura X. Formato final tabla de datos IED por año y tipo de inversión

7.2. fdi_2_state_investment

Descripción General y Ejecución

El pipeline *fdi_2.py* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos con relación a los tipos de inversión por trimestre y entidades federativas. Los datos son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python fdi_2.py
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y su estructura inicial se compone de 2.848 filas y 12 columnas. A continuación, se visualiza la estructura inicial de los datos:

	Entidad federativa	Año	Trimestre	Monto Cuentas entre compañías	Monto Nuevas inversiones	Monto Reinversión de utilidades	Recuento Cuentas entre compañías	Recuento Nuevas inversiones	Recuento Reinversión de utilidades	Monto C cuentas entre compañías	Monto C nuevas inversiones	Monto C reinversión de utilidades
0	Aguascalientes	1999	1	20.645964	7.720677	8.860865	58	46	41	20.645964	7.720677	8.860865
1	Aguascalientes	1999	2	24.018523	-0.168652	6.299668	64	36	22	24.018523	-0.168652	6.299668
2	Aguascalientes	1999	3	16.640456	696.548325	2.034928	67	51	18	16.640456	696.548325	2.034928
3	Aguascalientes	1999	4	15.316704	18.769218	2.159396	67	53	23	15.316704	18.769218	2.159396
4	Aguascalientes	2000	1	16.988831	129.370891	61.172054	70	62	53	16.988831	129.370891	61.172054
...	...	...	...	...	...	...	...	...	...	...	...	...
2843	Zacatecas	2020	1	-168.749753	21.751855	168.507596	45	7	44	-168.749753	21.751855	168.507596
2844	Zacatecas	2020	2	19.600942	1.405455	4.525801	32	6	6	19.600942	1.405455	4.525801
2845	Zacatecas	2020	3	-182.763024	0.413678	25.387107	34	4	5	-182.763024	0.413678	25.387107
2846	Zacatecas	2020	4	-270.995383	-1.690795	-22.942449	38	5	8	-270.995383	-1.690795	-22.942449

Figura X. Formato original datos IED por trimestre, estado y tipo de inversión

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Eliminación de Entidades Federativas que hagan mención al total acumulado por Entidad. Esto se realiza dado que es redundante en los datos.
- Procesamiento de la columna *year* y *quarter_id* para generar correctamente la columna *quarter_id.*
- Eliminación de columnas que no serán utilizadas: *value_between_companies, value_new_investments* y *value_re_investments*.
- Transformación del *DataFrame* a formato tidy. Para ello, se crea una nueva columna denominada *investment_type*, la cual posee valores enteros para referirse a los tipos de inversiones (*between_companies*, *new_investments*, y *re_investments*) las cuales anteriormente estaban separadas en columnas individuales.
- Eliminación de datos por estado que posean una cantidad de registros confidenciales mayor a 10%.
- Eliminación de valores confidenciales. Dichos valores se almacenaban en la columna *value_c*.

En cuanto a la salida del proceso de ETL, se obtiene un *DataFrame* con 8.465 filas y 6 columnas. La estructura final de la tabla se presenta a continuación:

	ent_id	year	quarter_id	count	value_c	investment_type
0	1	1999	19991	58	20.645964	1
1	1	1999	19992	64	24.018523	1
2	1	1999	19993	67	16.640456	1
3	1	1999	19994	67	15.316704	1
4	1	2000	20001	70	16.988831	1
...	...	...	...	...	...	...
2843	32	2020	20201	44	168.507596	3
2844	32	2020	20202	6	4.525801	3
2845	32	2020	20203	5	25.387107	3
2846	32	2020	20204	8	-22.942449	3
2847	32	2021	20211	21	60.264757	3

Figura X. Formato final tabla de datos IED por trimestre, estado y tipo de inversión

7.3. Fdi_3_country_origin

Descripción General y Ejecución

El presente pipeline procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos relacionados a los orígenes de las inversiones. Los datos son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python fdi_3.py
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería bamboo-lib se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx; y su estructura inicial se compone de 17.873 filas y 6 columnas. A continuación, se visualiza la estructura inicial de los datos:

	Entidad federativa	Año	País	Monto	Recuento	Monto C
0	Aguascalientes	1999.0	Alemania	2.219975	9	2.219975
1	Aguascalientes	1999.0	Argentina	[illegible]	2	C
2	Aguascalientes	1999.0	Bélgica	[illegible]	2	C
3	Aguascalientes	1999.0	Brasil	[illegible]	2	C
4	Aguascalientes	1999.0	Canadá	0.000175	3	0.000175
...	...	...	...	...	...	...
17868	Zacatecas	2021.0	Japón	[illegible]	2	C
17869	Zacatecas	2021.0	Otros países	307.708870	5	307.70887
17870	Zacatecas	2021.0	Países Bajos	[illegible]	2	C
17871	Zacatecas	2021.0	Reino Unido de la Gran Bretaña e Irlanda del N...	[illegible]	1	C
17872	Total general	NaN	NaN	618651.568548	117518	618651.568548

Figura X. Formato original datos IED por país, año y estado

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Eliminación de Entidades Federativas que hagan mención al total acumulado por Entidad. Esto se realiza dado que es redundante en los datos.
- Reemplazo de Entidades Federativas por sus id.
- Renombre de columnas a: ent_id, year, country, value, count, y value_c.
- Eliminación de valores confidenciales. Dichos valores se almacenaban en la columna value_c.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, obtenemos un *DataFrame* con 10.058 filas y 5 columnas. La estructura final de la tabla se presenta a continuación:

ent_id	year	country	count	value_c
1	1999.0	deu	9	2.219975
1	1999.0	can	3	0.000175
1	1999.0	esp	11	12.304624
1	1999.0	usa	170	87.008375
1	1999.0	fra	12	0.463365
...	...	...	...	...
32	2020.0	che	3	-0.286696
32	2021.0	can	3	0.829693
32	2021.0	esp	4	6.086107
32	2021.0	usa	17	-110.344698
32	2021.0	xxa	5	307.708870

Figura X. Formato final tabla de datos IED por país, año y estado

7.4. fdi_4_investment_type

Descripción General y Ejecución

El pipeline *fdi_4.py* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos relacionados a los orígenes de las inversiones, considerando el tipo de inversión realizada de manera anual y según el estado receptor. Los datos son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python fdi_4.py
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería bamboo-lib se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx; y su estructura inicial se compone de 33.907 filas y 7 columnas. A continuación, se visualiza la estructura inicial de los datos:

Entidad federativa	Año	País	Inversión	Monto	Recuento	Monto C
Aguascalientes	1999.0	Alemania	Cuentas entre compañías	1.886538	5	1.886538
Aguascalientes	1999.0	Alemania	Nuevas inversiones	0.182343	3	0.182343
Aguascalientes	1999.0	Alemania	Reinversión de utilidades	[illegible]	2	C
Aguascalientes	1999.0	Argentina	Nuevas inversiones	[illegible]	2	C
Aguascalientes	1999.0	Bélgica	Cuentas entre compañías	[illegible]	1	C
...	...	...	...	...	...	...
Zacatecas	2021.0	Otros países	Reinversión de utilidades	46.401340	3	46.40134
Zacatecas	2021.0	Países Bajos	Cuentas entre compañías	[illegible]	2	C
Zacatecas	2021.0	Países Bajos	Reinversión de utilidades	[illegible]	1	C
Zacatecas	2021.0	Reino Unido de la Gran Bretaña e Irlanda del N...	Reinversión de utilidades	[illegible]	1	C
Total general	NaN	NaN	NaN	618651.568548	117518	618651.568548

Figura X. Formato original datos IED por año, país, tipo y entidad federativa

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar caracteres.
- Eliminación de Entidades Federativas que hagan mención al total acumulado por Entidad. Esto se realiza dado que es redundante en los datos.
- Reemplazo de Entidades Federativas por sus id, entre otros reemplazos.
- Renombre de columnas a: *ent_id, year, country, investment_type, value, count,* y *value_c.*
- Eliminación de valores confidenciales. Dichos valores se almacenaban en la columna *value_c*.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, obtenemos un *DataFrame* con 15.828 filas y 6 columnas. A continuación puede ser visualizada su estructura de salida:

ent_id	year	country	investment_type	count	value_c
1	1999.0	deu	1	5	1.886538
1	1999.0	deu	2	3	0.182343
1	1999.0	can	2	3	0.000175
1	1999.0	esp	2	10	12.303613
1	1999.0	usa	1	65	69.478034
...	...	...	...	...	...
32	2021.0	esp	3	4	6.086107
32	2021.0	usa	1	11	-157.643724
32	2021.0	usa	3	9	7.293840
32	2021.0	xxa	1	4	261.307530
32	2021.0	xxa	3	3	46.401340

Figura X. Formato original tabla de datos IED por año, país, tipo y entidad federativa

7.5. fdi_9_quarter/quarter_investment/_year/_year_investment

Descripción General y Ejecución

El pipeline *fdi_9.py* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos con relación a la inversión total por año, inversión total trimestral, inversión total por año y tipo de inversión e inversión total por trimestre y tipo de inversión, todas las anteriores a nivel nacional. Dado ello, son cuatro los cubos generados a partir de este pipeline. En cuanto a los datos, estos son almacenados en Google Storage.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python fdi_9.py
```

Descarga de datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y su estructura inicial varía de acuerdo al archivo procesado.

El archivo que contiene los datos de inversión total anual contaba con 24 filas y 4 columnas y su estructura inicial es como se muestra en la imagen siguiente:

Año	Monto	Recuento	Monto C
1999	13935.878705	8885	13935.878705
2000	18247.365344	8665	18247.365344
2001	30056.238976	8523	30056.238976
2002	24098.547790	9016	24098.547790
2003	18270.689675	9288	18270.689675

Figura X. Formato original datos IED anuales a nivel nacional

El archivo con datos de inversión total trimestral contenía 90 filas y 5 columnas y su estructura es la siguiente:

Año	Trimestre	Monto	Recuento	Monto C
1999	1.0	3571.645864	3889	3571.645864
1999	2.0	3395.880570	3952	3395.880570
1999	3.0	3028.447643	3336	3028.447643
1999	4.0	3939.904628	3567	3939.904628
2000	1.0	4599.300171	3631	4599.300171
...	...	...	...	...
2020	2.0	7329.820611	2403	7329.820611
2020	3.0	1330.397128	2694	1330.397128
2020	4.0	2374.609352	2821	2374.609352
2021	1.0	11864.030463	2764	11864.030463
Total general	NaN	618651.568548	117518	618651.568548

Figura X. Formato original datos IED trimestrales a nivel nacional

El archivo con datos de inversión total anual por tipo de inversión contenía 24 filas y 10 columnas y su estructura es la siguiente:

Año	Monto cuentas entre compañías	Monto nuevas inversiones	Monto reinversión de utilidades	Recuento cuentas entre compañías	Recuento nuevas inversiones	Recuento reinversión de utilidades	Monto C cuentas entre compañías	Monto C nuevas inversiones	Monto C reinversión de utilidades
1999	4986.214950	6606.924368	2342.739387	2548	6570	586	4986.214950	6606.924368	2342.739387
2000	5637.606776	8704.326873	3905.431695	2551	6332	533	5637.606776	8704.326873	3905.431695
2001	3046.299236	23110.766941	3899.172799	2400	6357	446	3046.299236	23110.766941	3899.172799
2002	5868.759710	15638.660235	2591.127845	2314	6869	313	5868.759710	15638.660235	2591.127845

Figura X. Formato original datos IED anuales por tipo de inversión a nivel nacional

Finalmente, el archivo con datos de inversión total trimestral por tipo de inversión contenía 90 filas y 11 columnas y su estructura es la siguiente:

Año	Trimestre	Monto cuentas entre compañías	Monto nuevas inversiones	Monto reinversión de utilidades	Recuento cuentas entre compañías	Recuento nuevas inversiones	Recuento reinversión de utilidades	Monto C cuentas entre compañías	Monto C nuevas inversiones	Monto C reinversión de utilidades
1999	1.0	1455.588937	1068.627613	1047.429314	1786	1968	380	1455.588937	1068.627613	1047.429314
1999	2.0	1228.553505	1549.699793	617.627272	1824	2074	270	1228.553505	1549.699793	617.627272
1999	3.0	527.945413	2244.442135	256.060095	1822	1467	221	527.945413	2244.442135	256.060095
1999	4.0	1774.127095	1744.154827	421.622706	1849	1714	216	1774.127095	1744.154827	421.622706

Figura X. Formato original datos IED trimestral por tipo de inversión a nivel nacional

Limpieza y Transformación

Como se ha mencionado anteriormente, dado que existen diferentes fuentes y salidas de datos, los procesos de ETL difieren entre sí. A continuación se presenta, para cada archivo, el proceso de ETL implementado que genera cada cubo de datos.

a) Inversión total anual

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas.
- Selección de filas que no corresponden a agregados totales.
- Transformación del *dtype* en las columnas *count* y *year* a enteros.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, obtenemos un *DataFrame* con 23 filas y 3 columnas. A continuación pueden ser visualizadas las primeras filas de su estructura de salida:

year	count	value_c
1999	8885	13935.878705
2000	8665	18247.365344
2001	8523	30056.238976
2002	9016	24098.547790
2003	9288	18270.689675
2004	10120	25032.076843

Figura X. Formato final tabla de datos IED anuales a nivel nacional

b) Inversión total trimestral

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas.
- Selección de filas que no corresponden a agregados totales.
- Creación de la columna *quarter_id* que posee los id temporales a nivel trimestral.
- Eliminación de columnas innecesarias.
- Transformación del *dtype* en las columnas *count* y *quarter_id* a enteros.

En cuanto a la salida del proceso de ETL, se obtiene un *DataFrame* con 89 filas y 3 columnas. A continuación puede ser visualizada su estructura de salida:

count	value_c	quarter_id
3889	3571.645864	19991
3952	3395.880570	19992
3336	3028.447643	19993
3567	3939.904628	19994
3631	4599.300171	20001
...	...	...
3252	16750.890078	20201
2403	7329.820611	20202
2694	1330.397128	20203
2821	2374.609352	20204
2764	11864.030463	20211

Figura X. Formato final tabla de datos IED trimestrales a nivel nacional

c) Inversión total por año y tipo de inversión

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas por medio de una lista.
- Selección de filas que no corresponden a agregados totales.
- Eliminación de columnas que no serán utilizadas: *value_between_companies, value_new_investments,* y *value_re_investments*.
- Transformación del *DataFrame* a formato *tidy*. Para ello, se crea una nueva columna denominada *investment_type*, la cual posee valores enteros para referirse a inversiones: *between_companies, new_investments*, y *re_investments*; las cuales anteriormente estaban separadas en columnas individuales.
- Reemplazo en columna *investment_type* por sus id's.

En cuanto a la salida del proceso de ETL, se obtiene un *DataFrame* con 69 filas y 4 columnas. A continuación puede ser visualizada su estructura de salida:

year	count	value_c	investment_type
1999.0	2548.0	4986.214950	1
2000.0	2551.0	5637.606776	1
2001.0	2400.0	3046.299236	1
2002.0	2314.0	5868.759710	1
2003.0	2285.0	7095.183763	1
...	...	...	...
2017.0	696.0	11951.846494	3
2018.0	820.0	13245.576041	3
2019.0	1007.0	17998.993194	3
2020.0	786.0	16064.865878	3
2021.0	423.0	7025.211257	3

Figura X. Formato final tabla de datos IED anuales por tipo de inversión a nivel nacional

d) Inversión total por trimestre y tipo de inversión

- Normalización de nombres de columnas. Se utiliza el módulo de Pandas *norm* para transformar cada *caracter* en minúscula, y también, el módulo *replace* para reemplazar *caracteres*.
- Renombre de columnas por medio de una lista.
- Selección de filas que no corresponden a agregados totales.
- Creación de la columna *quarter_id* que posee los id temporales a nivel trimestral.
- Eliminación de columnas que no serán utilizadas: *year, quarter, value_between_companies, value_new_investments* y *value_re_investments*.
- Transformación del *DataFrame* a formato *tidy*. Para ello, se crea una nueva columna denominada *investment_type*, la cual posee valores enteros para referirse a los tipos de inversiones (*between_companies, new_investments* y *re_investments*) las cuales anteriormente estaban separadas en columnas individuales.
- Reemplazo en columna *investment_type* por sus id's.

En cuanto a la salida del proceso de ETL, se obtiene un *DataFrame* con 267 filas y 4 columnas. La estructura final de la tabla se presenta a continuación:

quarter_id	count	value_c	investment_type
19991.0	1786.0	1455.588937	1
19992.0	1824.0	1228.553505	1
19993.0	1822.0	527.945413	1
19994.0	1849.0	1774.127095	1
20001.0	1873.0	980.191516	1
...	...	...	...
20201.0	645.0	14336.759885	3
20202.0	139.0	815.770815	3
20203.0	126.0	801.913478	3
20204.0	99.0	110.421700	3
20211.0	423.0	7025.211257	3

Figura X. Formato datos IED trimestral por tipo de inversión a nivel nacional

7.6. fdi_quarter_* y fdi_year_*

Descripción General y Ejecución

El pipeline *fdi_additional_tables.py* procesa los datos facilitados por la Secretaría de Economía del Gobierno de México que toman relación con Inversión Extranjera Directa (IED). En específico, procesa datos relacionados con la inversión total trimestral y anual para cada sector, subsector e industria económica y tipo de inversión, además de inversión total anual por industria y país de origen y la inversión total anual por entidad federativa. De acuerdo a lo anterior, se generan 6 cubos a partir del pipeline. Los problemas de anonimización impiden juntar todos los datos en un único cubo, dado que a medida que se agregan los niveles inferiores se pierden valores en los niveles superiores. Por ellos es necesario procesar los datos por separado y generar 6 cubos diferentes:

1) *fdi_quarter_industry_investment*: IED trimestral por sector/subsector/industria y tipo de inversión.
2) *fdi_year_industry_country*: IED anual por sector/subsector/industria y países.
3) *fdi_year_state_industry*: IED anual por sector/subsector/industria y entidad federativa.
4) *fdi_year_industry*: IED anual por sector/subsector/industria.
5) *fdi_quarter_industry*: IED trimestral por sector/subsector/industria.
6) *fdi_year_investment_industry*: IED anual por sector/subsector/industria y tipo de inversión.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Además, dado que existen diferentes fuentes y combinaciones de parámetros involucrados, se creó el script *run_fdi_additional_tables.py* que almacena los parámetros utilizados para el procesamiento de los datos que generan los seis cubos mencionados.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/foreign_direct_investment/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/foreign_direct_investment/$ python run_fdi_additional_tables.py
```

Descarga de datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx, cada archivo contiene diferentes hojas de datos y su estructura inicial varía de acuerdo al archivo procesado.

Al momento de escribir esta documentación, el archivo con datos de inversión total trimestral por sector industrial y tipo de inversión contaba con de 6.141 filas y 7 columnas. Cabe destacar que un archivo de las mismas características existe para el subsector industrial y rama industrial.

Sector	Año	Trimestre	Inversión	Monto	Recuento	Monto C
11 Agricultura, cría y explotación de animales...	1999	1	Cuentas entre compañías	0.435078	4	0.435078
21 Minería	1999	1	Cuentas entre compañías	6.720601	26	6.720601
22 Generación, transmisión y distribución de e...	1999	1	Cuentas entre compañías	6.893001	7	6.893001
23 Construcción	1999	1	Cuentas entre compañías	29.682373	18	29.682373
31 Industrias manufactureras	1999	1	Cuentas entre compañías	590.494225	270	590.494225
...	...	...	...	...	...	...

Figura X. Formato original datos IED por sector, trimestre y tipo de inversión

El archivo con datos de inversión total anual por industria y país de origen se estructuraba en 11.566 filas y 6 columnas.

Año de materialización	País de Origen_otros	Sector	Suma de Monto en millones	Recuento distinto de Expediente	Monto C
1999	Alemania	11 Agricultura, cría y explotación de animales...	[illegible]	2	C
1999	Alemania	23 Construcción	61.988283	7	61.988283
1999	Alemania	31 Industrias manufactureras	2.723321	11	2.723321
1999	Alemania	32 Industrias manufactureras	199.682383	33	199.682383
1999	Alemania	33 Industrias manufactureras	180.533753	91	180.533753
...	...	...	...	...	...

Figura X. Formato original datos IED por sector, año y país

El archivo con datos de inversión total anual por estados e industrias contaba con 12.908 filas y 6 columnas.

Año de materialización	Entidad federativa	Sector	Suma de Monto en millones	Recuento distinto de Expediente	Monto C
1999	Aguascalientes	11 Agricultura, cría y explotación de animales...	[illegible]	1	C
1999	Aguascalientes	22 Generación, transmisión y distribución de e...	0.385442	4	0.385442
1999	Aguascalientes	23 Construcción	0.002578	3	0.002578
1999	Aguascalientes	31 Industrias manufactureras	37.647984	43	37.647984
1999	Aguascalientes	32 Industrias manufactureras	21.530571	24	21.530571
...	...	...	...	...	...

Figura X. Formato original datos IED por sector, año y estado

El archivo con datos de inversión total por año e industria se estructura en 529 filas y 5 columnas.

Año	Sector	Monto	Recuento	Monto C
1999	11 Agricultura, cría y explotación de animales...	87.791614	63	87.791614
1999	21 Minería	229.078375	104	229.078375
1999	22 Generación, transmisión y distribución de e...	345.558844	39	345.558844
1999	23 Construcción	206.980556	158	206.980556
1999	31 Industrias manufactureras	1642.160354	724	1642.160354
...	...	...	...	...

Figura X. Formato original datos IED por año e industria

El archivo con datos de inversión total trimestral por industrias contaba de 2.047 filas y 6 columnas.

Sector	Año	Trimestre	Monto	Recuento	Monto C
11 Agricultura, cría y explotación de animales...	1999	1	28.431735	19	28.431735
21 Minería	1999	1	120.914406	51	120.914406
22 Generación, transmisión y distribución de e...	1999	1	12.942499	16	12.942499
23 Construcción	1999	1	80.416861	61	80.416861
31 Industrias manufactureras	1999	1	791.493028	399	791.493028
...	...	...	...	...	...

Figura X. Formato original datos IED por industria y trimestre

Finalmente, el archivo con datos de inversión total anual por tipo de inversión e industrias se estructuraba en 1.587 filas y 6 columnas.

Sector	Inversión	Año	Monto	Recuento	Monto C
11 Agricultura, cría y explotación de animales...	Cuentas entre compañías	1999	2.301394	8	2.301394
21 Minería	Cuentas entre compañías	1999	19.418630	33	19.41863
22 Generación, transmisión y distribución de e...	Cuentas entre compañías	1999	32.162537	11	32.162537
23 Construcción	Cuentas entre compañías	1999	67.081704	29	67.081704
31 Industrias manufactureras	Cuentas entre compañías	1999	794.708157	400	794.708157
...	...	...	...	...	...

Figura X. Formato original datos IED por industria, año y tipo de inversión

Limpieza y Transformación

Como se ha mencionado anteriormente, dado que existen diferentes fuentes y salidas de datos, los procesos de ETL difieren entre sí. Por ello, a continuación se presenta, para cada cubo generado, el proceso de ETL implementado.

a) Inversión total trimestral por sector/subsector/rama industrial y tipo de inversión

- Renombre de columnas.
- Eliminación de filas que contienen totales. Esto dado que es información redundante.
- Reformulación de variable *quarter_id*, donde su id corresponde a una combinación de la variable *year* y *quarter_id*.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Eliminación de datos por tipo de inversión que posean un número de valores confidenciales mayor a 10%.
- Reemplazo de tipos de inversión por su id.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 4.049 filas y 8 columnas. A continuación puede ser visualizada su estructura de salida:

sector_id	quarter_id	investment_type	value	count	value_c	subsector_id	industry_group_id
11	19991.0	1.0	0.435078	4.0	0.43507799999999996	0.0	0.0
21	19991.0	1.0	6.720601	26.0	6.7206009999999985	0.0	0.0
22	19991.0	1.0	6.893001	7.0	6.893001000000003	0.0	0.0
23	19991.0	1.0	29.682373	18.0	29.682372999999995	0.0	0.0
31-33	19991.0	1.0	590.494225	270.0	590.4942249999998	0.0	0.0
...	...	...	...	...	...	...	...
31-33	20211.0	3.0	808.532526	66.0	808.5325263500009	0.0	0.0
31-33	20211.0	3.0	1685.157338	157.0	1685.1573375899982	0.0	0.0
43	20211.0	3.0	703.078245	54.0	703.0782450699996	0.0	0.0
55	20211.0	3.0	0.000000	0.0	0	0.0	0.0
93	20211.0	3.0	0.000000	0.0	0	0.0	0.0

Figura X. Formato final tabla de datos IED por sector, trimestre y tipo de inversión

b) Inversión total anual por industria y país de origen

- Renombrar columnas
- Selección de claves primarias. Dichas claves serán guardadas en una lista llamada *pk_id*.
- Eliminación de filas cuya clave primaria sea nula.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Reemplazo de países por sus id.
- Selección de filas con valores no confidenciales.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL, y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 6.040 filas y 8 columnas. A continuación puede ser visualizada su estructura de salida:

year	country_id	sector_id	value	count	value_c	subsector_id	industry_group_id
1999.0	deu	23	61.988283	7.0	61.988283	0.0	0.0
1999.0	deu	31-33	2.723321	11.0	2.7233209999999994	0.0	0.0
1999.0	deu	31-33	199.682383	33.0	199.68238300000002	0.0	0.0
1999.0	deu	31-33	180.533753	91.0	180.53375299999993	0.0	0.0
1999.0	deu	43	188.638393	57.0	188.6383930000003	0.0	0.0
...	...	...	...	...	...	...	...
2021.0	che	31-33	-0.297268	4.0	-0.29726779000000025	0.0	0.0
2021.0	che	43	-0.297364	4.0	-0.29736410999999885	0.0	0.0
2021.0	che	48-49	16.277043	5.0	16.2770426	0.0	0.0
2021.0	twn	31-33	5.010208	5.0	5.01020845	0.0	0.0
2021.0	ven	72	1.146261	17.0	1.14626135	0.0	0.0

Figura X. Formato final tabla de datos IED por sector, año y país

c) Inversión total anual por estado e industria

- Renombrar columnas.
- Selección de claves primarias. Dichas claves serán guardadas en una lista llamada *pk_id*.
- Eliminación de filas cuya clave primaria sea nula.
- Reemplazo de entidades federativas por sus ids.
- Selección de filas con valores no confidenciales.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Reemplazo de países por sus id.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL, y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 10.682 filas y 8 columnas. A continuación puede ser visualizada su estructura de salida:

year	ent_id	sector_id	value	count	value_c	subsector_id	industry_group_id
1999.0	1	22	0.385442	4.0	0.385442	0.0	0.0
1999.0	1	23	0.002578	3.0	0.002578	0.0	0.0
1999.0	1	31-33	37.647984	43.0	37.647984	0.0	0.0
1999.0	1	31-33	21.530571	24.0	21.530571	0.0	0.0
1999.0	1	31-33	737.684528	59.0	737.684528	0.0	0.0
...	...	...	...	...	...	...	...
2021.0	32	31-33	-0.063735	5.0	-0.063735	0.0	0.0
2021.0	32	31-33	-4.742262	4.0	-4.742262	0.0	0.0
2021.0	32	46	4.597508	3.0	4.597508	0.0	0.0
2021.0	32	52	6.857873	7.0	6.857873	0.0	0.0
2021.0	32	54	0.234930	3.0	0.234930	0.0	0.0

Figura X. Formato final tabla de datos IED por sector, año y estado

d) Inversión total por año y sector industrial

- Renombrar columnas.
- Selección de claves primarias. Dichas claves serán guardadas en una lista llamada *pk_id*.
- Eliminación de filas cuya clave primaria sea nula.
- Eliminación de datos que agrupados por clave primaria posean un número de valores confidenciales mayor a 10%.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Selección de valores en columna *value_c* que no posean valores confidenciales o igual a *false*.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL, y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 528 filas y 7 columnas. A continuación puede ser visualizada su estructura de salida:

year	sector_id	value	count	value_c	subsector_id	industry_group_id
1999.0	11	87.791614	63.0	87.791614	0.0	0.0
1999.0	21	229.078375	104.0	229.078375	0.0	0.0
1999.0	22	345.558844	39.0	345.558844	0.0	0.0
1999.0	23	206.980556	158.0	206.980556	0.0	0.0
1999.0	31-33	1642.160354	724.0	1642.160354	0.0	0.0
...	...	...	...	...	...	...
2021.0	62	4.110380	5.0	4.110380	0.0	0.0
2021.0	71	42.076211	11.0	42.076211	0.0	0.0
2021.0	72	375.273216	906.0	375.273216	0.0	0.0
2021.0	81	16.236747	6.0	16.236747	0.0	0.0
2021.0	93	0.000000	0.0	0.000000	0.0	0.0

Figura X. Formato final tabla de datos IED por año e industria

e) Inversión total trimestral por sector industrial

- Renombrar columnas.
- Selección de claves primarias. Dichas claves serán guardadas en una lista llamada *pk_id*.
- Eliminación de filas cuya clave primaria sea igual a *Total general*.
- Reformulación de columna *quarter_id*. Se agrega el año previo a la referencia trimestral.
- Eliminación de datos que agrupados por clave primaria posean un número de valores confidenciales mayor a 10%.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Selección de valores en columna *value_c* que no posean valores confidenciales o igual a *false*.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL, y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 1.952 filas y 7 columnas. A continuación puede ser visualizada su estructura de salida:

sector_id	quarter_id	value	count	value_c	subsector_id	industry_group_id
11	19991.0	28.431735	19.0	28.431735	0.0	0.0
21	19991.0	120.914406	51.0	120.914406	0.0	0.0
22	19991.0	12.942499	16.0	12.942499	0.0	0.0
23	19991.0	80.416861	61.0	80.416861	0.0	0.0
31-33	19991.0	791.493028	399.0	791.493028	0.0	0.0
...	...	...	...	...	...	...

Figura X. Formato final tabla de datos IED por industria y trimestre

f) Inversión total anual por tipo de inversión y sector industrial

- Renombrar columnas.
- Selección de claves primarias. Dichas claves serán guardadas en una lista llamada *pk_id*.
- Eliminación de filas cuya clave primaria sea igual a *Total general*.
- Eliminación de datos que agrupados por tipo de inversión posean un número de valores confidenciales mayor a 10%.
- Creación de columnas faltantes: *sector_id, subsector_id* e *industry_group_id*. Dado que existe un archivo para cada nivel industrial, en cada archivo se agregan columnas con ceros para los niveles faltantes.
- Reemplazo del nombre de sectores industriales por sus id.
- Selección de valores no confidenciales.
- Reemplazo de tipos de inversión por sus ids.
- Transformación de columnas de datos a tipo *float*.

En cuanto a la salida del proceso de ETL, y considerando el archivo con valores para sectores industriales, se obtiene un *DataFrame* con 1352 filas y 7 columnas. La estructura final de la tabla se presenta a continuación:

sector_id	investment_type	year	value	count	subsector_id	industry_group_id
11	1.0	1999.0	2.301394	8.0	0.0	0.0
21	1.0	1999.0	19.418630	33.0	0.0	0.0
22	1.0	1999.0	32.162537	11.0	0.0	0.0
23	1.0	1999.0	67.081704	29.0	0.0	0.0
31-33	1.0	1999.0	794.708157	400.0	0.0	0.0
...	...	...	...	...	...	...

Figura X. Formato final tabla de datos IED por industria, año y tipo de inversión

8. Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES)

8.1. Anuies_enrollment

Descripción General y Ejecución

El pipeline *anuies_enrollment.py* procesa los datos facilitados por la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES) que toman relación con los procesos de inscripción de estudiantes. Los datos son descargados desde la plataforma de la Asociación Nacional de Universidades e Instituciones de Educación Superior, y son almacenados en Google Storage para su posterior procesamiento. Cabe destacar que los datos vienen desagregados a nivel municipal.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de Clickhouse, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/anuies/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli) `~/data_etl/etl/anuies/$ bamboo-cli --folder . --entry enrollment_pipeline`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y se identifican 66.593 filas y 76 columnas. A continuación, se visualiza la estructura inicial de los datos:

	ENTIDAD	MUNICIPIO	CVE CAMPO UNITARIO	NIVEL	CICLO	CLAVE INSTITUCIÓN	CLAVE CENTRO DE TRABAJO	NOMBRE CARRERA SEP	NOMBRE INSTITUCIÓN ANUIES	NOMBRE ESCUELA/CAMPUS ANUIES	...	Suma de PNI-TLA	Suma de PNI-VER
	AGUASCALIENTES	001	11000	MAESTRÍA	2016-2017	01MSU0009F	01PSU0007E	MAESTRÍA EN CIENCIAS DE LA EDUCACIÓN	UNIVERSIDAD DEL VALLE DE MÉXICO	UNIVERSIDAD DEL VALLE DE MÉXICO - CAMPUS AGUAS...	...	0	0
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total UNIVERSIDAD DEL VALLE DE MÉXICO - CAMPUS...	...	0	0
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total UNIVERSIDAD DEL VALLE DE MÉXICO	NaN	...	0	0
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total MAESTRÍA EN CIENCIAS DE LA EDUCACIÓN	NaN	NaN	...	0	0
	NaN	NaN	NaN	NaN	NaN	NaN	Total 01PSU0007E	NaN	NaN	NaN	...	0	0

Figura X. Formato original datos ANUIES - matriculados

Extracción y Transformación

Este paso del pipeline sigue la secuencia mencionada a continuación:

- Renombre de columnas
- Se completan celdas *NaN* con los valores correspondientes haciendo uso de la función *ffill()*.
- Reemplazo de las entidades federativas (*ent_id*) por sus ids.
- Limpieza de celdas que incluyen el *string* *Total* acompañando a valores numéricos. Dicho *string* es eliminado para facilitar agregaciones y futuros análisis. Así también, se eliminan puntos (.) y dos puntos (:) para mejorar el formato del set de datos.
- Reemplazo de ids para los municipios. Para ello, se hace una *query* a *Clickhouse* con motivo de traer la dimensión *dim_shared_geography_mun* que contiene dichos ids.

- Uso de la función *melt* de pandas para llevar el set de datos a un formato *tidy*.
- Reemplazo de los valores en las columnas *sex*, *type*, y *age* por medio de diccionarios definidos en el script *static.py*.
- Se añaden los ids para las columnas *program* y *campus*. Esto se realiza posterior a la lectura de archivos almacenados en Google Cloud Storage y que contienen dicha información.
- Selección de variables a utilizar: *mun_id, campus_id, program, type, sex, value* y *age*.
- Procesamiento del nombre de instituciones, campus, carreras y programas.
- Creación de la columna *year* la cual contiene información con respecto al periodo de tiempo en estudio.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 87.580 filas y 8 columnas. La estructura final de la tabla se presenta a continuación:

	mun_id	campus_id	program	type	sex	value	age	year
0	1001.0	1600001.0	11500021.0	13.0	1.0	2.0	5.0	2017
1	1001.0	1100001.0	31100173.0	13.0	1.0	2.0	5.0	2017
2	1001.0	1600001.0	42100257.0	13.0	1.0	1.0	5.0	2017
3	1001.0	1600001.0	41200055.0	13.0	1.0	1.0	5.0	2017
4	1001.0	1100004.0	41400021.0	13.0	1.0	1.0	5.0	2017

Figura X. Formato final tabla de datos ANUIES - matriculados

8.2. Anuies_origin

Descripción General y Ejecución

El pipeline *origin_pipeline.py* procesa los datos facilitados por la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES) que toman relación con la entidad federativa de origen de los estudiantes. Los datos son descargados desde la plataforma de la Asociación Nacional de Universidades e Instituciones de Educación Superior, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/anuies/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli) `~/data_etl/etl/anuies/$ bamboo-cli --folder . --entry origin_pipeline`

Descarga de datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y se identifican 66.593 filas y 76 columnas. A continuación, se visualiza la estructura inicial de los datos:

ENTIDAD	MUNICIPIO	CVE CAMPO UNITARIO	NIVEL	CICLO	CLAVE INSTITUCIÓN	CLAVE CENTRO DE TRABAJO	NOMBRE CARRERA SEP	NOMBRE INSTITUCIÓN ANUIES	NOMBRE ESCUELA/CAMPUS ANUIES	...	Suma de PNI-TLA	Suma de PNI-VER
AGUASCALIENTES	001	11000	MAESTRÍA	2016-2017	01MSU0009F	01PSU0007E	MAESTRÍA EN CIENCIAS DE LA EDUCACIÓN	UNIVERSIDAD DEL VALLE DE MÉXICO	UNIVERSIDAD DEL VALLE DE MÉXICO - CAMPUS AGUAS...	...	0	0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total UNIVERSIDAD DEL VALLE DE MÉXICO - CAMPUS...	...	0	0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total UNIVERSIDAD DEL VALLE DE MÉXICO	NaN	...	0	0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Total MAESTRÍA EN CIENCIAS DE LA EDUCACIÓN	NaN	NaN	...	0	0
NaN	NaN	NaN	NaN	NaN	NaN	Total 01PSU0007E	NaN	NaN	NaN	...	0	0

Figura X. Formato original datos ANUIES - origen de estudiantes

Extracción y Transformación

Este paso del pipeline sigue la secuencia mencionada a continuación:

- Renombre de columnas
- Se completan celdas *NaN* con los valores correspondientes haciendo uso de la función *ffill()*.
- Reemplazo de las entidades federativas (*ent_id*) por sus id.
- Limpieza de celdas que incluyen el *string* *Total* acompañando a valores numéricos. Dicho *string* es eliminado para facilitar agregaciones y futuros análisis. Así también, se eliminan los puntos (.) y dos puntos (:) para mejorar el formato del set de datos.
- Reemplazo de ids para los municipios. Para ello, se hace una *query* a Clickhouse con motivo de traer la dimensión *dim_shared_geography_mun* que contiene dichos ids.
- Uso de la función *melt* de pandas para llevar el set de datos a un formato *tidy*.
- Reemplazo de los valores en las columnas *type*, y *origin* por medio de diccionarios definidos en el script *static.py*.
- Procesamiento del nombre de instituciones, campus, carreras y programas. Se añaden los ids para las columnas *program*, *campus*. Esto se realiza posterior a

la lectura de archivos almacenados en Google Cloud Storage y que contienen dicha información.

- Selección de variables a utilizar: *mun_id, campus_id, program, type, origin* y *value*.
- Creación de la columna *year* la cual contiene información con respecto al periodo de tiempo en estudio.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 18.212 filas y 7 columnas. La estructura final de la tabla se presenta a continuación:

	mun_id	campus_id	program	type	origin	value	year
0	1001.0	1200001.0	11500028.0	13.0	1.0	11.0	2017
1	1001.0	1600001.0	11500021.0	13.0	1.0	43.0	2017
2	1001.0	1000002.0	11100061.0	13.0	1.0	23.0	2017
3	1001.0	1000002.0	11500066.0	14.0	1.0	16.0	2017
4	1001.0	2100001.0	12604025.0	13.0	1.0	2.0	2017

Figura X. Formato final tabla de datos ANUIES - origen de estudiantes

8.3. Anuies_status

Descripción General y Ejecución

El pipeline *status_pipeline.py* procesa los datos facilitados por la Asociación Nacional de Universidades e Instituciones de Educación Superior (ANUIES) que toman relación con el grado académico de los estudiantes (egresados, graduados y titulados). Los datos son descargados desde la plataforma de la Asociación Nacional de Universidades e Instituciones de Educación Superior, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/anuies/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(bamboo-cli) `~/data_etl/etl/anuies/$ bamboo-cli --folder . --entry status_pipeline`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y se identifican 40.549 filas y 49 columnas. A continuación, se visualiza la estructura inicial de los datos:

ENTIDAD FEDERATIVA	MUNICIPIO	CLAVE CAMPO UNITARIO	NIVEL	NOMBRE INSTITUCIÓN	NOMBRE PROGRAMA EDUCATIVO	NOMBRE ESCUELA/CAMPUS/PLANTEL	SOSTENIMIENTO	CAMP UNITARI D FORMACIÓ
AGUASCALIENTES	AGUASCALIENTES	11000.0	Maestría	CENTRO REGIONAL DE EDUCACIÓN NORMAL DE AGUASCA...	MAESTRÍA EN INNOVACIÓN DIDÁCTICA	CENTRO REGIONAL DE EDUCACIÓN NORMAL DE AGUASCA...	Público	No aplic Camp Unitar
NaN	NaN	11100.0	Licenciatura Universitaria y Tecnológica	CENTRO UNIVERSITARIO BRITÁNICO DE MÉXICO	LICENCIATURA EN PEDAGOGÍA	CENTRO UNIVERSITARIO BRITÁNICO DE MÉXICO	Particular	No aplic Camp Unitar
NaN	NaN	NaN	NaN	INSTITUTO UNIVERSITARIO DEL CENTRO DE MÉXICO	LICENCIATURA EN PEDAGOGÍA	INSTITUTO UNIVERSITARIO DEL CENTRO DE MÉXICO -...	Particular	No aplic Camp Unitar
NaN	NaN	NaN	NaN	TECNOLÓGICO UNIVERSITARIO AGUASCALIENTES	LICENCIATURA EN PEDAGOGÍA	TECNOLÓGICO UNIVERSITARIO AGUASCALIENTES	Particular	No aplic Camp Unitar
NaN	NaN	NaN	NaN	UNIDAD U.P.N. 011 AGUASCALIENTES	LICENCIATURA EN EDUCACIÓN E INNOVACIÓN	UNIDAD U.P.N. 011 AGUASCALIENTES	Público	No aplic Camp Unitar

Figura X. Formato original datos ANUIES - grados académicos

Extracción y Transformación

Este paso del pipeline sigue la secuencia mencionada a continuación:

- Renombre de columnas
- Se completan celdas *NaN* con los valores correspondientes haciendo uso de la función *ffill()*.
- Reemplazo de las entidades federativas (*ent_id*) por sus ids.
- Limpieza de celdas que incluyen el *string* *Total* acompañando a valores numéricos. Dicho *string* es eliminado para facilitar agregaciones y futuros análisis. Así también, se eliminan los puntos (.) y dos puntos (:) para mejorar el formato del set de datos.
- Reemplazo de ids para los municipios. Para ello, se hace una *query* a Clickhouse con motivo de traer la dimensión *dim_shared_geography_mun* que contiene dichos ids.
- Cambios menores en el nombre de columnas para posteriormente usar la función *melt* de pandas para llevar el set de datos a un formato *tidy*.
- Reemplazo de los valores en las columnas *stat, sex* y *type* por medio de diccionarios definidos en el *script static.py*.
- Procesamiento del nombre de instituciones, campus, carreras y programas. Se añaden los ids para las columnas *program*, *campus*. Esto se realiza posterior a

la lectura de archivos almacenados en Google Cloud Storage y que contienen dicha información.

- Selección de variables a utilizar: *mun_id, campus_id, program, type, sex, value* y *stat*.
- Creación de la columna *year* la cual contiene información con respecto al periodo de tiempo en estudio.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 26.109 filas y 8 columnas. A continuación puede ser visualizada su estructura de salida:

	mun_id	campus_id	program	type	sex	value	stat	year
0	1001.0	1000001.0	11100061.0	13.0	1.0	1.0	1.0	2019
1	1001.0	1100001.0	11200021.0	13.0	1.0	9.0	1.0	2019
2	1001.0	1100001.0	11200142.0	14.0	1.0	4.0	1.0	2019
3	1001.0	900002.0	11500004.0	13.0	1.0	2.0	1.0	2019
4	1001.0	300001.0	11500021.0	13.0	1.0	3.0	1.0	2019

Figura X. Formato final tabla de datos ANUIES - grados académicos

9. Población y Vivienda

9.1. Conapo_metro_area_population

Descripción General y Ejecución

El pipeline *metro_areas_population.py* procesa los datos facilitados por el Consejo Nacional de Población (CONAPO) que toman relación con la población total a nivel de Zona Metropolitana. Los datos son descargados desde la plataforma del Consejo Nacional de Población, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/metro_areas_population/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/metro_areas_population/$ bamboo-cli --folder . --entry metro_areas_population`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y, con respecto a su estructura, se identifican 415 filas y 13 columnas. A continuación, se visualiza la estructura inicial de los datos:

	zona_metropolitana	mun_id	1900	...	DMU2 (hab/ha)	mun	zona_metropolitana_id
0	Aguascalientes	1001	506274	...	108.2	Aguascalientes	99101
1	Aguascalientes	1005	41092	...	75.0	Jesús María	99101
2	Aguascalientes	1011	n.a.	...	83.3	San Francisco de los Romo	99101
3	Ensenada	2001	259979	...	54.3	Ensenada	99201
4	Mexicali	2002	601938	...	59.3	Mexicali	99202

Figura X. Formato original datos población CONAPO

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de datos.
- Reemplazo de valores *n.a* por el valor 0.
- Transformación de las columnas a un formato *tidy*. Se crean dos nuevas columnas: la primera será el año en estudio, y la segunda, la población.
- Se renombran las columnas para tener mejor definición de sus atributos.
- Se elimina la variable *zm_id*. Esto ya que el municipio puede ser conocido a partir de la variable *zm_name*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 1.245 filas y 5 columnas . A continuación puede ser visualizada su estructura de salida:

	zm_name	mun	mun_id	year	population
0	Aguascalientes	Aguascalientes	1001	2000	643419
1	Aguascalientes	Jesús María	1005	2000	64097
2	Aguascalientes	San Francisco de los Romo	1011	2000	20066
3	Ensenada	Ensenada	2001	2000	370730
4	Mexicali	Mexicali	2002	2000	764602

Figura X. Formato final tabla de datos población CONAPO

9.2. Inegi_population_total

Descripción General y Ejecución

El pipeline *population_total_pipeline.py* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía de México (INEGI) que toman relación con la Población Total extraída del Censo de Población y Vivienda de los años 2010 y 2020. Los datos, una vez descargados desde su sitio, son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/inegi_census/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/inegi_census/$ bash population_total_ingest.sh
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y separada en archivos por entidades federativas para el 2010 y 2020. A modo de ejemplo, en el caso de la entidad federativa de Aguascalientes en 2020, la estructura inicial de archivo se compone de 2.058 filas y 232 columnas. A continuación, se visualiza la estructura inicial de estos datos:

ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	LONGITUD	LATITUD	ALTITUD	POBTOT	...	VPH_CEL	VPH_INTER
1	Aguascalientes	0	Total de la entidad Aguascalientes	0	Total de la Entidad	NaN	NaN	NaN	1425607	...	359895	236003
1	Aguascalientes	0	Total de la entidad Aguascalientes	9998	Localidades de una vivienda	NaN	NaN	NaN	3697	...	732	205
1	Aguascalientes	0	Total de la entidad Aguascalientes	9999	Localidades de dos viviendas	NaN	NaN	NaN	3021	...	470	146
1	Aguascalientes	1	Aguascalientes	0	Total del Municipio	NaN	NaN	NaN	948990	...	251719	178619
1	Aguascalientes	1	Aguascalientes	1	Aguascalientes	102°17'45.768" W	21°52'47.362" N	1878.0	863893	...	232793	169675
...	...	...	...	...	...	...	...	...	...	...	...	...

Figura X. Formato original datos del Censo Poblacional 2020 - Aguascalientes

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Selección de columnas a utilizar, las cuales son: *entidad, mun, loc, nom_mun, nom_loc, pobfem, y pobmas*.
- Eliminación de filas que, en el valor de localidad, posean el *string* *Total del Municipio*. Esto debido a que las filas con valores totales son redundantes.
- Reformulación de la columna *mun_id*. Se agrega el id de la entidad previo al id del municipio.
- Renombre de columnas.
- Creación de columna *year*, la cual define en año de realización del Censo.
- Transformación del *DataFrame* hacia un formato *tidy*. Para ello, se utiliza el módulo *melt* de Pandas. En ello, se crea la variable *sex* que define el sexo, y la variable *population* que define el total de la población.
- Reemplazo de la variable *sex* por sus ids.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, se obtiene un *DataFrame* con 22 filas y 4 columnas. El proceso es iterativo para cada entidad federativa y finalmente se unen todas las tablas en una única tabla que contendrá los valores para las 32 entidades federativas. A continuación puede ser visualizado un extracto de la estructura de salida de la tabla para Aguascalientes:

mun_id	year	sex	population
1001	2020	2	486917
1002	2020	2	26275
1003	2020	2	29687
1009	2020	1	11114
1010	2020	1	10446
1011	2020	1	30705

Figura X. Formato final tabla de datos del Censo de Población - Aguascalientes

9.3. Inegi_population

Descripción General y Ejecución

El pipeline *population_pipeline.py* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía de México que toman relación con datos a nivel de Personas desde la Encuesta Intercensal 2015 y el cuestionario ampliado del Censo de Población y Vivienda de 2020. Los datos, una vez descargados desde su sitio, son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/inegi_intercensal_survey/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/inegi_intercensal_survey/$ bash population_ingest.sh
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería bamboo-lib se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y organizada por entidades federativas. A modo de ejemplo, el archivo con datos para la entidad federativa de Aguascalientes en 2020 presenta una estructura de 95.983 filas y 92 columnas. A continuación, se visualiza la estructura inicial de los datos:

ENT	MUN	LOC50K	ID_VIV	ID_PERSONA	COBERTURA	ESTRATO	UPM	FACTOR	CLAVIVP	...	HIJOS_SOBREVIV	FECHA_NAC_M
1	1	1	10010000001	1001000000100001	2	01-001-0001-00	1	59	1	...	2.0	9.0
1	1	1	10010000001	1001000000100002	2	01-001-0001-00	1	59	1	...	NaN	NaN
1	1	1	10010000001	1001000000100003	2	01-001-0001-00	1	59	1	...	NaN	NaN
1	1	1	10010000002	1001000000200001	2	01-001-0001-00	1	59	1	...	NaN	NaN
1	1	1	10010000002	1001000000200002	2	01-001-0001-00	1	59	1	...	2.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...

Figura X. Formato original datos cuestionario ampliado Censo Poblacional - Aguascalientes

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Creación de variables definidas en la lista *extra_columns_low* con el valor *numpy.nan*. En dicha lista existen variables que no existen en la Encuesta Intercensal 2015, o el Censo de Población y Vivienda de 2020, o en ambos. Esto se realiza con motivo de integrar los datos en un mismo *DataFrame*.
- Reformulación de la variable *mun_id*. Se define mediante la agregación de: entidad, y municipio. En ese orden.
- Creación de la columna mun_id_trab mediante la agregación de las variables *ent_pais_trab*. La variable describe la zona geográfica en la que desarrolla su trabajo.
- Reemplazo de valores *numpy.nan* con el valor 0. Se utiliza el módulo *fillna* de Pandas.
- Se transforma el *dtype* de las variables *mun_id, mun_id_trab, factor* y *edad* a *int*.
- Creación del *DataFrame df_l*, el cual guarda transformaciones para los ids actuales de variables cualitativas, hacia id's de base común. Dichas transformaciones son obtenidas desde un archivo llamado *Intercensal Census IDs.xlsx*.
- Reemplazo de ids actuales para variables cualitativas por los ids de base común contenidos en el *DataFrame df_l*.
- Renombre de variables a inglés.

- Creación de la variable *filtered_age*, la cual es una variable binaria que denota el valor unitario cuando la edad es mayor o igual a 12 años, y cero en otro caso.
- Se asigna el valor *numpy.nan* a las variables en la lista *params_translated_add1*, cuando presentan un valor igual a 888888.
- Creación de la variable *year*, la cual define el periodo de realización del Censo.
- Reemplazo de valores en las columnas *state_of_birth* y *state_of_residency* por sus ids.
- Definición de la variable *country_of_residency_5_years*. Dicha variable toma el valor definido en la variable *state_of_residency* cuando su *dtype* es un *string*. En caso contrario, se le asigna el valor *numpy.nan*.
- Definición de la variable *state_of_residency_5_years*. Dicha variable toma el valor definido en la variable *state_of_residency* cuando su *dtype* no es un *string*. En caso de ser *string*, se le asigna el valor *numpy.nan*.
- Definición de la variable *country_of_birth*. Dicha variable toma el valor definido en la variable *state_of_birth* cuando su *dtype* es un *string*. En caso contrario, se le asigna el valor *numpy.nan*.
- Definición de la variable *state_of_birth*. Dicha variable toma el valor definido en la variable *state_of_birth* cuando su *dtype* no es un *string*. En caso de ser *string*, se le asigna el valor *numpy.nan*.
- Transformación de variables a objeto.
- Eliminación de la variable *state_of_residency*.
- Definición de la variable *foreign_migrant*. Dicha variable toma el valor unitario cuando la variable *country_of_residency_5_years* es diferente al valor *pd.isnull* (nulo), y cero en caso contrario.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, se obtiene un *DataFrame* con 95.983 filas y 49 columnas. A continuación puede ser visualizada un extracto de su estructura de salida:

sex	parent	sersalud	dhsersal1	laboral_condition	time_to_work	transport_mean_work	...	state_of_residency_5_years	country_of_birth	foreign_migrant
1	1	1	1	1.0	1.0	1.0	...	1.0	NaN	0
1	1	1	1	1.0	1.0	1.0	...	1.0	NaN	0
1	1	1	1	1.0	1.0	1.0	...	1.0	NaN	0
1	1	1	1	1.0	1.0	1.0	...	1.0	NaN	0
1	1	1	1	1.0	1.0	1.0	...	1.0	NaN	0
...	...	...	...	...	...	...	...	...	...	...

Figura X. Formato final tabla de datos cuestionario ampliado Censo Poblacional - Aguascalientes

9.4. Population_projection

Descripción General y Ejecución

El pipeline *population_projection.py* procesa los datos facilitados por el Consejo Nacional de Población (CONAPO) que toman relación con la proyección de la población. Los datos son descargados desde la plataforma del Consejo Nacional de Población, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/population_projection/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/population_projection/$ bamboo-cli --folder . --entry population_projection`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y, con respecto a su estructura, se identifican 1.100.736 filas y 9 columnas. A continuación, se visualiza la estructura inicial de los datos:

	RENGLON	CLAVE	CLAVE_ENT	NOM_ENT	MUN	SEXO	AÑO	EDAD_QUIN	POB
0	1	1001	1	Aguascalientes	Aguascalientes	Mujeres	2015	pobm_00_04	39403
1	2	1001	1	Aguascalientes	Aguascalientes	Mujeres	2016	pobm_00_04	39204
2	3	1001	1	Aguascalientes	Aguascalientes	Mujeres	2017	pobm_00_04	38891
3	4	1001	1	Aguascalientes	Aguascalientes	Mujeres	2018	pobm_00_04	38581
4	5	1001	1	Aguascalientes	Aguascalientes	Mujeres	2019	pobm_00_04	38272

Figura X. Formato original datos proyección de población

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de datos. Se utiliza la función *read_csv* de Pandas.
- Transformación de nombres de columnas a minúscula. Se utiliza la función *str.lower()*.
- Selección de columnas a utilizar: *clave, sexo, año, edad_quin* y *pob*.
- Renombre de columnas al inglés. Los nombres anteriores se transforman a: *mun_id, sex, year, age* y *population*.
- Reemplazo de valores en columnas *sex* y *age* por sus *id's*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 1.100.736 filas y 5 columnas . A continuación puede ser visualizada su estructura de salida:

	mun_id	sex	year	age	population
0	1001	2	2015	1	39403
1	1001	2	2016	1	39204
2	1001	2	2017	1	38891
3	1001	2	2018	1	38581
4	1001	2	2019	1	38272

Figura X. Formato final tabla de datos proyección de población

9.5. Inegi_housing

Descripción General y Ejecución

Para generar el cubo de datos *inegi_housing* se procesan los datos del 2010 y 2020 en pipelines separados debido a sus estructuras originales. Los pipeline trabajados son *housing_pipeline_2010.py* y *housing_pipeline_2020.py* y procesan los datos facilitados por el Instituto Nacional de Estadística y Geografía de México que toman relación con el cuestionario ampliado de viviendas del Censo Poblacional. Los datos, una vez descargados desde su sitio, son almacenados en Google Storage para ser, posteriormente, procesados mediante un proceso de ETL (extracción, transformación, y carga).

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

A) ETL para datos del 2010

Ejecución:

Para ejecutar el pipeline *housing_pipeline_2010.py* se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/inegi_intercensal_survey/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/inegi_intercensal_survey/$ bash housing_pipeline_2010.sh
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .dbf y su estructura inicial se compone de 16.572 filas y 56 columnas. A continuación, se visualiza la estructura inicial de los datos:

	ENT	NOM_ENT	MUN	NOM_MUN	LOC50K	...	ESTRATO	UPM	TAM_LOC	CERTEZA	IDH125
0	01	Aguascalientes	001	Aguascalientes	0000	...	400-01-001-0003	0002	01	0	0
1	01	Aguascalientes	001	Aguascalientes	0000	...	400-01-001-0003	0002	01	0	0
2	01	Aguascalientes	001	Aguascalientes	0000	...	400-01-001-0003	0002	01	0	0
3	01	Aguascalientes	001	Aguascalientes	0000	...	401-01-001-0001	0006	01	0	0
4	01	Aguascalientes	001	Aguascalientes	0000	...	401-01-001-0001	0023	01	0	0

Figura X. Formato original datos de vivienda Censo Económico 2010

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Selección de columnas a utilizar, las cuales se encuentran definidas en la lista *labels*.
- Transformación de *dtypes*. Esto se realiza, debido a la gran cantidad de variables, con el diccionario *dtypes*.
- Creación del *DataFrame* data, el cual guarda transformaciones para los *id's* actuales de variables cualitativas, hacia *id's* de base común. Dichas transformaciones son obtenidas desde un archivo llamado *Intercensal Census IDs.xlsx*.
- Creación de la variable *loc_id*, la cual se compone por la agregación de los siguientes id: *entidad*, *municipio* y *localidad*.
- Reemplazo de intervalos de ingreso por sus *id's*.
- Reemplazo de *id's* actuales para variables cualitativas por los *id's* de base común.
- Renombre de columnas a inglés.
- Creación de columna *year*, la cual define el periodo de tiempo en estudio.
- Se agregan con valor *numpy.nan*, todas aquellas variables nuevas en el Censo de Población y Vivienda de 2020, y la Encuesta Intercensal 2015, que no fueron definidas en el censo actual. Esto se realiza con motivo de poder generar un *dataset* que posea la agregación de todos los datos existentes.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, se obtiene un *DataFrame* con 15.893 filas y 40 columnas. A continuación puede ser visualizada su estructura de salida:

	loc_id	home_type	acquisition	wall	...	sex	age	national_financial_aid	retirement_financial_aid
0	10010000.0	1.0	1.0	2.0	...	NaN	NaN	NaN	NaN
1	10010000.0	1.0	1.0	7.0	...	NaN	NaN	NaN	NaN
2	10010000.0	1.0	1.0	7.0	...	NaN	NaN	NaN	NaN
3	10010000.0	1.0	1.0	7.0	...	NaN	NaN	NaN	NaN
4	10010000.0	1.0	1.0	7.0	...	NaN	NaN	NaN	NaN

Figura X. Formato final tabla de datos de vivienda Censo Económico 2010

B) ETL para datos del 2020

Ejecución:

Para ejecutar el pipeline *housing_pipeline_2010.py* se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/inegi_intercensal_survey/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

```
~/data_etl/etl/inegi_intercensal_survey/$ python housing_pipeline_2020.py
```

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv; y su estructura inicial se compone de 24.349 filas y 83 columnas. A continuación, se visualiza la estructura inicial de los datos:

	ENT	MUN	LOC50K	ID_VIV	COBERTURA	...	TIPOHOG	INGTRHOG	JEFE_SEXO	JEFE_EDAD	TAMLOC
0	1	1	1	10010000001	2	...	1	NaN	3	55	5
1	1	1	1	10010000002	2	...	1	30100.0	1	45	5
2	1	1	1	10010000003	2	...	5	20000.0	1	60	5
3	1	1	1	10010000004	2	...	1	66000.0	1	62	5
4	1	1	1	10010000005	2	...	1	25000.0	1	52	5

Figura X. Formato original datos de vivienda Censo Económico 2020

Limpieza y Transformación

Esta etapa sigue el siguiente proceso:

- Selección de columnas a utilizar, las cuales se encuentran definidas en las listas *labels* y *extra_labels*. Se agrega *extra_labels* por separado ya que son las variables del Censo que no fueron incluidas en el Censo previo, es decir, del año 2010.

- Transformación de *dtypes*. Esto se realiza, debido a la gran cantidad de variables, con el diccionario *dtypes*.
- Creación del *DataFrame* data, el cual guarda transformaciones para los *id's* actuales de variables cualitativas, hacia *id's* de base común. Dichas transformaciones son obtenidas desde un archivo llamado *Intercensal Census IDs.xlsx*.
- Creación de la variable *loc_id*, la cual se compone por la agregación de los siguientes *id's*: *entidad*, *municipio* y *localidad*.
- Reemplazo de intervalos de ingreso por sus id.
- Reemplazo de *id's* actuales para variables cualitativas por los id de base común definidos en el dataframe data.
- Renombre de columnas a inglés.
- Creación de columna *year*, la cual define el periodo de tiempo en estudio.

En cuanto a la salida del proceso de ETL, y considerando el ejemplo mencionado previamente, se obtiene un *DataFrame* con 24.347 filas y 40 columnas. A continuación puede ser visualizada su estructura de salida:

	loc_id	home_type	acquisition	wall	...	national_financial_aid	retirement_financial_aid	households	year
0	10010000.0	1.0	1.0	8.0	...	4.0	8.0	12.0	2020
1	10010000.0	1.0	1.0	8.0	...	4.0	8.0	11.0	2020
2	10010000.0	1.0	1.0	8.0	...	4.0	7.0	11.0	2020
3	10010000.0	1.0	1.0	8.0	...	4.0	8.0	11.0	2020
4	10010000.0	1.0	1.0	8.0	...	4.0	8.0	12.0	2020

Figura X. Formato final tabla de datos de vivienda Censo Económico 2020

10. Consejo Nacional de Evaluación de Política de Desarrollo Social (CONEVAL)

10.1. Coneval_gini_nat

Descripción General y Ejecución

El pipeline *gini_pipeline_nat.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con el cálculo del Coeficiente de Gini. En ello, el pipeline se refiere a los datos a nivel nacional. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry gini_pipeline_nat`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, con respecto a su estructura, se identifican 35 filas y 7 columnas. Es

importante señalar que del archivo inicial, se agregan en un mismo conjunto de datos las hojas de cálculo “Coeficiente de Gini 2008-2014” y “Coeficiente de Gini 2016-201”8, dado que contienen el Coeficiente Gini para diferentes años. A continuación, se visualiza la estructura inicial de los datos:

	Unnamed: 0	Entidad federativa	...	Unnamed: 5	Unnamed: 6
0	NaN	NaN	...	2012.000000	2014.000000
1	NaN	Aguascalientes	...	0.479191	0.486294
2	NaN	Baja California	...	0.464538	0.433568
3	NaN	Baja California Sur	...	0.492758	0.454270
4	NaN	Campeche	...	0.533018	0.499936

Figura X. Formato original datos GINI nivel nacional

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de las hojas de cálculo en el archivo que poseen información del Coeficiente Gini para diferentes periodos de tiempo. Dichos conjuntos de datos temporales son concatenados en un único *DataFrame*. Para ello, se utiliza la función *append* de Pandas.
- Selección de entidades con nombre Estados Unidos Mexicanos. De esta forma, se obtienen sólo los datos a nivel nacional.
- Reemplazo de la entidad Estados Unidos Mexicanos con su *id*.
- Transformación del tipo de dato para las columnas: *ent_id*, *year* y *gini*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 6 filas y 3 columnas . A continuación puede ser visualizada su estructura de salida:

	ent_id	year	gini
0	0	2008	0.505337
1	0	2010	0.508857
2	0	2012	0.497666
3	0	2014	0.503281
4	0	2016	0.498373
5	0	2018	0.468821

Figura X. Formato final tabla de datos GINI nivel nacional

10.2. Coneval_gini_ent

Descripción General y Ejecución

El pipeline *gini_pipeline_ent.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con el cálculo del Coeficiente de Gini. En ello, el pipeline se refiere a los datos a nivel de Entidad Federativa. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso de limpieza y transformación, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry gini_pipeline_ent`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, con respecto a su estructura, se identifican 35 filas y 7 columnas. Es importante señalar que del archivo inicial, se agregan en un mismo conjunto de datos las hojas de cálculo “Coeficiente de Gini 2008-2014” y “Coeficiente de Gini

2016-2018", dado que contienen el Coeficiente Gini para diferentes años. A continuación, se visualiza la estructura inicial de los datos:

	Unnamed: 0	Entidad federativa	...	Unnamed: 5	Unnamed: 6
0	NaN	NaN	...	2012.000000	2014.000000
1	NaN	Aguascalientes	...	0.479191	0.486294
2	NaN	Baja California	...	0.464538	0.433568
3	NaN	Baja California Sur	...	0.492758	0.454270
4	NaN	Campeche	...	0.533018	0.499936

Figura X. Formato original datos GINI nivel estatal

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de las diferentes hojas de cálculo que poseen datos para distintos periodos de tiempo.
- Selección de entidades cuyo nombre difiere a *Estados Unidos Mexicanos*. Esto permite eliminar datos a nivel nacional.
- Reemplazo de entidades por sus *id's*. Dicha operación se realiza haciendo uso del archivo *countries-EF-codes.xlsx*.
- Transformación del *dtype* para las columnas: *ent_id*, *year* y *gini*.

Posterior al proceso de transformación, obtenemos un *DataFrame* con 192 filas y 3 columnas . A continuación puede ser visualizada su estructura de salida:

	ent_id	year	gini
0	1	2008	0.515696
1	2	2008	0.453242
2	3	2008	0.495741
3	4	2008	0.523707
4	5	2008	0.469642

Figura X. Formato final tabla de datos GINI nivel estatal

10.3. Coneval_gini_mun

Descripción General y Ejecución

El pipeline *gini_pipeline_mun.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con el cálculo del Coeficiente de Gini. En ello, el pipeline se refiere a los datos a nivel de municipio. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso de limpieza y transformación, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry gini_pipeline_mun`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, con respecto a su estructura, se identifican 2.463 filas y 7 columnas. Es importante señalar que del archivo inicial, se agregan en un mismo conjunto de datos las hojas de cálculo 2010 y 2015, dado que contienen el Coeficiente Gini para diferentes años. A continuación, se visualiza la estructura inicial de los datos:

	Unnamed: 0	Clave de entidad	...	Coeficiente de Gini	Razón de ingreso 1
0	NaN	NaN	...	NaN	NaN
1	NaN	NaN	...	NaN	NaN
2	NaN	01	...	0.426187	0.117213
3	NaN	01	...	0.442576	0.119099
4	NaN	01	...	0.426041	0.117333

Figura X. Formato original datos GINI nivel municipal

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de hojas de cálculo que poseen datos a nivel municipal.
- Selección de datos cuyo valor en la variable *Clave de municipio* difiere a *NaN*.
- Selección de columnas a utilizar: *Clave de municipio*, *Coeficiente de Gini* y *Razón de ingreso 1*.
- Creación de la columna *year*. Dicha columna se refiere al periodo de tiempo en estudio medido en años.
- Renombre de columnas a: *mun_id*, *gini* y *income_rate*.

Posterior al proceso de transformación, obtenemos un *DataFrame* con 4.913 filas y 4 columnas . A continuación puede ser visualizada su estructura de salida:

	mun_id	gini	income_rate	year
2	1001	0.426187	0.117213	2010
3	1002	0.442576	0.119099	2010
4	1003	0.426041	0.117333	2010
5	1004	0.427528	0.122823	2010
6	1005	0.449637	0.125425	2010

Figura X. Formato final tabla de datos GINI nivel municipal

10.4. Coneval_poverty

Descripción General y Ejecución

El pipeline *poverty_pipeline.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con la medición de la pobreza a nivel municipal. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso de limpieza y transformación, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry poverty_pipeline --year=<year>`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y, con respecto a su estructura, se identifican 2.456 filas y 37 columnas. A continuación, se visualiza la estructura inicial de los datos:

	clave_entidad	entidad_federativa	clave_municipio	municipio	...	plb	plb_pob	plbm	plbm_pob
0	1	Aguascalientes	1001	Aguascalientes	...	39.2	314,300	9.9	79,703
1	1	Aguascalientes	1002	Asientos	...	70.8	34,247	33.8	16,326
2	1	Aguascalientes	1003	Calvillo	...	71.9	41,428	33.9	19,519
3	1	Aguascalientes	1004	Cosío	...	59.9	8,950	22.3	3,331
4	1	Aguascalientes	1005	Jesús María	...	49.8	50,888	16.9	17,287

Figura X. Formato original datos de pobreza

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura del archivo .csv.
- Renombre de columnas a inglés.
- Creación de la columna *year*, la cual hace referencia al periodo de estudio.
- Reemplazo de *caracteres* no deseados: comas (,) y textos (n.d).

Posterior al proceso de transformación, se obtiene un *DataFrame* con 2.456 filas y 19 columnas . A continuación puede ser visualizada su estructura de salida:

	mun_id	population	poverty	...	income_below_welfare_line	income_below_min_welfare_line	year
0	1001	801807.0	242317.0	...	314300.0	79703.0	2010
1	1002	48358.0	31694.0	...	34247.0	16326.0	2010
2	1003	57627.0	39419.0	...	41428.0	19519.0	2010
3	1004	14929.0	8091.0	...	8950.0	3331.0	2010
4	1005	102211.0	43315.0	...	50888.0	17287.0	2010

Figura X. Formato final tabla de datos de pobreza

10.5. Coneval_social_lag_ent

Descripción General y Ejecución

El pipeline *social_lag_pipeline_ent.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con el Índice de Rezago Social. En ello, el presente pipeline procesa datos a nivel de Entidad Federativa. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso de limpieza y transformación, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry social_lag_pipeline_ent --year=<year>`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, con respecto a su estructura, se identifican 39 filas y 17 columnas. A continuación, se visualiza la estructura inicial de los datos:

	Clave entidad	...	Lugar que ocupa en el contexto nacional
0	NaN	...	NaN
1	00	...	NaN
2	01	...	29.0
3	02	...	26.0
4	03	...	18.0

Figura X. Formato original datos del Índice de Rezago Social a nivel estatal

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de datos. Se utiliza la hoja de cálculo “Estados”.
- Selección de entidades que no poseen valores *NaN*, o sus *id’s* poseen el valor 00.
- Renombre de columnas a inglés.
- Reemplazo de valores en la variable *social_lag_degree* por sus *id’s*.
- Eliminación de variables no utilizadas: *ent_name*, y *place_in_country*.
- Creación de columna *year*, la cual hace referencia al periodo en estudio.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 32 filas y 16 columnas . A continuación puede ser visualizada su estructura de salida:

	ent_id	population	illiterate_population	...	social_lag_index	social_lag_degree	year
0	1	1425607	2.108761	...	-1.101561	1	2020
1	2	3769020	1.821198	...	-0.642387	2	2020
2	3	798447	2.330508	...	-0.317027	2	2020
3	4	928363	5.854886	...	0.244535	3	2020
4	5	3146771	1.666687	...	-1.147587	1	2020

Figura X. Formato final tabla de datos del Índice de Rezago Social a nivel estatal

10.6. coneval_social_lag_mun

Descripción General y Ejecución

El pipeline *social_lag_pipeline_mun.py* procesa los datos facilitados por el Consejo Nacional de Evaluación de la Política de Desarrollo Social (CONEVAL) que toman relación con el Índice de Rezago Social. En ello, el presente pipeline procesa datos a nivel de municipio. Los datos son descargados desde la plataforma del Consejo Nacional de Evaluación de la Política de Desarrollo Social, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso de limpieza y transformación, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/coneval/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando la siguiente forma:

(Bamboo-cli) `~/data_etl/etl/coneval/$ bamboo-cli --folder . --entry social_lag_pipeline_mun --year=<year>`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .xlsx y, con respecto a su estructura, se identifican 2.475 filas y 19 columnas. A continuación, se visualiza la estructura inicial de los datos:

	Clave entidad	...	Lugar que ocupa en el contexto nacional
0	NaN	...	NaN
1	NaN	...	NaN
2	01	...	2435.0
3	01	...	1986.0
4	01	...	2076.0

Figura X. Formato original datos del Índice de Rezago Social a nivel municipal

Lectura y Transformación

Esta etapa sigue el siguiente proceso:

- Lectura de datos. Se utiliza la hoja de cálculo “Municipios”.
- Selección de municipios que no poseen valores *NaN*, o sus *id’s* poseen el valor *Clave*.
- Renombre de columnas a inglés.
- Reemplazo de valores en la variable *social_lag_degree* por sus id.
- Creación de columna *year*, la cual hace referencia al periodo en estudio.
- Eliminación de variables no utilizadas: *ent_id*, *ent_name*, *mun_name* y *place_in_country*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 2.469 filas y 16 columnas . A continuación puede ser visualizada su estructura de salida:

	mun_id	population	illiterate_population	...	social_lag_index	social_lag_degree	year
0	1001	948990.0	1.642183	...	-1.315320	1	2020
1	1002	51536.0	3.523404	...	-0.857301	1	2020
2	1003	58250.0	4.487288	...	-0.918554	1	2020
3	1004	17000.0	3.139545	...	-1.004023	1	2020
4	1005	129929.0	2.377387	...	-1.173361	1	2020

Figura X. Formato final tabla de datos del Índice de Rezago Social a nivel municipal

11. Seguridad Pública

11.1. Inegi_envipe

Descripción General y Ejecución

Este pipeline *envipe_pipeline.py* procesa los datos facilitados por el Instituto Nacional de Estadística y Geografía (INEGI) que toman relación con la Encuesta Nacional de Victimización y Percepción de Seguridad Pública (ENVIPE). Los datos de dicha encuesta son descargados desde la plataforma del Instituto Nacional de Estadística y Geografía, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/envipe/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

```
~/data_etl/etl/envipe/$ source envipe_ingest.sh
```

(bamboo-cli) `~/data_etl/etl/envipe/$ bamboo-cli --folder . --entry envipe_pipeline --year=<year_value>`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .dbf por lo tanto, posterior a su organización en un *DataFrame* de la librería

Pandas, se identifican 186 columnas y más de 90.000 filas. A continuación, se visualiza un estracto de la estructura inicial de los datos:

ID_VIV	ID_HOG	ID_PER	UPM	VIV_SEL	HOGAR	RESUL_H	R_SEL	SEXO	EDAD	...	AP5_6_10	AP5_8	FAC_HOG	FAC_ELE	FAC_HO
100013.01	0100013.01.01	0100013.01.01.01	100013	1	1	B	1	2	53	...	NaN	4	184	184	
100013.02	0100013.02.01	0100013.02.01.02	100013	2	1	A	2	2	42	...	3.0	3	184	368	
100013.03	0100013.03.01	0100013.03.01.01	100013	3	1	B	1	2	44	...	3.0	4	184	184	
100013.04	0100013.04.01	0100013.04.01.02	100013	4	1	B	2	2	62	...	NaN	2	184	551	
100013.05	0100013.05.01	0100013.05.01.03	100013	5	1	B	3	1	19	...	NaN	2	184	551	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3260731.17	3260731.17.01	3260731.17.01.01	3260731	17	1	B	1	1	75	...	4.0	1	246	491	
3260731.18	3260731.18.01	3260731.18.01.01	3260731	18	1	A	1	2	26	...	NaN	3	246	246	

Figura X. Formato original datos de ENVIPE

Lectura

En esta etapa es donde se transforma el archivo .dbf a un *DataFrame* de Pandas, y además, se estandarizan los nombres de las columnas a *caracteres* en minúscula.

Transformación

En esta etapa se sigue el siguiente proceso:

- Seleccionar las columnas relevantes para el análisis de datos.
- Modificar el nombre de las columnas en el *DataFrame*, y también, el *dtype* de: *homes_factor, people_factor, expenses_in_protection_against_crime*.
- Reemplazar los valores en la columna *expenses_in_protection_against_crime* por su *id*. Este proceso se lleva a cabo leyendo un archivo .xlsx desde Google Drive que contiene el nivel de gasto en protección para diferentes niveles de ingreso (*income*).
- Modificación de la columna *mun_id* que toma relación con la geografía a nivel municipal de los datos. En ello, se agrega el *id* del estado en estudio.
- Reorganizar el orden de las columnas en el *DataFrame*.
- Creación de una nueva columna que hace referencia al año llamada *year*.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 23 columnas y más de 90.000 filas. A continuación puede ser visualizada su estructura de salida:

security_perception_in_their_state	sociodemographic_stratum	expenses_in_protection_against_crime	neighbors_trust	coworkers_trust	family_trust	friends_trust	t
1.0	1.0	1.0	1.0	1.0	1.0	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	
1.0	1.0	1.0	1.0	1.0	1.0	1.0	

Figura X. Formato final tabla de datos de ENVIPE

11.2. Sensnsp_crimes

Descripción General y Ejecución

El pipeline *crimes_pipeline.py* procesa los datos facilitados por el Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública (SESNSP) que toman relación con los reportes de incidencia delictiva. Los datos son descargados desde la plataforma del Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública, y son almacenados en Google Storage para su posterior procesamiento.

Para llevar a cabo el proceso mencionado anteriormente, se utiliza el lenguaje de programación Python, donde son fundamentales las librerías pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

Antes de ingestar los datos en una base de datos de *Clickhouse*, estos son llamados desde Google Storage para ser sometidos a un proceso de limpieza y transformación.

Ejecución

Para ejecutar el pipeline se debe activar un entorno virtual de Python 3 que cuente con las librerías requeridas:

```
~/data_etl/etl/crime/$ source venv/bin/activate
```

Luego se puede ejecutar el pipeline utilizando una de estas dos formas:

```
~/data_etl/etl/crime/$ source crimes_ingest.sh
```

(bamboo-cli) `~/data_etl/etl/crime/$ bamboo-cli --folder . --entry crimes_pipeline`

Descarga de Datos

Utilizando el módulo de descarga que facilita la librería *bamboo-lib* se procede a descargar la data desde Google Storage. Cabe destacar que la data se encuentra en formato .csv y se identifican 1.346.618 filas y 21 columnas. A continuación, se visualiza un extracto de la estructura inicial de los datos:

	Año	Clave_Ent	Entidad	Cve. Municipio	Municipio	Bien jurídico afectado	Tipo de delito	Subtipo de delito	Modalidad	Enero	...	Marzo	Abril	Mayo	Junio	Ju
0	2015	1	Aguascalientes	1001	Aguascalientes	La vida y la Integridad corporal	Homicidio	Homicidio doloso	Con arma de fuego	2	...	1	1	0	1	
1	2015	1	Aguascalientes	1001	Aguascalientes	La vida y la Integridad corporal	Homicidio	Homicidio doloso	Con arma blanca	1	...	0	0	0	1	
2	2015	1	Aguascalientes	1001	Aguascalientes	La vida y la Integridad corporal	Homicidio	Homicidio doloso	Con otro elemento	0	...	1	1	3	2	

Figura X. Formato original datos del SESNSP

Lectura y Transformación

En esta etapa se desarrollan los siguientes pasos:

- Eliminación de columnas no relevantes en el proceso de ETL.
- Renombre de columnas. En ello, para hacer renombre de las columnas que hacen referencia a los meses se utiliza un diccionario definido como *dict_months*.
- Transformación del *DataFrame* mediante la función *melt* de Pandas. Con ello, se obtiene un *DataFrame* en formato *tidy*.
- Creación de la columna *month_id*. Dicha columna se crea mediante la agregación del año y el mes en estudio.
- Reemplazo de valores en la columna *crime_modality*. Dichos valores se reemplazan con el *id* del tipo de crimen. El *id* se obtiene desde un Google Spreadsheet, y se utiliza la función *replace* de Pandas para llevar a cabo el proceso.

Posterior al proceso de transformación, se obtiene un *DataFrame* con 16.159.414 filas y 5 columnas. A continuación puede ser visualizada su estructura de salida:

	mun_id	crime_subtype_id	crime_modality_id	value	month_id
0	1001	50302	8	2	201501
1	1001	50302	7	1	201501
2	1001	50302	9	0	201501
3	1001	50302	27	1	201501
4	1001	50301	8	0	201501

Figura X. Formato final tabla de datos del SESNSP