DOCUMENTACIÓN CUBOS DE DATOS DISPONIBLES EN DATAMÉXICO

Descripción de limpieza y transformaciones de datos

Versión 1

(16 de marzo de 2021)

SE SECRETARÍA DE ECONOMÍA | Data MÉXICO

Datawheel

TABLA DE CONTENIDO

1. Créditos

1.1. imss_credits

Descripción General y Ejecución

Este pipeline procesa los datos a nivel estatal y municipal de los créditos otorgados por el Instituto Mexicano de Seguro Social (IMSS). Los datos son descargados desde Google Storage, se someten a un proceso de limpieza y transformación, y luego son ingestados en una base de datos ClickHouse siguiendo un modelo relacional para formar el cubo de datos. Para llevar a cabo este proceso de ETL, se utiliza el lenguaje de programación Python, la librería pandas y dos librerías desarrolladas por Datawheel: bamboo-lib y dw-bamboo-cli.

Descarga de Datos

En este paso se obtienen los archivos en formato .csv de los créditos otorgados a nivel estatal y municipal. 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 donde se descargan para ser procesados. Los datos vienen en el formato presentado en las figuras 1 y 2.

	ENTIDAD	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antig�edad	conteo_anonimizado
0	AGUASCALIENTES	H	fisica	De 1 a 10	19 a�os o menos	C
1	AGUASCALIENTES	H	fisica	De 1 a 10	20 a 29 a�os	104
2	AGUASCALIENTES	H	fisica	De 1 a 10	30 a 39 a�os	296
3	AGUASCALIENTES	H	fisica	De 1 a 10	40 a 49 a�os	311
4	AGUASCALIENTES	H	fisica	De 1 a 10	50 a 59 a�os	259
...	...	...	...	...	...	...
1627	ZACATECAS	M	fisica	De 21 a 50	40 a 49 a�os	C
1628	ZACATECAS	M	moral	De 1 a 10	20 a 49 a�os	13
1629	ZACATECAS	M	moral	De 1 a 10	50 a 69 a�os	21
1630	ZACATECAS	M	moral	De 11 a 20	50 a 69 a�os	C
1631	ZACATECAS	M	moral	Mas de 50	20 a 49 a�os	C

Figura 1. Formato original datos de créditos a nivel estatal

	ENTIDAD	MUNICIPIO	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antig�edad	conteo_anonimizado
0	AGUASCALIENTES	AGUASCALIENTES	H	fisica	De 1 a 10	19 a�os o menos	C
1	AGUASCALIENTES	AGUASCALIENTES	H	fisica	De 1 a 10	20 a 29 a�os	87
2	AGUASCALIENTES	AGUASCALIENTES	H	fisica	De 1 a 10	30 a 39 a�os	265
3	AGUASCALIENTES	AGUASCALIENTES	H	fisica	De 1 a 10	40 a 49 a�os	272
4	AGUASCALIENTES	AGUASCALIENTES	H	fisica	De 1 a 10	50 a 59 a�os	230
...	...	...	...	...	...	...	...
14260	ZACATECAS	ZACATECAS	M	fisica	De 11 a 20	60 y m�s a�os	C
14261	ZACATECAS	ZACATECAS	M	moral	De 1 a 10	20 a 49 a�os	6
14262	ZACATECAS	ZACATECAS	M	moral	De 1 a 10	50 a 69 a�os	8
14263	ZACATECAS	ZACATECAS	M	moral	De 11 a 20	50 a 69 a�os	C
14264	ZACATECAS	ZACATECAS	M	moral	Mas de 50	20 a 49 a�os	C

Figura 2. Formato original datos de créditos a nivel municipal

Lectura

En este paso se modifica el nombre de las columnas en el DataFrame que se genera en el paso anterior. Además, se añade una columna 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 un DataFrame en formato tidy, listo para ser ingestado en la base de datos (figura 3).

	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 3. Formato final tabla de datos de créditos

1.2. households

Descripción General y Ejecución

El presente pipeline 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

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, posee 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
0	1	Aguascalientes	H	Física	NaN	20 a 29 años	6
1	1	Aguascalientes	H	Física	NaN	30 a 39 años	18
2	1	Aguascalientes	H	Física	NaN	40 a 49 años	51
3	1	Aguascalientes	H	Física	NaN	50 a 59 años	64
4	1	Aguascalientes	H	Física	NaN	60 años o más	92

Figura 4. Formato original datos de créditos a trabajadores del hogar

Lectura

En esta etapa se procede a modificar los nombres de las columnas en el DataFrame. Además, se define una nueva variable para describir 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 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 5. Formato final tabla de datos de créditos para trabajadores del hogar

1.3. Wellness

Descripción General y Ejecución

Este pipeline 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería pandas y dos librerías desarrolladas por Datawheel: *bamboo-lib* y *dw-bamboo-cli*.

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, posee 738 filas y 7 columnas, y su formato inicial se muestra a continuación:

	CLAVE	ENTIDAD	genero	tipo_persona	TAMANIO_PATRON	rango_edad_antigüedad	conteo_anonimizado
0	1	Aguascalientes	H	Física	c	19 años o menos	2
1	1	Aguascalientes	H	Física	c	20 a 29 años	476
2	1	Aguascalientes	H	Física	c	30 a 39 años	867
3	1	Aguascalientes	H	Física	c	40 a 49 años	1073
4	1	Aguascalientes	H	Física	c	50 a 59 años	978

Figura 6. Formato inicial datos de créditos a micronegocios

Lectura

En esta etapa se procede inicialmente a modificar los nombres de las columnas en el DataFrame. Además, se define una nueva variable para describir 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. A continuación, puede ser visualizada su estructura:

	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 7. Formato final tabla de datos de créditos para micronegocios

2. Seguridad Pública

2.1. Inegi_envipe

Descripción General y Ejecución

Este pipeline 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 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 8. 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 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 9. Formato final tabla de datos de envipe

2.2. crimes_pipeline

Descripción General y Ejecución

El presente pipeline 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 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 10. Formato original datos de seguridad pública

Lectura y Transformación

En esta etapa se desarrollan los siguientes pasos:

- Lectura del archivo de incidencia delictiva desde Google Storage.
- 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 11. Formato final tabla de datos de seguridad pública

3. Censo Económico

3.1. pipeline_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 involucrado total, 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*.

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 de Python. Al momento en el que se elaboró esta documentación, el DataFrame inicial cuenta con 437.990 filas y 192 columnas. La siguiente imagen 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 perifi¿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 12. 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 correspondes 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" 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 la base de datos.

Al momento en el que se elaboró esta documentación, el DataFrame final contenía 426.563 filas y 101 columnas. La siguiente imagen muestra algunas filas y columnas de éste.

	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 13. Formato final tabla de datos Censo Económico

3.2. 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 entregados por INEGI (Instituto nacional de estadística y geografía). Estos datos incluyen cantidades asociadas a las actividades económicas por entidad federativa y municipio, tales como; unidades económicas (UE), producción bruta total, depreciación total de activos fijos, entre otros. En este punto es importante destacar que los datos del Censo Económico son anonimizados en determinados niveles por principios de anonimización. **En este conjunto de datos, si bien se encuentran los datos a nivel estatal, estos son una agrupación de los municipios que los componen, por ende, si un municipio ha sido anonimizado el total de la Entidad Federativa se verá afectado. Por ello se recomienda trabajar con los valores a nivel municipal.**

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 Entidad Federativa. En este pipeline en específico, se desagregan los datos por cada municipio.

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. Para la Entidad Federativa 28, al momento en el que se elaboró esta documentación, el DataFrame inicial contaba con 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 14. Formato original datos Censo Económico 2014

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 estudio, que en este caso es 2014.

Al momento en el que se elaboró esta documentación, para la Entidad Federativa 28, el DataFrame final contenía 7.124 filas y 101 columnas. La siguiente imagen muestra las primeras filas y algunas columnas de éste.

	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 15. Formato final tabla Censo Económico 2014

3.3. 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 entregados por INEGI (Instituto nacional de estadística y geografía). Estos datos incluyen cantidades asociadas a las actividades económicas por entidad federativa, tales como; unidades económicas (UE), producción bruta total, depreciación total de activos fijos, 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.

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 el estado de turno, estos se leen y almacenan en un DataFrame de la librería pandas. Para la entidad federativa 28, al momento en el que se elaboró esta documentación, el DataFrame inicial contaba con 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 16. Formato original datos Censo Económico 2014 por entidad federativa

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 estudio, que en este caso es 2014.

Al momento en el que se elaboró esta documentación, para la entidad federativa 28, el DataFrame final contiene 1.960 filas y 101 columnas. La siguiente imagen muestra las primeras filas y algunas columnas de éste.

	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 17. Formato final tabla de datos Censo Económico 2014 por entidad federativa

3.4. inegi_economic_census_sex

Descripción general y ejecución

El pipeline del censo económico (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). Estos datos incluyen cantidades asociadas a las actividades económicas, tales como; unidades económicas (UE), personal involucrado total, Inversión total, entre otros, desagregados por sexo. 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*.

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 tiene 437.990 filas y 192 columnas. La siguiente imagen 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 perifi¿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 18. Formato original datos Censo Económico 2014 con desagregación 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",* la columna original *H101B* 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, entro otros.

Al momento en el que se elaboró esta documentación, el DataFrame final contaba con 853.126 filas y 13 columnas. La siguiente imagen muestra algunas filas y algunas columnas de éste.

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 19. Formato final tabla de datos Censo Económico 2014 desagregada por sexo

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

4.1. enigh_household_income_pipeline

Descripción General y Ejecución

El presente 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 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 20. Formato original datos de ingresos de los hogares

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 en estudio.
- 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.
- 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 reemplazar el *monthly_wage* por su *id*.
- 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. A continuación, puede ser visualizada su estructura de salida:

	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 21. Formato final tabla de datos de ingresos de los hogares

4.2. enigh_jobs_pipeline

Descripción General y Ejecución

El presente 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 serán utilizados tres archivos: Empleo, Vivienda, y Población. Con respecto a su estructura, 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 22. Formato original datos de empleo en los hogares

	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 23. Formato original datos de vivienda

	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 24. Formato original datos de población

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 DataFrame) 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 que toma relación 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 se hace para 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. A continuación, puede ser visualizada su estructura de salida:

	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 25. Formato final tabla de datos de trabajas en los hogares

4.3. enigh_population_pipeline

Descripción General y Ejecución

El presente 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 serán utilizados dos archivos con referencia a: Población y Hogar. Con respecto a su estructura, 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 26. 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 27. Formato original datos de población 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_aa* y *ss_mm*.
- Reemplazo de valores por *pd.np.nan* para las personas que son menores de edad o no se han adherido al seguro social.
- 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* a *np.nan*.
- Creación de nueva columna *year*, considerando el año de estudio.
- Cambio de *dtype* para columnas específicas.

Posterior al proceso de transformación, obtenemos un DataFrame con 86.348 filas y 35 columnas. A continuación, puede ser visualizada su estructura de salida:

	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 28. Formato final tabla de datos de población ENIGH

5. Ocupaciones y empleo

5.1. enoe_pipeline

Descripción general y ejecución

El pipeline 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*.

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 encuesta 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 imágenes de 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 29. Formato original datos archivo coe1t ENOE

	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 30. Formato original datos archivo coe1t ENOE

	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 31. Formato original datos archivo coe1t ENOE

	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 32. Formato original datos archivo coe1t ENOE

Al momento de la elaboración de esta documentación, el DataFrame inicial proveniente del archivo *coe1t* contaba con 312.167 filas por 23 columnas, el DataFrame proveniente del archivo *coe2t* contenía 312.167 filas y 14 columnas, el DataFrame proveniente del archivo *sdmet* contenía 406.797 filas por 24 columnas y el DataFrame proveniente de *vivt* contaba con 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 de estos 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 es que si posee trabajo o negocio y el *2* si es que 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 siguiente imagen muestra algunas filas y algunas columnas de éste.

	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 33. Formato final tabla de datos de ENOE

5.2. etoe_pipeline

Descripción general y ejecución

El pipeline 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. con los datos.

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*.

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 imágenes de 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 34. Formato original datos archivo coe1t ETOE

	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 35. Formato original datos archivo coe2t ETOE

	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 36. Formato original datos archivo sdmet ETOE

	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 37. Formato original datos archivo vivt ETOE

Al momento de la elaboración de esta documentación, el DataFrame inicial proveniente del archivo *coe1t* contiene 23.893 filas por 22 columnas. El DataFrame proveniente del archivo *coe2t* contiene 23.893 filas por 14 columnas. El DataFrame proveniente del archivo *sdmet* contiene 29.704 filas por 24 columnas y el DataFrame proveniente de *vivt* contiene 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 de estos 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 21.315 filas y 42 columnas. La siguiente imagen muestra algunas filas y algunas columnas de éste.

	mun_id	represented_city	age	has_job_or_business	search_job_overseas	\
0	01001	14.0	15.0	2.0	NaN	
1	01001	14.0	15.0	NaN	NaN	
2	01001	14.0	16.0	2.0	NaN	
3	01001	14.0	16.0	2.0	NaN	
4	01001	14.0	16.0	2.0	NaN	

	search_job_mexico	search_start_business	search_no_search	\
0	NaN	NaN	1.0	
1	NaN	NaN	NaN	
2	NaN	NaN	1.0	
3	NaN	NaN	1.0	
4	NaN	NaN	1.0	

	search_no_knowledge	search_job_year	...	work_history	\
0	NaN	NaN	...	0.0	
1	NaN	NaN	...	0.0	
2	NaN	NaN	...	0.0	
3	NaN	NaN	...	0.0	
4	NaN	NaN	...	0.0	

	unoccupied_condition	classification_duration_unemployment	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	underemployed_population	underemployed_classification	\
0	0.0	0.0	
1	1.0	3.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	classification_self_employed_unqualified_activities	\
0	0.0	
1	0.0	
2	0.0	
3	0.0	
4	0.0	

	classification_formal_informal_jobs_first_activity	population	income_id	\
0	0.0	2534	NaN	
1	1.0	4362	4.0	
2	0.0	3389	NaN	
3	0.0	4352	NaN	
4	0.0	3492	NaN	

	month_id
0	202004
1	202004
2	202004
3	202004
4	202004

Figura 38. Formato final tabla de datos de ETOE

6. Salud

6.1. imss_pipeline

Descripción General y Ejecución

El presente pipeline 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 ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 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 39. Formato original datos de asegurados IMMS

Lectura y Transformación

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

- Lectura de los datos proveídos por el IMSS. 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 obtenemos 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 por 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 creados 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, obtenemos 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 40. Formato final tabla de datos de asegurados del IMSS

6.2. pregnancy_mortality_pipeline

Descripción General y Ejecución

El presente pipeline procesa los datos facilitados por la Dirección General de Información en Salud (DGIS) que toman relación con Mortalidad Materna. Los datos son descargados desde la plataforma de la Dirección General de Información en Salud, y son almacenados en Google Storage para su posterior procesamiento.

Para ejecutar este proceso de ETL, se utiliza el lenguaje de programación Python, la librería 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.

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 13 columnas. A continuación, se visualiza la estructura inicial de los datos

EDAD CUMPLIDA	ESTADO CONYUGAL	ENTIDAD DE RESIDENCIA	MUNICIPIO DE RESIDENCIA	OCUPACIÓN HABITUAL	ESCOLARIDAD	DERECHOHABIENCIA	ENTIDAD DE OCURRENCIA	MUNICIPIO DE OCURRENCIA	SITIO DONDE OCURRIO LA DEFUNCIÓN	AÑO DE L DEFUNCIÓ
25	5	10	35	0	3	0	10	35	12	19
37	4	30	87	2	2	1	30	87	8	19
33	5	10	12	0	0	0	10	12	0	19
35	5	12	47	2	3	1	12	47	11	19
30	5	10	4	2	2	1	5	35	9	19

Figura 41. 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 subconjuntos. Esto se realiza cuando existen grandes volúmenes de datos y no queremos colapsar la memoria. Además, se agrega el parámetro *usecols* para seleccionar las columnas relevantes en este proceso de ETL.
- 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 vive la persona involucrada en el echo.
- 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 de *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*.

Posterior al proceso de transformación, obtenemos un DataFrame con 19.054 filas y 12 columnas. A continuación, puede ser visualizada su estructura de salida:

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 42. Formato final tabla de datos de mortalidad materna

6.3. covid_pipeline

Descripción general y ejecución

El pipeline de datos de la situación mexicana frente al COVID-19 es una herramienta de ETL (Extracción, Transformación y Carga de datos). Estos datos ofrecen información respecto de los casos de contagios reportados en México, considerando tanto datos geográficos como datos de la persona contagiada. 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*

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 se pone 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 5.685.052 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 la primeras filas y algunas columnas del DataFrame inicial.

	FECHA_ACTUALIZACION	ID_REGISTRO	ORIGEN	SECTOR	ENTIDAD_UM	SEXO
0	2021-03-10	z482b8	1	12	9	2
1	2021-03-10	z49a69	1	12	23	1
2	2021-03-10	z23d9d	1	12	22	2
3	2021-03-10	z24953	1	12	9	1
4	2021-03-10	zz8e77	1	12	9	2

ENTIDAD_NAC	ENTIDAD_RES	...	RESULTADO_LAB	TOMA_MUESTRA_ANTIGENO	RESULTADO_ANTIGENO
9	9	...	97	2	97
23	23	...	97	2	97
24	22	...	97	2	97
9	9	...	2	2	97
9	9	...	97	2	97

CLASIFICACION_FINAL	MIGRANTE	PAIS_NACIONALIDAD	PAIS_ORIGEN	UCI
1	99	MÃ©xico	97	97
2	99	MÃ©xico	97	1
6	99	MÃ©xico	97	97
7	99	MÃ©xico	97	97
6	99	MÃ©xico	97	97

Figura 43. Formato inicial datos de 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*.

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* indicia si el paciente dio positivo al covid, *2* si no fue positivo y *3* si el resultado se encuentra pendiente (caso sospechoso).

Al momento en el que se elaboró esta documentación, el DataFrame final contenía 5.685.052 filas y 41 columnas (y continúa aumentando día a día). La siguiente imagen muestra algunas filas y algunas columnas de éste.

	updated_date	origin	type_health_institution_attended	health_institution_attended_ent	sex	patient_origin_ent_id
0	20210310	1	12	9	1	9
1	20210310	1	12	23	2	23
2	20210310	1	12	22	1	24
3	20210310	1	12	9	2	9
4	20210310	1	12	9	1	9

patient_type	...	country_nationality	country_origin	required_ICU	patient_residence_mun_id	is_dead	id	time_id
1	...	mex	xxa	97	9012	0	1	20210310
2	...	mex	xxa	1	23004	1	2	20210310
1	...	mex	xxa	97	22009	0	3	20210310
1	...	mex	xxa	97	9010	0	4	20210310
1	...	mex	xxa	97	9002	0	5	20210310

Figura 44. Formato final tabla de datos de mortalidad materna

6.4. covid_stats_state

Descripción general y ejecución

El pipeline de estadísticas estatales de la situación mexicana frente al COVID-19 es una herramienta de ETL (Extracción, Transformación y Carga de datos). Estos datos ofrecen información respecto de los casos de contagios reportados en cada estado de México considerando tanto datos geográficos como datos de la persona contagiada. 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. y generar contenido con los datos.

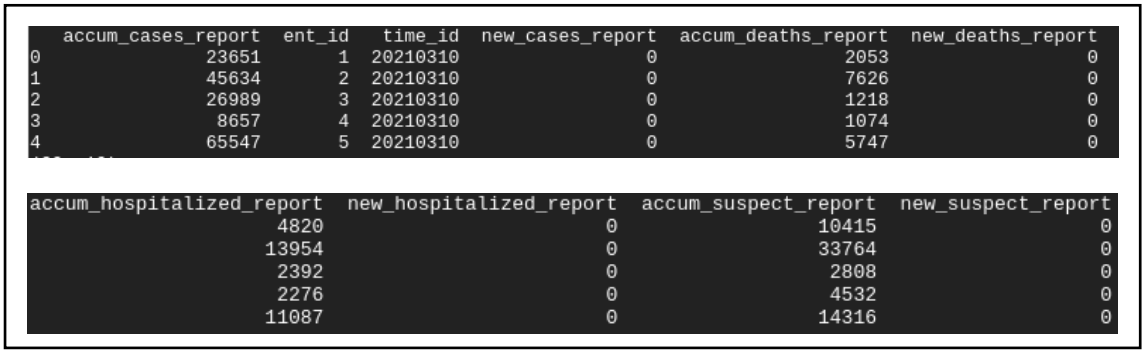
Cabe destacar que este pipeline es necesario para obtener las métricas correctas a diferentes niveles geográficos, en este caso, para cada entidad federativa. Al momento de ingestar los datos, cuando hay diferentes niveles geográficos estos se agrupan desde los niveles más pequeños a los niveles superiores 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.*

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: casos reportados, fallecidos, hospitalizados y casos sospechosos. 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 contenía 32 filas por 10 columnas. La siguiente imagen muestra la primeras filas y algunas columnas del DataFrame inicial.

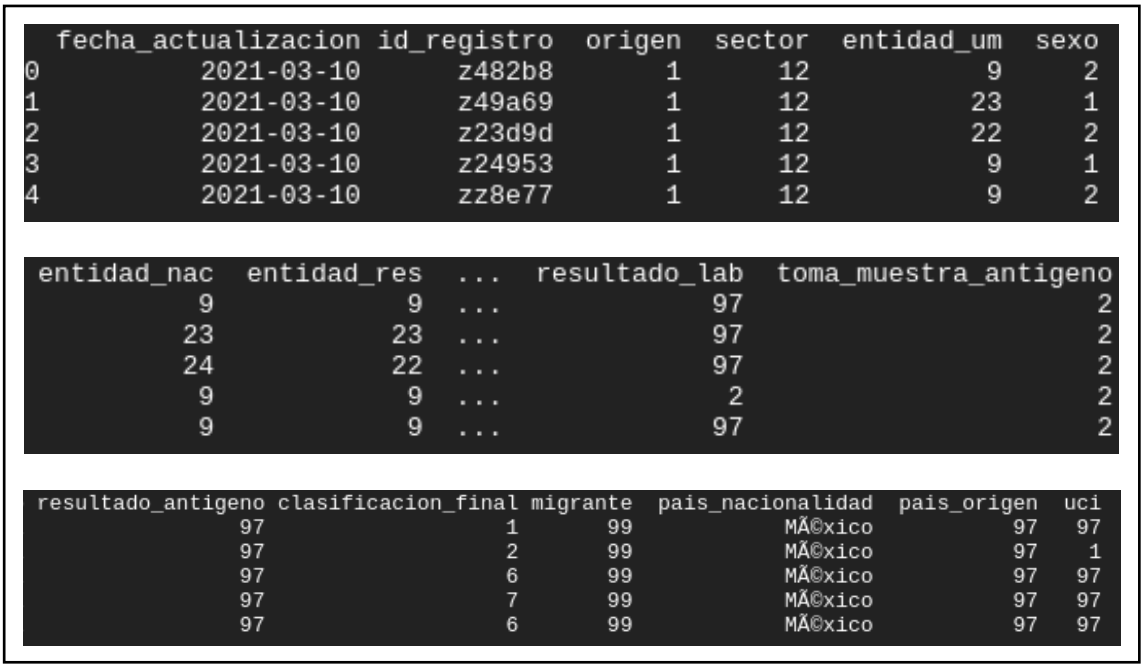


	accum_cases_report	ent_id	time_id	new_cases_report	accum_deaths_report	new_deaths_report
0	23651	1	20210310	0	2053	0
1	45634	2	20210310	0	7626	0
2	26989	3	20210310	0	1218	0
3	8657	4	20210310	0	1074	0
4	65547	5	20210310	0	5747	0

accum_hospitalized_report	new_hospitalized_report	accum_suspect_report	new_suspect_report
4820	0	10415	0
13954	0	33764	0
2392	0	2808	0
2276	0	4532	0
11087	0	14316	0

Figura 45.Formato inicial dataFrame con la unión de las 4 llamadas a la API

Los datos del último reporte de covid son descargados por el pipeline *covid_pipeline* y puestos a disposición del presente pipeline, al momento de la elaboración de esta documentación, el DataFrame inicial contenía 5.685.052 filas por 40 columnas. La siguiente imagen muestra la primeras filas y algunas columnas del DataFrame inicial



	fecha_actualizacion	id_registro	origen	sector	entidad_um	sexo
0	2021-03-10	z482b8	1	12	9	2
1	2021-03-10	z49a69	1	12	23	1
2	2021-03-10	z23d9d	1	12	22	2
3	2021-03-10	z24953	1	12	9	1
4	2021-03-10	zz8e77	1	12	9	2

entidad_nac	entidad_res	...	resultado_lab	toma_muestra_antigeno
9	9	...	97	2
23	23	...	97	2
24	22	...	97	2
9	9	...	2	2
9	9	...	97	2

resultado_antigeno	clasificacion_final	migrante	pais_nacionalidad	pais_origen	uci
97	1	99	MÃ©xico	97	97
97	2	99	MÃ©xico	97	1
97	6	99	MÃ©xico	97	97
97	7	99	MÃ©xico	97	97
97	6	99	MÃ©xico	97	97

Figura 46. Formato inicial datos COVID-19

Transformación y limpieza

Primero, se estandarizan los ID *de covid_positive* donde *1* indica que el paciente dio positivo al covid, *2* si fue negativo al covid y 3 si no se ha obtenido resultado (casos sospechosos). 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. Para cada medida (casos confirmados, fallecidos, hospitalizados y sospechosos) se tiene una columna con el valor diario, el valor acumulado, el promedio móvil a 7 días y el total cada 7 días, además de tasa cada 100 mil habitantes.

Finalmente se agrega una columna adicional con valores correlativos, partiendo en 1 el día que se contabilizaron 50 casos de COVID-19 confirmados y otra columna que inicia en 1 el día que se contabilizaron 10 fallecidos por COVID-19. Esto es solo con fines de visualización posterior.

Al momento en el que se elaboró esta documentación, el DataFrame final contenía 13.869 filas y 45 columnas. La siguiente imagen muestra algunas filas y algunas columnas de éste.

	time_id	ent_id	daily_cases	daily_hospitalized	daily_suspect	daily_deaths	...	rate_daily_deaths
24	20200103	1	0	0	2	0	...	0.0
52	20200104	1	0	0	0	0	...	0.0
80	20200105	1	0	0	0	0	...	0.0
109	20200106	1	0	0	1	0	...	0.0
141	20200107	1	0	0	1	0	...	0.0

rate_accum_deaths	rate_new_deaths_report	rate_accum_deaths_report	day_from_50_cases	day_from_10_deaths
0.0	0.0	0.0	0	0
0.0	0.0	0.0	0	0
0.0	0.0	0.0	0	0
0.0	0.0	0.0	0	0
0.0	0.0	0.0	0	0

Figura 47. Formato final tabla de datos COVID-19 estadísticas a nivel estatal