Documentación de modelado

Más Allá de las Fronteras: El Viaje de los Colombianos y el Poder de las Remesas

Facultad de Ingeniería Industrial

Asignatura:
StoryTelling

Presentado por:
Jaime Arturo Ramírez
Keisy González
Carlos Galindo Aguilera
William Pinilla

Marzo 2025

Universidad de los Andes

Tabla de Contenido

1. Descripción de Variables

Dataset – Colombianos registrados fuera del país.

Nombre original	Nuevo nombre	Tipo	Descripción
Área Conocimiento	Area_Conocimiento	Fecha/Hora	Área de conocimiento en la que se especializa la persona o entidad.
Cantidad de personas	Cantidad_Personas	Número	Número de personas involucradas en el registro.
Pais nacimiento	Ciudad_Nacimiento	Texto	Ciudad en la que nació la persona.
Ciudad de Residencia	Ciudad_Residencia	Texto	Ciudad en la que reside actualmente la persona.
Edad (años)	Edad_Anios	Número	Edad de la persona expresada en años.
Estado civil	Estado_Civil	Texto	Estado civil de la persona (Soltero, Casado, etc.).
Etnia de la persona	Etnia_Persona	Texto	Identificación étnica de la persona según categorías oficiales.
Fecha de Registro	Fecha_Registro	Fecha	Fecha en la que se realizó el registro.
Género	Género	Texto	Género de la persona (Masculino, Femenino, Otro).
Grupo edad	Grupo_Edad	Texto	Clasificación de la edad en rangos (Ej: 18-25, 26-35).
Nivel Académico	Nivel_Academico	Texto	Nivel educativo alcanzado por la persona (Ej: Primaria, Secundaria, Universitario).
Oficina de registro	Oficina_Registro	Texto	Ciudad donde trabaja la persona.
País	País	Texto	País relacionado con el registro o residencia de la persona.
Pais nacimiento	Pais_Nacimiento	Número	País en el que nació la persona.
Sub Area Conocimiento	Sub_Area_Conocimiento	Texto	Subcategoría dentro del área de conocimiento principal.

Dataset – Remesas de Colombianos Por País

Nombre original	Nuevo nombre	Tipo	Descripción
Año	Año	Fecha	Área de conocimiento en la que se especializa la persona o entidad.
Año original formato num	Año original formato num	Número	Número de personas involucradas en el registro.
Año_Trimestre	Trimestre del Año	Número	Ciudad en la que nació la persona.

Dataset – Remesas de Colombianos Por Departamento

Nombre original	Nuevo nombre	Tipo	Descripción
Año	Año	Fecha/Hora	Área de conocimiento en la que se especializa la persona o entidad.
Trimestre	Trimestre	Texto	Número de personas involucradas en el registro.
Año_Trimestre	Año_Trimestre	Texto	Ciudad en la que nació la persona.
Total_Colombia	Total_Colombia	Número	PIB Total del Pais
Departamento -> *	[Nombre del Departamento]	Número	PIB por departamento

Dataset – PIB (Se tomó el top 3 de los departamentos)

Nombre original	Nuevo nombre	Tipo	Descripción
Anio	Anio	Fecha	Año del PIV
Antioquia	Antioquia	Número	Porcentaje que representan las remesas del PIB de Antioquia
Cundinamarca	Cundinamarca	Número	Porcentaje que representan las remesas del PIB de Cundinamarca
Valle	Valle	Número	Porcentaje que representan las remesas del PIB del Valle

2. Modelado de Variables

Fueron pocas las variables que se derivaron o cambiaron, al trabajar con bases de datos de Colombia, estas no requieren renombramiento, ni traducción.

Dataset – Colombianos registrados fuera del país.

Nombre Original	Transformación	Procedimiento	Descripción
Ciudad de Residencia	Ciudad_Residencia	Derivación de una nueva columna para solo la ciudad de residencia, por script	Ciudad en la que reside actualmente la persona.
Ciudad de Residencia	Pais_Residencia	Derivación de una nueva columna para solo para País de residencia, por script	País en la que reside actualmente la persona.
Ciudad_Pais_nacimiento	Concatena Ciudad y País	Dado que para hacer el mapa de calor para	

		el origen de los colombianos, se debía poner explícitamente el país y la ciudad se concatenó de nuevo en otra columna, no servía la original porque tenía otra información y no tenía el formato correcto	
Anio	Extraer solo el año de fecha de registro	No se requería el detalle de la fecha exacta, solo elanio	Año de la fecha de registro

Dataset - Remesas de Colombianos por País.

Nombre Original	Transformación	Procedimiento	Descripción
Anio	Extraer solo el año de fecha de registro	No se requería el detalle de la fecha exacta, solo elanio	Año de la fecha de registro
(PAIS)	Dado que los países no estaba dentro de una columna en el dataset, sino una fila, se hizo una agregación (**SUM**) para obtener el total de remesas por País.	Se realizó este procedimiento desde Looker.	Total de remesas por País

A continuación se muestra la derivación de columnas en Looker.

Dataset - Remesas de Colombianos por Departamento.

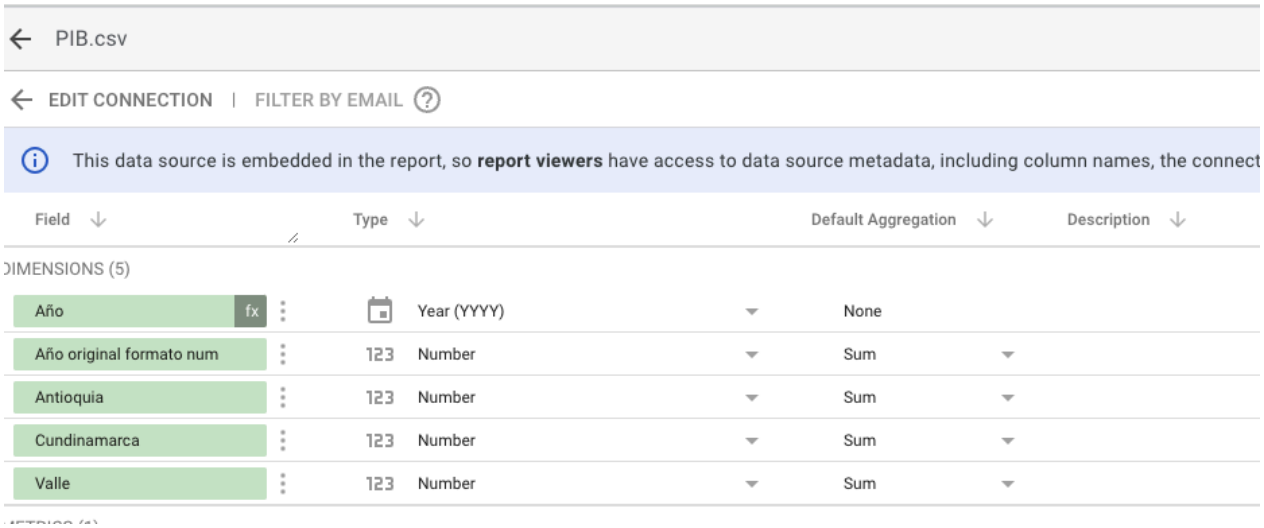
Nombre Original	Transformación	Procedimiento	Descripción
Anio	Extraer solo el año de fecha de registro	No se requería el detalle de la fecha exacta, solo elanio	Año de la fecha de registro
(PAIS)	Dado que los países no estaba dentro de una columna en el dataset, sino una fila, se hizo una agregación (**SUM**) para obtener el total de remesas por País.	Se realizó este procedimiento desde Looker.	Total de remesas por Departamento

Dataset - Remesas de colombianos por Departamento.

Field	Type	Default Aggregation	Description
DIMENSIONS (34)			
Amazonas	123 Number	Sum	
Año fx	Year (YYYY)	None	
Año original formato num	123 Number	Sum	
Año_Trimestre	ABC Text	None	
Antioquia	123 Number	Sum	
Arauca	123 Number	Sum	
Atlántico	123 Number	Sum	
Bolívar	123 Number	Sum	
Boyacá	123 Number	Sum	
Caldas	123 Number	Sum	
Caquetá	123 Number	Sum	
Casanare	123 Number	Sum	
Cauca	123 Number	Sum	
Cesar	123 Number	Sum	
Chocó	123 Number	Sum	
Córdoba	123 Number	Sum	
Cundinamarca	123 Number	Sum	
Guaviare	123 Number	Sum	
Huila	123 Number	Sum	
La Guajira	123 Number	Sum	
Magdalena	123 Number	Sum	
Meta	123 Number	Sum	
Nariño	123 Number	Sum	
Norte Santander	123 Number	Sum	
Putumayo	123 Number	Sum	
Quindío	123 Number	Sum	
Risaralda	123 Number	Sum	
San Andrés	123 Number	Sum	
Santander	123 Number	Sum	
Sucre	123 Number	Sum	

Number

Dataset - PIB



3. Fuente de datos

Datos Abiertos:

- Migración colombianos a Canadá, Estados Unidos, Sudáfrica y Australia

Dadas las limitaciones de Looker, se decidió obtener una muestra aleatoria para realizar el dashboard con 100K registros. Pero la muestra se tomó de casi 2 millones de registros.

Otras fuentes de datos usadas para el Dashboard:

Banco de la República:

- https://www.banrep.gov.co/en/node/34593

- https://suameca.banrep.gov.co/estadisticas-economicas/#/informacionSerie/4150/Remesas%20de%20trabajadores

4. Código Fuente

Notebooks:

- Collab: Migration Notebook
- Collab: Remesas Notebook

Se ha desarrollado un **notebook de Python en Google Collab** que permite un trabajo colaborativo en un entorno compartido. Esto facilita la integración de nuevas modificaciones, como:

- Creación de columnas derivadas
- Limpieza avanzada de datos e imputación.

Dado que **Looker Studio tiene limitaciones en el procesamiento de datos**, se optó por realizar el preprocesamiento mediante **scripting en Python**, utilizando Looker exclusivamente para la visualización. En casos más simples, se emplearon **filtros y derivación de columnas directamente en Looker** para completar el dashboard inicial.

Nota: Al ejecutar el notebook, se generará un nuevo CSV con las columnas derivadas, el cual será almacenado en **Google Drive** para su posterior uso en Looker.

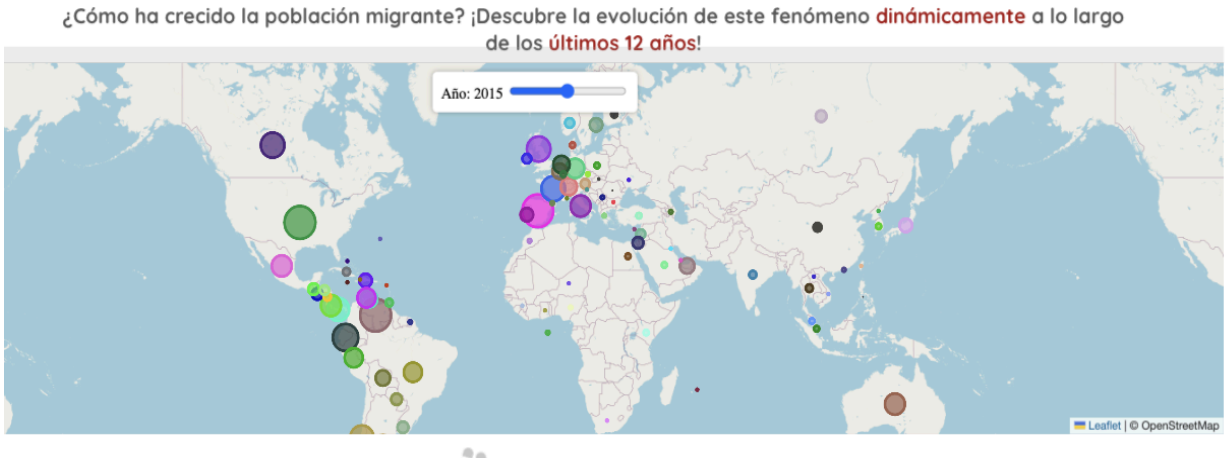
Proyecto Externo:

Mapa Dinámico - Crecimiento Poblacional

Desplegado en la siguiente ruta:

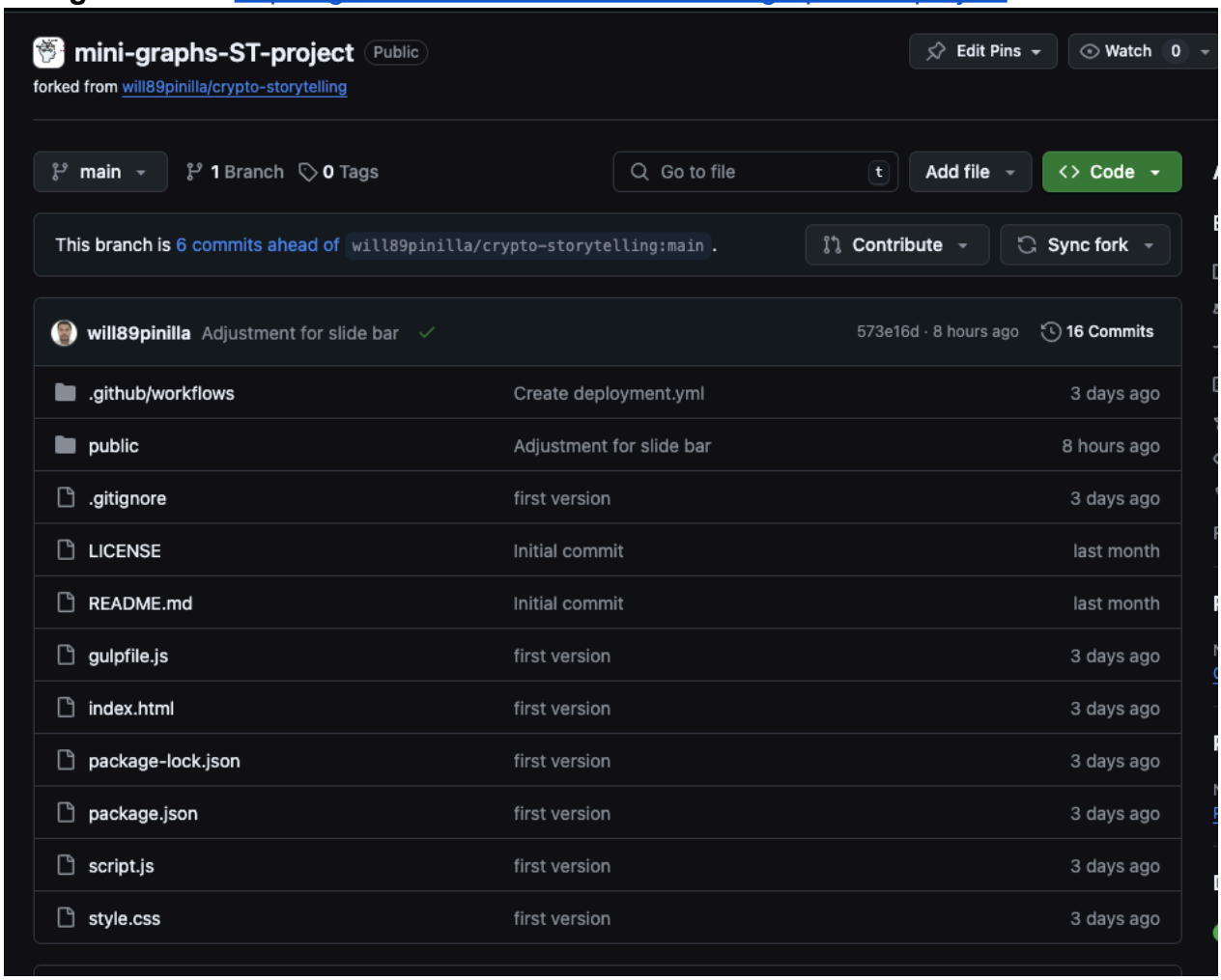
https://outlier-hunters.github.io/mini-graphs-ST-project/graphTwo.html

Este proyecto sirve para mostrar el mapa interactivo, que se embebe en dashboard de Looker. URL Embed, los datos que alimentaron la alimentacion fue el mismo CSV que se usó en el Dashboard, como se puede apreciar en el repositorio de Github.



Es un proyecto en JavaScript que se despliega con GitHub Actions.

Código Fuente: https://github.com/outlier-hunters/mini-graphs-ST-project



5. Procedimiento para realizar cambios

1. **Modificar el notebook** con las nuevas transformaciones necesarias.

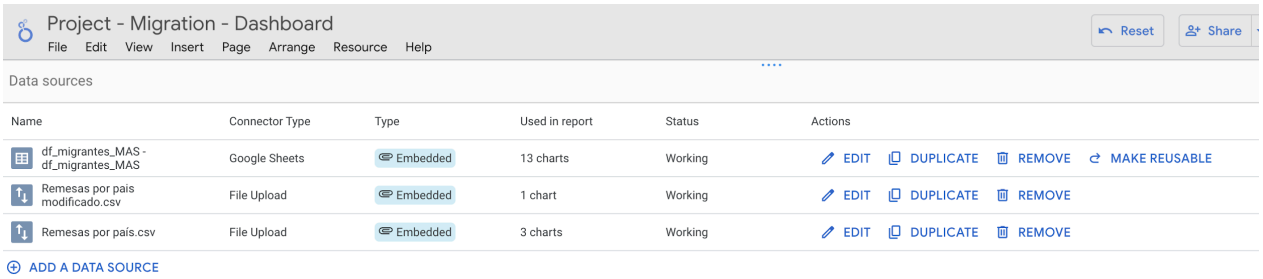
   Por ejemplo, sobre la columna work_city se desea hacer algunas transformaciones

```python
# Renombra columnas
df_registrados.rename(columns={
    "País": "Pais",
    "Código ISO país": "Codigo_ISO_pais",
    "Ciudad de Residencia": "Ciudad_Residencia",
    "Oficina de registro": "Oficina_Registro",
    "Grupo edad": "Grupo_Edad",
    "Edad (años)": "Edad_Anios",
    "Área Conocimiento": "Area_Conocimiento",
    "Sub Area Conocimiento": "Sub_Area_Conocimiento",
    "Nivel Académico": "Nivel_Academico",
    "Estado civil": "Estado_Civil",
    "Género": "Genero",
    "Etnia de la persona": "Etnia_Persona",
    "Estatura (CM)": "Estatura_CM",
    "Ciudad de Nacimiento": "Ciudad_Nacimiento",
    "Localización": "Localizacion",
    "Fecha de Registro": "Fecha_Registro",
    "Cantidad de personas": "Cantidad_Personas",
    "Ciudad Residencia": "Ciudad_Residencia",
}, inplace=True)
```

```python
df_registrados["Fecha_Registro"] = df_registrados["Fecha_Registro"].dt.year.map(lambda y: pd.Timestamp(year=y, month=1, day=1))
```

Se ejecuta todo el notebook y este genera un archivo nuevo CSV.

2. **Subir el archivo actualizado a Google Drive** y compartir el enlace.
3. **Crear una nueva fuente de datos en Looker** para incluir las nuevas columnas.



4. **Actualizar las visualizaciones** en Looker utilizando la nueva fuente de datos, adicionar nuevas columnas ya sea como una métrica o una dimensión, por ejemplo si sé crear un chart y se agrega otro campo como salary_cop ya se podría seleccionar sin problema adicionando el nuevo datasource.

Limpieza y Modelado de Datos - Migrantes

1. Introducción

En esta sección del documento se describen los procedimientos de limpieza y modelado de datos implementados en el notebook "Migrantes.ipynb". Se detallan las transformaciones aplicadas, las estrategias de preprocesamiento y los enfoques de modelado utilizados, con una explicación detallada de cada paso del código.

2. Limpieza de datos

2.1. Carga de datos

Los datos fueron obtenidos desde Google Drive utilizando la librería gdown. El archivo viene en un formato parquet que optimiza el almacenamiento de la información.

```
[ ]  # ID del archivo en Google Drive
     file_id_col_emigrants = "1l7NiPkB7Jnd5FPyWNqJhl72VhEQS1HEO"

     output_col_emigrants = "colombianEmigrants.parquet"

     def download_drive_files(file_id, output_file):
       url = f"https://drive.google.com/uc?id={file_id}"
       gdown.download(url, output_file, quiet=False)

     download_drive_files(file_id_col_emigrants, output_col_emigrants)
```

```
Downloading...
From: https://drive.google.com/uc?id=1l7NiPkB7Jnd5FPyWNqJhl72VhEQS1HEO
To: /content/colombianEmigrants.parquet
100%|██████████| 17.8M/17.8M [00:00<00:00, 66.5MB/s]
```

Se cargan los datos en un dataframe y se copian por separado para posteriormente realizar el proceso de limpieza y modelado.

```
[ ]  raw_df_registrados = pd.read_parquet(output_col_emigrants)
```

```
[ ]  df_registrados = raw_df_registrados.copy()
```

2.2. Validaciones previas

Se verificaron valores nulos, duplicados y tipos de datos:

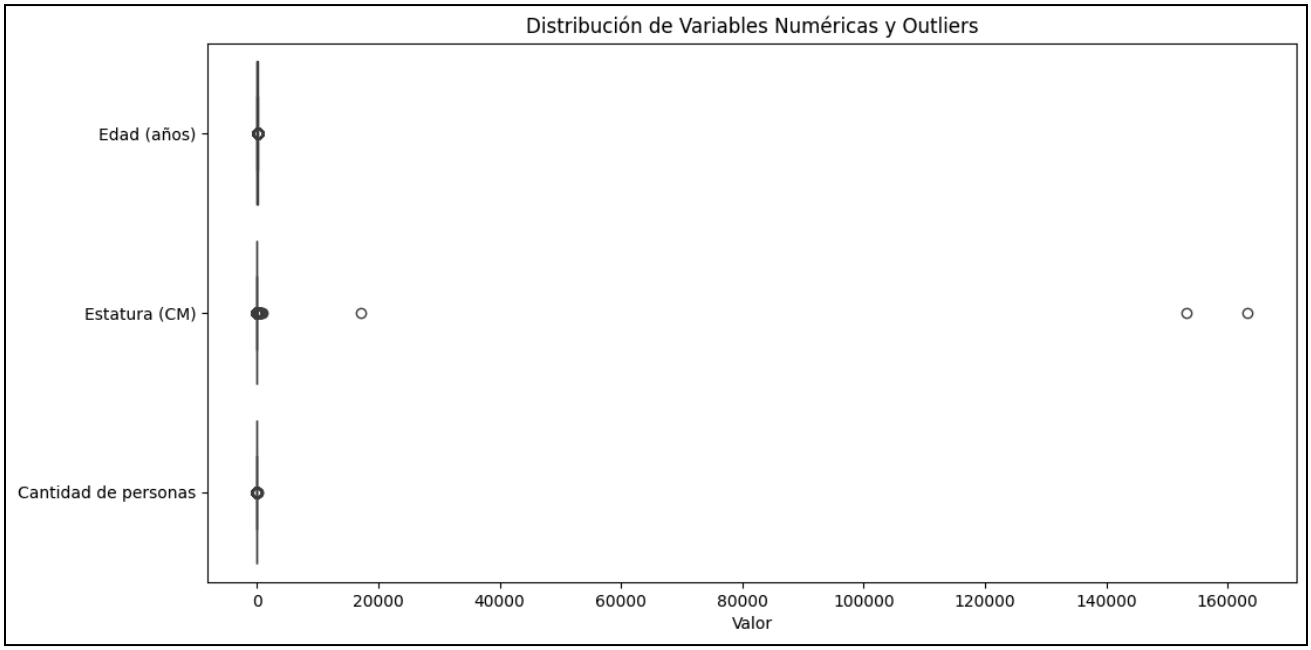
```
[ ] df_registrados.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1634319 entries, 0 to 1634318
Data columns (total 17 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   País                  1634319 non-null  object
 1   Código ISO país       1634319 non-null  object
 2   Ciudad de Residencia  1634319 non-null  object
 3   Oficina de registro   1634319 non-null  object
 4   Grupo edad            1634319 non-null  object
 5   Edad (años)           1634319 non-null  int64
 6   Área Conocimiento     1634319 non-null  object
 7   Sub Area Conocimiento 1634319 non-null  object
 8   Nivel Académico       1634319 non-null  object
 9   Estado civil          1634319 non-null  object
 10  Género                1634319 non-null  object
 11  Etnia de la persona   1634319 non-null  object
 12  Estatura (CM)         1634319 non-null  int64
 13  Ciudad de Nacimiento  1634319 non-null  object
 14  Localización          1634319 non-null  object
 15  Fecha de Registro     1634319 non-null  object
 16  Cantidad de personas  1634319 non-null  int64
dtypes: int64(3), object(14)
memory usage: 212.0+ MB

[ ] df_registrados.duplicated().sum()

0
```

Se indaga la existencia de outliers en las variables numéricas y se encuentran valores anormales en la variable Estatura (CM):



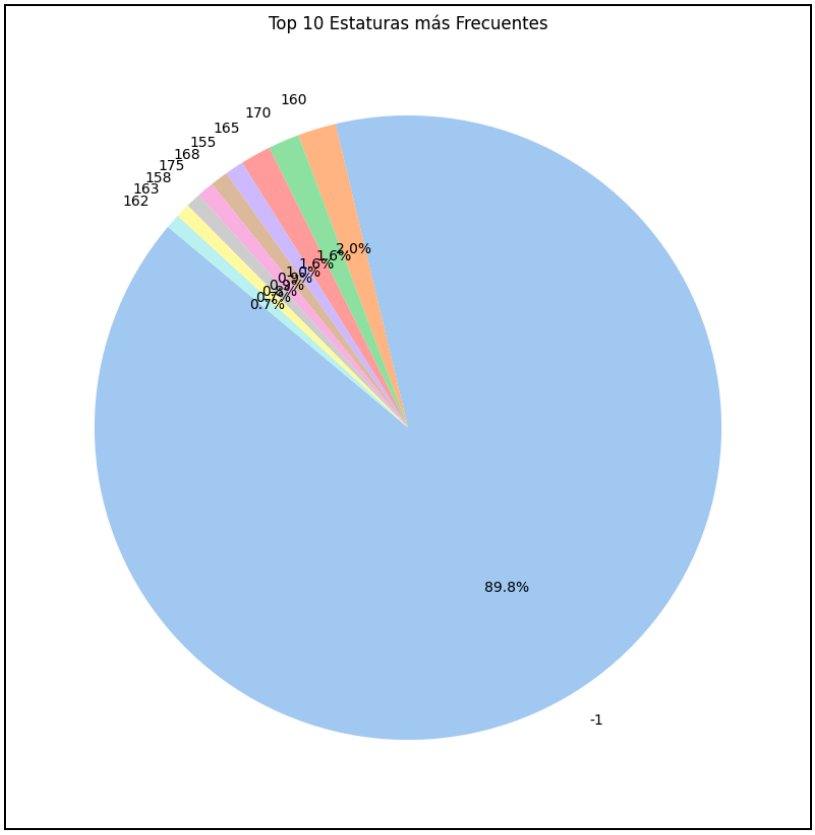
Se identifican los outliers y en total se hallan 18 registros con estaturas por encima de los 3mts:

```
[ ]  # Contar registros donde la estatura sea mayor a 300 cm
     conteo_outliers = df_registrados[df_registrados['Estatura (CM)'] > 300].shape[0]

     # Mostrar el resultado
     print(f"Número de registros con estatura mayor a 300 cm: {conteo_outliers}")
```

```
Número de registros con estatura mayor a 300 cm: 18
```

Se grafican el top 10 de las estaturas más frecuentes para verificar posibles anomalías en los datos que no fueron visibles en la gráfica de outliers y se observa que casi el 90% de los valores registrados tienen valor -1.



2.3. Limpieza de datos (Ejemplo de Limpieza o adaptación de alguna nueva columna derivada)

- Teniendo en cuenta los hallazgos previos, se reemplazan las estaturas con valor -1 y en su lugar se introduce la estatura media de los datos conocidos.

```
[ ]  # Calcular la media de las estaturas válidas (excluyendo -1)
     media_estatura = df_registrados[df_registrados['Estatura (CM)'] > 0]['Estatura (CM)'].mean()
     print(media_estatura)

     # Reemplazar los valores de -1 con la media
     df_registrados['Estatura (CM)'] = df_registrados['Estatura (CM)'].replace(-1, media_estatura)

     # Verificar que ya no hay valores -1
     print(df_registrados['Estatura (CM)'].value_counts())
```

- Adicionalmente, se filtran del dataset los registros donde la estatura sea mayor a 300cms.

```
[ ]  # Filtrar y mantener en el dataset solo estaturas "normales" < 3mts
     df_registrados = df_registrados[df_registrados['Estatura (CM)'] < 300]

     len(df_registrados)

     1634301
```

- Se realiza un análisis de conteo agrupado para las variables categóricas y de allí se desprenden las siguientes transformaciones en los datos:

  Se corrigen los caracteres en formatos no legibles de la columna “País”.

```
[ ]  #Se corrigen los nombres con caracteres en formatos no legigles
     correcciones = {
         "CHINA, REP�BLICA POPULAR": "CHINA, REPÚBLICA POPULAR",
         "CURA�AO": "CURAÇAO",
         "ESPA�A": "ESPAÑA",
         "LAO, REPUBLIC DEMOCRATICA POPULAR DE": "LAOS, REPÚBLICA DEMOCRÁTICA POPULAR DE",
         "REPUBLICA DE CHINA-TAIWAN": "TAIWÁN",
         "SUAZILANDIA": "ESWATINI",
         "VIET NAM": "VIETNAM",
         "CONGO": "REPÚBLICA DEL CONGO",
         "CONGO, REPUBLICA DEMOCRATICA DE": "REPÚBLICA DEMOCRÁTICA DEL CONGO",
         "COREA, REPUBLICA DE": "COREA DEL SUR",
         "COREA, REPUBLICA DEMOCRATICA POPULAR DE": "COREA DEL NORTE",
         "FEDERACION DE RUSIA": "RUSIA",
         "IRAN, REPUBLICA ISLAMICA DE": "IRÁN",
         "MACEDONIA, EX REPUBLICA YUGOSLAVA DE": "NORTE DE MACEDONIA",
         "REPUBLICA ARABE SIRIA": "SIRIA",
         "MOLDOVA, REPUBLICA DE": "MOLDAVIA",
         "TANZANIA, REPUBLICA UNIDA DE": "TANZANIA",
         "SANTO TOME Y PRINCIPE": "SANTO TOMÉ Y PRÍNCIPE",
         "ISLAS VIRGENES BRITANICAS": "ISLAS VÍRGENES BRITÁNICAS",
         "REPUBLICA CHECA": "CHEQUIA"
     }

     df_registrados["País"] = df_registrados["País"].replace(correcciones)
```

  Se estandarizan los nombres de los países en inglés pues resulta beneficioso al momento de aplicar métodos de geolocalización.

```
        "SUDAN":  "Sudan",
        "SUDAN DEL SUR": "South Sudan",
        "SURINAM": "Suriname",
        "UGANDA": "Uganda",
        "UZBEKISTAN": "Uzbekistan",
        "TRINIDAD Y TOBAGO": "Trinidad and Tobago",
        "TUNEZ": "Tunisia",
        "VANUATU": "Vanuatu",
        "TOGO": "Togo"
    }

    # Es mejor matener las veriones en ingles para temas de geolocalizacion
    df_registrados["País"] = df_registrados["País"].replace(country_translation)
```

Se separan el “Estado/Pais” de la “Ciudad” en la residencia del migrante.

```
[ ] # Separar Estado-Pais de Ciudad en dos columnas y llenar los vacíos con NaN
    df_registrados[['Estado/País Residencia', 'Ciudad Residencia']] = df_registrados['Ciudad de Residencia'].str.split('/', n=1, expand=True)

    df_registrados['Ciudad Residencia'] = df_registrados['Ciudad Residencia'].fillna("Desconocido")

    # Verificar los primeros valores después de la transformación
    print(df_registrados[['Estado/País Residencia', 'Ciudad Residencia']].head())
```

```
  Estado/País Residencia Ciudad Residencia
0                  KABOL             KABUL
1                 KHOWST          A'IKHEL
2                 TIRANE            FACESH
3                 BERLIN            BERLIN
4                 BERLIN            BERLIN
```

Se remueve el texto prefijo de la oficina de registro “C. “.

```
[ ] # Remover "C. " al inicio de los valores de las Oficinas de Registro y volver a Mayuscula
    df_registrados["Oficina de registro"] = df_registrados["Oficina de registro"].str.replace(r"^C\.", "", regex=True)
    df_registrados["Oficina de registro"] = df_registrados["Oficina de registro"].str.strip().str.upper()

    df_registrados["Oficina de registro"].head()
```

	Oficina de registro
0	NUEVA DELHI
1	NUEVA DELHI
2	ROMA
3	BERLIN
4	BERLIN

Se corrige la columna “Área Conocimiento” que contiene valores con errores de ortografía.

```
[ ] # Reemplazar caracteres ilegibles
    df_registrados["Área Conocimiento"] = df_registrados["Área Conocimiento"].replace({"": "Ó"}, regex=True)

    correcciones = {
        "MATEMÓTICAS": "MATEMÁTICAS",
        "ECONOMÓA": "ECONOMÍA",
        "INGENIERÓA": "INGENIERÍA",
        "AGRONOMÓA": "AGRONOMÍA"
    }

    # Replace incorrect characters
    df_registrados["Área Conocimiento"] = df_registrados["Área Conocimiento"].replace(correcciones, regex=True)

    df_registrados["Área Conocimiento"].unique()
```

Se corrige y se estandariza el “Nivel Académico”.

```
[ ]  # Corregir Nivel Academico
     correcciones = {
         'POSTGRADO - ESPECIALIZACI@N': 'ESPECIALIZACIÓN',
         'POSTGRADO - MAESTR@A': 'MAESTRÍA',
         'PREGRADO - PROFESIONAL': 'PROFESIONAL',
         'PREGRADO - TECNOL@GICO': 'TECNOLÓGICO',
         'PREGRADO - T@CNICO PROFESIONAL': 'TÉCNICO PROFESIONAL',
         'SIN PROFESI@N': 'SIN PROFESIÓN'
     }

     df_registrados["Nivel Académico"] = df_registrados["Nivel Académico"].replace(correcciones, regex=True)

     df_registrados["Nivel Académico"].unique()
```

```
array(['NO INDICA', 'BACHILLERATO', 'PRIMARIA', 'PROFESIONAL',
       'POSTGRADO - MAESTRIA', 'NINGUNO', 'POSTGRADO - DOCTORADO',
       '(NO REGISTRA)', 'TÉCNICO PROFESIONAL', 'TECNOLÓGICO',
       'ESPECIALIZACIÓN', 'SIN PROFESIÓN'], dtype=object)
```

Se estandariza el “Estado Civil”.

```
[ ]  # Diccionario de mapeo con la nueva agrupación del estado civil
     estado_civil_map = {
         "CASADO": "Casado",
         "UNION_LIBRE": "Unión Libre",
         "SEPARADO_MATRIMONIO": "Divorciado",
         "SEPARADO_UNION_LIBRE": "Divorciado",
         "DIVORCIADO": "Divorciado",
         "VIUDO": "Viudo",
         "SOLTERO": "Soltero",
         "DESCONOCIDO": "Desconocido"
     }

     # Aplicar la agrupación en el DataFrame
     df_registrados["Estado civil"] = df_registrados["Estado civil"].replace(estado_civil_map)
```

Se corrigen valores de la “Etnia de la persona”.

```
[ ]  # Corregir Etnia
     correcciones = {
         'IND@GENA': 'INDÍGENA',
         'INDIGENA': 'INDÍGENA',
     }

     df_registrados["Etnia de la persona"] = df_registrados["Etnia de la persona"].replace(correcciones, regex=True)

     df_registrados["Etnia de la persona"].unique()
```

```
array(['SIN ETNIA REGISTRADA', 'NINGUNA', 'OTRO', 'AFRODESCENDIENTE',
       'GITANO', 'RAIZAL DEL ARCHIPIELAGO DE SAN ANDRES',
       'PALENQUERO DE SAN BASILIO', 'INDÍGENA'], dtype=object)
```

Se separa el “Pais_Nacimiento” de “Ciudad_Nacimiento”.

```
[ ] # Separar 'País Nacimiento' y 'Ciudad Nacimiento' asegurando siempre tres valores
    df_registrados[['Pais_Nacimiento', '_', 'Ciudad_Nacimiento']] = df_registrados['Ciudad de Nacimiento'].str.split('/', n=2, expand=True)

    # Eliminar la columna intermedia '_' ya que no la necesitamos y la col original
    df_registrados.drop(columns=['_'], inplace=True)
    df_registrados.drop(columns=['Ciudad de Nacimiento'], inplace=True)
```

Se formatea la “Fecha de Registro” que viene como una cadena de texto para almacenarla como tipo de dato datetime (Period), pues no se cuenta con el día sino la relación año-mes.

```
[ ] # Transformar Fecha de Registro de categórico a tipo fecha o periodo para  análisis mensuales
    df_registrados['Fecha de Registro'] = pd.to_datetime(df_registrados['Fecha de Registro'], format='%Y-%m').dt.to_period('M')
```

Posteriormente, se agrega por defecto el día “01” para establecer el formato “YYYY-mm-dd”.

```
[ ] df_registrados["Fecha de Registro"] = df_registrados["Fecha de Registro"].astype(str) + "-01"
    df_registrados["Fecha de Registro"] = pd.to_datetime(df_registrados["Fecha de Registro"], format='%Y-%m-%d', errors='coerce')
```

Se renombran las columnas siguiendo convenciones tipo SQL para facilitar el manejo y visualización en Looker.

```
[ ] # Renombra columnas
    df_registrados.rename(columns={
        "País": "Pais",
        "Código ISO país": "Codigo_ISO_pais",
        "Ciudad de Residencia": "Ciudad_Residencia",
        "Oficina de registro": "Oficina_Registro",
        "Grupo edad": "Grupo_Edad",
        "Edad (años)": "Edad_Anios",
        "Área Conocimiento": "Area_Conocimiento",
        "Sub Area Conocimiento": "Sub_Area_Conocimiento",
        "Nivel Académico": "Nivel_Academico",
        "Estado civil": "Estado_Civil",
        "Género": "Genero",
        "Etnia de la persona": "Etnia_Persona",
        "Estatura (CM)": "Estatura_CM",
        "Ciudad de Nacimiento": "Ciudad_Nacimiento",
        "Localización": "Localizacion",
        "Fecha de Registro": "Fecha_Registro",
        "Cantidad de personas": "Cantidad_Personas",
        "Ciudad Residencia": "Ciudad_Residencia",
    }, inplace=True)
```

Por último, se realizan invitaciones sobre la “Ciudad_Nacimiento” para aquellos migrantes cuyo valor es nulo “NaN”.

1. Se codifica "Ciudad_Nacimiento" en números usando LabelEncoder().
2. Se seleccionan las columnas numéricas y se eliminan las categóricas.
3. Se usa un modelo de Random Forest para predecir los valores faltantes.
4. Se reemplazan los valores nulos con las predicciones.
5. Se recuperan los nombres originales de las ciudades.
6. Se verifica que no haya valores nulos en "Ciudad_Nacimiento".
7. Se elimina la columna codificada.

Este método permite imputar datos de ciudades faltantes de manera inteligente usando RandomForestClassifier, lo cual es más preciso que usar la media o la moda.

```
[ ]  le = LabelEncoder()
     df_registrados['Ciudad_Nacimiento_encoded'] = le.fit_transform(df_registrados['Ciudad_Nacimiento'].astype(str))

     # Seleccionar variables predictoras (excluyendo la columna original con strings)
     # Excluir columnas categóricas del proceso de imputación
     # The line below was changed to drop the column from the original dataframe first
     variables_predictoras = df_registrados.drop(columns=['Ciudad_Nacimiento']).select_dtypes(include=np.number) # Exclude string columns

     # Configurar el imputador con RandomForestClassifier
     imputer = IterativeImputer(estimator=RandomForestClassifier(), max_iter=10, random_state=42)

     # Aplicar la imputación a los datos
     df_imputed_numeric = imputer.fit_transform(variables_predictoras)  # Impute only on numeric columns

     # Convertir de nuevo a DataFrame
     df_imputed_numeric = pd.DataFrame(df_imputed_numeric, columns=variables_predictoras.columns, index=variables_predictoras.index)  # keep original index

     # Unir las columnas imputadas con las columnas originales
     df_imputed = df_registrados.drop(columns=variables_predictoras.columns).join(df_imputed_numeric)

     # Mapear los valores imputados de vuelta a las ciudades originales
     df_imputed['Ciudad_Nacimiento'] = le.inverse_transform(df_imputed['Ciudad_Nacimiento_encoded'].astype(int))

     # Verificar valores nulos después de la imputación
     print("Valores nulos después de la imputación:", df_imputed['Ciudad_Nacimiento'].isna().sum())

     # Eliminar la columna codificada si no se necesita
     df_imputed.drop(columns=['Ciudad_Nacimiento_encoded'], inplace=True)
```

Una vez completada la limpieza se exporta el dataset que servirá de insumo para el diseño e implementación del dashboard. Para esto se utiliza un método aleatorio simple en la selección de los registros y se toman 100.000 de la base total (1.6MM).

```
Metodo Aleatorio Simple

# Seleccionar una muestra aleatoria de 100000 registros (Metodo Aleatorio Simple)
df_migrantes_MAS = df_final.sample(n=100000, random_state=42)
```

Limpieza y Modelado de Datos - Remesas

1. Introducción

En esta sección del documento se describen los procedimientos de limpieza y modelado de datos implementados en el notebook **"MigrantesRemesas.ipynb"**. Se detallan las transformaciones aplicadas, las estrategias de preprocesamiento y los enfoques de modelado utilizados, con una explicación detallada de cada paso del código.

2. Limpieza de datos

2.1. Carga de datos

Este código intenta descargar un archivo Excel desde Google Sheets utilizando su identificador único. Primero, construye la URL de descarga y usa requests.get() para obtener el archivo en formato xlsx, verificando que la solicitud sea exitosa. Luego, guarda el contenido descargado en un archivo temporal y lo lee con Pandas, extrayendo las hojas Ingresos Remesas por País e Ingresos Remesas por Depto. Una vez cargados los datos en DataFrames, muestra las primeras filas para verificación. Si ocurre un error en la descarga o lectura, se intenta una carga manual del archivo en Google Collab usando files.upload(). Si el usuario sube el archivo, se extrae su nombre, se leen las mismas hojas del Excel y se informa que los datos se han cargado correctamente; en caso contrario, se notifica que no se subió ningún archivo.

```python
# Descargar y leer el archivo especificando el motor
try:
    # Descargar el archivo
    r = requests.get(f"https://docs.google.com/spreadsheets/d/{archivo_id}/export?format=xlsx")
    r.raise_for_status()  # Verificar si la respuesta fue exitosa

    # Guardar a un archivo temporal para facilitar la lectura
    with open('temp_excel.xlsx', 'wb') as f:
        f.write(r.content)

    # Leer ambas hojas del Excel usando openpyxl como motor
    df_pais = pd.read_excel('temp_excel.xlsx', sheet_name='Ingresos Remesas por País', engine='openpyxl')
    df_depto = pd.read_excel('temp_excel.xlsx', sheet_name='Ingresos Remesas por Depto', engine='openpyxl')

    # Verificar primeras filas
    print("Datos de Ingresos Remesas por País:")
    display(df_pais.head())
    print("\nDatos de Ingresos Remesas por Depto:")
    display(df_depto.head())

except Exception as e:
    print(f"Error al leer el archivo: {e}")
    print("\nIntentando método alternativo...")

    # Método alternativo: subir manualmente el archivo
    from google.colab import files
    print("\nPor favor, sube el archivo Excel manualmente:")

    # Esperar a que el usuario suba el archivo
    uploaded = files.upload()

    if uploaded:
        # Obtener el nombre del archivo subido
        filename = list(uploaded.keys())[0]
        print(f"Archivo subido: {filename}")

        # Leer el archivo
        df_pais = pd.read_excel(filename, sheet_name='Ingresos Remesas por País', engine='openpyxl')
        df_depto = pd.read_excel(filename, sheet_name='Ingresos Remesas por Depto', engine='openpyxl')
```

2.2. Validaciones previas

Este código proporciona una visión general de los DataFrames df_pais y df_depto después de haber sido cargados desde el archivo Excel. Primero, muestra las dimensiones de cada DataFrame usando .shape, lo que indica el número de filas y columnas. Luego, examina los tipos de datos de las primeras 10 columnas con .dtypes.head(10), permitiendo identificar el tipo de cada variable (numérica, categórica, etc.). Finalmente, verifica la presencia de valores nulos en cada DataFrame utilizando .isnull().sum().sum(), lo que devuelve el número total de valores faltantes en cada conjunto de datos.

```python
# Verificar valores nulos
print("\nValores nulos en df_pais:")
print(df_pais.isnull().sum().sum())
print("\nValores nulos en df_depto:")
print(df_depto.isnull().sum().sum())
```

```
Dimensiones de df_pais: (59, 35)
Dimensiones de df_depto: (59, 33)

Tipos de datos en df_pais:
Fecha        object
Alemania     object
Argentina    object
Aruba        object
Australia    object
Bolivia      object
Brasil       object
Bélgica      object
Canadá       object
Chile        object
dtype: object

Tipos de datos en df_depto:
Fecha        object
Amazonas     object
Antioquia    object
Arauca       object
Atlántico    object
Bolívar      object
Boyacá       object
Caldas       object
Caquetá      object
Casanare     object
dtype: object

Valores nulos en df_pais:
0

Valores nulos en df_depto:
0
```

2.3. Limpieza de datos

Este código transforma la columna de fechas en los DataFrames df_pais y df_depto para extraer el año y clasificar los meses en trimestres (Q1, Q2, Q3, Q4). Luego, reorganiza los datos en un formato largo utilizando pd.melt(), lo que facilita el análisis comparativo entre países y departamentos. Esto permite que cada fila represente un país o departamento con su respectiva cantidad de remesas en dólares y el periodo de tiempo correspondiente.

```python
# Función para transformar fecha a trimestre y año
def convertir_fecha(fecha):
    # Extraer año y mes
    año = fecha.year
    mes = fecha.month

    # Determinar trimestre
    if mes in [1, 2, 3]:
        trimestre = 1
    elif mes in [4, 5, 6]:
        trimestre = 2
    elif mes in [7, 8, 9]:
        trimestre = 3
    else:
        trimestre = 4

    return año, f"Q{trimestre}"

# Aplicar la transformación a ambos dataframes
df_pais['Fecha'] = pd.to_datetime(df_pais['Fecha'])
df_pais['Año'], df_pais['Trimestre'] = zip(*df_pais['Fecha'].apply(convertir_fecha))

df_depto['Fecha'] = pd.to_datetime(df_depto['Fecha'])
df_depto['Año'], df_depto['Trimestre'] = zip(*df_depto['Fecha'].apply(convertir_fecha))

# Reorganizar los datos de países para facilitar el análisis (formato largo)
df_pais_largo = pd.melt(
    df_pais,
    id_vars=['Fecha', 'Año', 'Trimestre'],
    value_vars=[col for col in df_pais.columns if col not in ['Fecha', 'Año', 'Trimestre']],
    var_name='País',
    value_name='Remesas_USD_Millones'
)

# Reorganizar los datos de departamentos para facilitar el análisis (formato largo)
df_depto_largo = pd.melt(
    df_depto,
    id_vars=['Fecha', 'Año', 'Trimestre'],
    value_vars=[col for col in df_depto.columns if col not in ['Fecha', 'Año', 'Trimestre']],
    var_name='Departamento',
    value_name='Remesas_USD_Millones'
)
```