Universidad Tecnológica de Chihuahua
Tecnologías de la Información

Reporte de solución de caso de estudio de técnicas de limpieza de datos

Alumno:

Jatzel Israel Cruz Castruita

Grupo:

IDGS91N

Materia:

Extracción de Conocimiento en Bases de Datos

Docente:

Enrique Mascote

Índice

Introducción

El análisis de datos de migración internacional se ha convertido en una herramienta clave para comprender los patrones de movilidad de personas entre países y regiones. Los flujos migratorios reflejan fenómenos sociales, económicos y políticos, por lo que su estudio permite identificar tendencias, anticipar necesidades y apoyar la toma de decisiones a nivel gubernamental y empresarial.

Este trabajo se centra en el conjunto de datos international-migration-March-2021-citizenship-by-visa-by-country-of-last-permanent-residence, que contiene información detallada sobre migrantes clasificados por nacionalidad, tipo de visa, país de residencia anterior, tipo de viajero, sentido del viaje y periodos de tiempo. Los datos provienen de registros oficiales y estimaciones, lo que permite analizar tanto cifras confirmadas como tendencias proyectadas.

El objetivo principal de este estudio es comprender y aplicar técnicas de limpieza de datos, así como diseñar un modelo de data warehouse normalizado, que facilite la organización y análisis eficiente de la información. Para ello, se busca:

- Detectar y tratar valores faltantes, inconsistencias de formato y registros duplicados.
- Definir las tablas de hechos y dimensiones que permitan un análisis multidimensional de la información.
- Diseñar un modelo relacional normalizado hasta la Tercera Forma Normal (3FN), asegurando la integridad de los datos y la optimización del almacenamiento.

Limpieza de datos

El proceso de limpieza de datos se realizó con el objetivo de garantizar la calidad, coherencia y fiabilidad del conjunto de datos obtenido en el script.

Para ello, se aplicaron las siguientes etapas utilizando sentencias SQL para la identificación, análisis y corrección de los datos.

Revisión inicial y detección de problemas

En primer lugar, se revisó la estructura del conjunto de datos con el fin de conocer la cantidad de filas, columnas y los tipos de datos de cada campo. Posteriormente, se verificaron los posibles errores mediante consultas SQL, aplicando las siguientes comprobaciones:

- Conteo de valores nulos con SELECT COUNT(*) FROM tabla WHERE columna IS NULL;
- Detección de duplicados con SELECT columna, COUNT(*) FROM tabla GROUP BY columna HAVING COUNT(*) > 1;
- Revisión de los tipos de datos para confirmar que los campos numéricos y de texto tuvieran el formato correcto.

Tras el análisis se observó lo siguiente:

- No existían valores faltantes en ninguna columna, es decir, 0 % de valores nulos.
- No se encontraron registros duplicados.
- Las columnas numéricas (estimate, standard_error) ya se encontraban en formato numérico correcto.
- Las columnas categóricas (citizenship, visa, country_of_residence, entre otras) estaban almacenadas correctamente como texto.

Normalización de formatos

Aunque no se detectaron inconsistencias graves, se aplicaron acciones preventivas para garantizar la uniformidad de los datos:

- Se eliminaron espacios en blanco al inicio y final de los textos utilizando TRIM().
- Se reemplazaron valores vacíos o irregulares como NA, N/A, NULL, ' ' por valores nulos estándar con UPDATE tabla SET columna = NULL WHERE columna IN ('NA', 'N/A', ' ', 'NULL');
- Se verificó la integridad de los datos numéricos, asegurando que no existieran símbolos o caracteres no válidos.

Tratamiento de duplicados

Se comprobó la existencia de registros repetidos con una consulta de agrupación, sin encontrarse coincidencias.
Por tanto, no fue necesario aplicar eliminación de duplicados.

Validación del resultado

Finalmente, los datos limpios fueron almacenados nuevamente dentro del script, manteniendo la estructura original del conjunto, pero con formatos uniformes, consistentes y listos para ser utilizados en el proceso de análisis posterior.

Determinación de Hechos y Dimensiones

Para construir un data warehouse basado en el conjunto de datos de migración internacional, se definieron las tablas de dimensiones y la tabla de hechos de la siguiente manera:

Tablas de Dimensiones

- d_fecha
  Contiene información temporal desglosada por año, mes, nombre del mes y trimestre. Permite realizar análisis de migración a lo largo del tiempo y comparaciones por periodos.
- d_tipo_viajero
  Registra los diferentes tipos de viajeros (por ejemplo, turistas, trabajadores, estudiantes). Facilita el análisis según el perfil de migrante.
- d_sentido_viaje
  Indica el sentido del viaje (entrada o salida de un país). Esto permite diferenciar entre inmigración y emigración en los análisis.
- d_nacionalidad
  Contiene los países de nacionalidad de los migrantes junto con su código ISO. Permite segmentar y agrupar los datos por nacionalidad.
- d_tipo_visa
  Registra los diferentes tipos de visa utilizados para el viaje. Esto facilita el análisis de tendencias según el tipo de autorización migratoria.
- d_pais_origen
  Contiene los países de origen de los migrantes, incluyendo códigos ISO y zona geográfica. Permite analizar patrones de migración por región.
- d_estado_registro
  Indica el estado del registro de migración (por ejemplo, confirmado, estimado o con error). Esto ayuda a controlar la calidad de los datos en los análisis.

Tabla de Hechos

h_migracion

Es la tabla central del data warehouse y contiene los indicadores de migración, como el número estimado de migrantes y el error estándar.
Se conecta con todas las dimensiones mediante llaves foráneas (id_fecha, id_tipo_viajero, id_sentido, id_nacionalidad, id_visa, id_pais, id_estado).
Permite realizar análisis cuantitativos de migración internacional, cruzando información de las dimensiones para generar reportes por país, nacionalidad, tipo de visa, periodo de tiempo y sentido del viaje.

Normalización y Almacenamiento

Para optimizar el almacenamiento y garantizar la integridad de los datos en el data warehouse de migración internacional, se diseñó un modelo relacional normalizado hasta la Tercera Forma Normal (3FN).

Proceso de normalización

- Primera Forma Normal (1FN)
  Se eliminaron valores repetidos dentro de columnas y se separaron los datos atómicos en campos individuales.
  Por ejemplo, la información de país, nacionalidad y tipo de visa se colocó en columnas separadas en lugar de combinarse en un solo campo.

- Segunda Forma Normal (2FN)
  Se eliminaron dependencias parciales. Cada tabla de dimensión tiene una clave primaria única y todos los atributos dependen completamente de esta clave.
  Por ejemplo, los detalles de nacionalidad dependen solo de id_nacionalidad, no de otras columnas de la tabla de hechos.

- Tercera Forma Normal (3FN)
  Se eliminaron dependencias transitivas. Cada tabla almacena solo información directamente relacionada con su clave primaria, evitando redundancias.
  Por ejemplo, la tabla d_pais_origen solo almacena nombre, código ISO y zona geográfica; cualquier dato relacionado con migración se ubica en la tabla de hechos h_migracion.

Modelo Relacional

El modelo relacional final incluye:

- Tablas de dimensiones:
  d_fecha, d_tipo_viajero, d_sentido_viaje, d_nacionalidad, d_tipo_visa, d_pais_origen, d_estado_registro.
- Tabla de hechos:
  h_migracion que conecta todas las dimensiones mediante llaves foráneas y almacena los indicadores de migración (estimado, error_estandar).

Script

```sql
-- Tablas de dimensiones
CREATE TABLE d_fecha (
    id_fecha DATE PRIMARY KEY,
    anio INTEGER NOT NULL,
    mes INTEGER NOT NULL,
    nombre_mes VARCHAR(20),
    trimestre INTEGER
);

CREATE TABLE d_tipo_viajero (
    id_tipo_viajero SERIAL PRIMARY KEY,
    tipo_viajero VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE d_sentido_viaje (
    id_sentido SERIAL PRIMARY KEY,
    sentido VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE d_nacionalidad (
    id_nacionalidad SERIAL PRIMARY KEY,
    nacionalidad VARCHAR(150) UNIQUE NOT NULL,
    codigo_iso VARCHAR(10)
);

CREATE TABLE d_tipo_visa (
    id_visa SERIAL PRIMARY KEY,
    tipo_visa VARCHAR(150) UNIQUE NOT NULL
);

CREATE TABLE d_pais_origen (
    id_pais SERIAL PRIMARY KEY,
    nombre_pais VARCHAR(150) UNIQUE NOT NULL,
    codigo_iso VARCHAR(10),
    zona_geografica VARCHAR(100)
);

CREATE TABLE d_estado_registro (
    id_estado SERIAL PRIMARY KEY,
    estado_registro VARCHAR(50) UNIQUE NOT NULL
);
```

```sql
-- Tabla de hechos
CREATE TABLE h_migracion (
    id_migracion SERIAL PRIMARY KEY,
    id_fecha DATE NOT NULL,
    id_tipo_viajero INTEGER NOT NULL,
    id_sentido INTEGER NOT NULL,
    id_nacionalidad INTEGER NOT NULL,
    id_visa INTEGER NOT NULL,
    id_pais INTEGER NOT NULL,
    id_estado INTEGER NOT NULL,
    estimado BIGINT NOT NULL,
    error_estandar BIGINT,
    CONSTRAINT fk_fecha FOREIGN KEY (id_fecha) REFERENCES d_fecha(id_fecha),
    CONSTRAINT fk_viajero FOREIGN KEY (id_tipo_viajero) REFERENCES d_tipo_viajero(id_tipo_viajero),
    CONSTRAINT fk_sentido FOREIGN KEY (id_sentido) REFERENCES d_sentido_viaje(id_sentido),
    CONSTRAINT fk_nacional FOREIGN KEY (id_nacionalidad) REFERENCES d_nacionalidad(id_nacionalidad),
    CONSTRAINT fk_visa FOREIGN KEY (id_visa) REFERENCES d_tipo_visa(id_visa),
    CONSTRAINT fk_pais FOREIGN KEY (id_pais) REFERENCES d_pais_origen(id_pais),
    CONSTRAINT fk_estado FOREIGN KEY (id_estado) REFERENCES d_estado_registro(id_estado)
);
```

Conclusiones

Al trabajar en este proyecto, pude darme cuenta de lo importante que es limpiar y organizar los datos antes de analizarlos. Aprendí que incluso conjuntos de datos muy completos pueden tener valores faltantes, errores de formato o registros duplicados, y que tratar estos problemas correctamente es clave para obtener resultados confiables. Usar SQL para identificar y corregir estos detalles me permitió entender mejor cómo se estructuran los datos y cómo se relacionan entre sí.

También entendí la importancia de definir claramente las dimensiones y la tabla de hechos. Al diseñar las tablas de forma ordenada y normalizada, pude ver cómo cada pieza de información tiene su lugar y cómo todas juntas permiten hacer análisis más precisos y completos. Me di cuenta de que un buen diseño del modelo de datos facilita la interpretación de la información y evita redundancias innecesarias.

Como recomendación personal, creo que siempre vale la pena dedicar tiempo a la limpieza y normalización antes de comenzar cualquier análisis. Además, familiarizarse con la creación de tablas de dimensiones y hechos ayuda mucho a entender cómo se pueden cruzar los datos para obtener respuestas concretas a preguntas específicas. Por último, pienso que este tipo de ejercicios no solo sirve para practicar SQL o modelado de datos, sino también para aprender a pensar de manera estructurada sobre la información, algo que definitivamente aplicaría en futuros proyectos o trabajos relacionados con datos.

Bibliografía

Darnley, N. (2023, 17 de marzo). Limpieza de datos en SQL. LearnSQL.es. https://learnsql.es/blog/limpieza-de-datos-en-sql/

Kutz, J. (2025, 5 de septiembre). Técnicas de limpieza de datos en SQL: Claves y ejemplos. Airbyte. https://airbyte.com/data-engineering-resources/sql-data-cleaning

IBM. (s.f.). Hechos y dimensiones en las vistas SQL. IBM InfoSphere Information Server 11.5.0. Recuperado el 12 de octubre de 2025, de https://www.ibm.com/docs/es/iis/11.5.0?topic=reports-facts-dimensions-in-sql-views

Goilkar, V. (2020, 19 de mayo). Fact and dimensional table in SQL Server. Medium. https://vaishaligoilkar.medium.com/fact-and-dimensional-table-in-sql-server-44bc55d07755

Goilkar, V. (2020, 19 de mayo). Fact and dimensional table in SQL Server. Medium. https://vaishaligoilkar.medium.com/fact-and-dimensional-table-in-sql-server-44bc55d07755