**Solución Prueba\_Tecnica\_Data\_Quality\_ETL\_Tester.**

***🔹 Parte 1: Validaciones SQL***

Lo primero que hice fue crear las tablas en SQL para poder hacerles las validaciones en tiempo real en un compilador online.

| CREATE TABLE customers\_raw (  customer\_id INT PRIMARY KEY,  name VARCHAR(100),  email VARCHAR(100),  country VARCHAR(50) );  CREATE TABLE transactions\_raw (  transaction\_id INT PRIMARY KEY,  customer\_id INT,  amount DECIMAL(10, 2),  date DATE ); |
| --- |

Luego inserte los datos proporcionados en la prueba

| -- Datos para 'customers\_raw' INSERT INTO customers\_raw (customer\_id, name, email, country) VALUES (1, 'Ana Torres', 'ana@email.com', 'Colombia'), (2, 'Juan Pérez', 'NULL', 'Mexico'), -- Email es 'NULL' (string) segun PDF, lo insertamos como NULL real(3, 'Laura Gómez', 'laura\_gomez@email.com', NULL), -- Country es NULL (4, 'Juan Pérez', 'juanperez@email.com', 'Mexico'), (5, NULL, 'andres@email.com', 'Chile'); -- Name es NULL   -- Datos para 'transactions\_raw' INSERT INTO transactions\_raw (transaction\_id, customer\_id, amount, date) VALUES (100, 1, 200.0, '2025-01-01'), (101, 2, 150.0, '2025-01-02'), (102, 2, 150.0, '2025-01-02'), (103, 3, NULL, '2025-01-03'), -- Amount es NULL (104, 6, 300.0, '2025-01-04'); -- customer\_id 6 no existe en clientes |
| --- |

ya con esta tablas con sus datos creados puedo empezar a ejecutar los puntos

1. Completitud: Escribe una consulta que detecte registros con valores NULL en campos obligatorios (`name`, `email`, `country` en `customers\_raw`; `amount` en `transactions\_raw`).

| SELECT   'customers\_raw' AS tabla\_afectada,  customer\_id AS id\_registro,  'name' AS campo\_con\_error FROM   customers\_raw WHERE   name IS NULL  UNION ALL  SELECT   'customers\_raw' AS tabla\_afectada,  customer\_id AS id\_registro,  'email' AS campo\_con\_error FROM   customers\_raw WHERE   email IS NULL  UNION ALL  SELECT   'customers\_raw' AS tabla\_afectada,  customer\_id AS id\_registro,  'country' AS campo\_con\_error FROM   customers\_raw WHERE   country IS NULL  UNION ALL  SELECT   'transactions\_raw' AS tabla\_afectada,  transaction\_id AS id\_registro,  'amount' AS campo\_con\_error FROM   transactions\_raw WHERE   amount IS NULL; |
| --- |

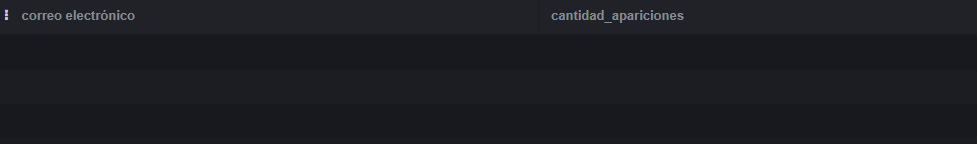
el resultado de la ejecución de esta consulta es la siguiente :



2. Unicidad: Escribe una consulta que identifique duplicados de clientes (por `email`) en `customers\_raw`.

| SELECT  email,  COUNT(1) AS cantidad\_apariciones FROM  customers\_raw WHERE  email IS NOT NULL -- Excluimos los NULL de la lógica de duplicados GROUP BY  email HAVING  COUNT(1) > 1; |
| --- |

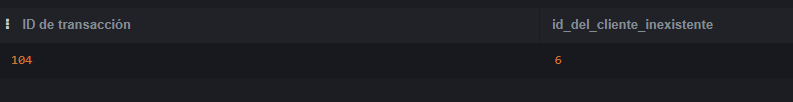
el resultado de la ejecución de la consulta es la siguiente:



3. Integridad referencial: Detecta transacciones con `customer\_id` inexistente en `customers\_raw`.

| SELECT  t.transaction\_id,  t.customer\_id AS customer\_id\_inexistente FROM  transactions\_raw t -- 't' es un alias para la tabla de transacciones LEFT JOIN  customers\_raw c ON t.customer\_id = c.customer\_id -- 'c' es un alias para clientes WHERE  c.customer\_id IS NULL; -- Filtra solo las filas donde no hubo match |
| --- |

el resultado de esta ejecución esel siguiente:



4. Reconciliación: Escribe una consulta que compare totales de `amount` por país entre tablas `transactions\_raw` y `transactions\_clean` (post-transformación).

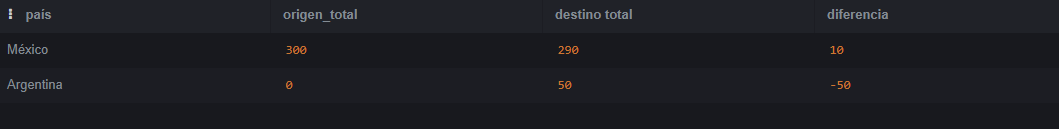
En la prueba no está la tabla ***transactions\_clean*** entonces la cree e inserté datos con errores para hacer la reconciliación.

| CREATE TABLE transactions\_clean (  transaction\_id INT PRIMARY KEY,  amount DECIMAL(10, 2),  country VARCHAR(50) );  INSERT INTO transactions\_clean (transaction\_id, amount, country) VALUES (100, 200.0, 'Colombia'), -- vamos a ponerlo como Caso OK (101, 150.0, 'Mexico'), -- vamos a ponerlo como Caso OK (102, 140.0, 'Mexico'), -- << ERROR 1: Monto incorrecto (debería ser 150.0) (105, 50.0, 'Argentina'); -- << ERROR 2: País "fantasma" que no está en el origen |
| --- |

ya luego creo la consulta para la reconciliación en donde me muestra las discrepancias.

| WITH Totales\_Origen AS (  SELECT  c.country,  SUM(t.amount) AS total\_amount\_raw  FROM  transactions\_raw t  INNER JOIN  customers\_raw c ON t.customer\_id = c.customer\_id  WHERE  t.amount IS NOT NULL  AND c.country IS NOT NULL  GROUP BY  c.country ), Totales\_Destino AS (  SELECT  country,  SUM(amount) AS total\_amount\_clean  FROM  transactions\_clean  GROUP BY  country ) SELECT  COALESCE(o.country, d.country) AS country,  COALESCE(o.total\_amount\_raw, 0) AS total\_origen,  COALESCE(d.total\_amount\_clean, 0) AS total\_destino,  (COALESCE(o.total\_amount\_raw, 0) - COALESCE(d.total\_amount\_clean, 0)) AS diferencia FROM  Totales\_Origen o FULL OUTER JOIN  Totales\_Destino d ON o.country = d.country WHERE  COALESCE(o.total\_amount\_raw, 0) != COALESCE(d.total\_amount\_clean, 0); |
| --- |

el resultado de esta consulta es el siguiente:



en donde la consulta detectó dos errores que metimos, los cuáles son que méxico tiene una diferencia de 10 transacciones, o sea que perdió ese valor y que argentina muestra -50 transacciones de más que no se justifican en el origen.

***En el archivo Validacion\_ETL\_Data.ipynb está la ejecución con la libreria pandas de Python.***

***🔹 Parte 2: Automatización de Pruebas de Calidad***

1. Implementa al menos 3 tests automatizados de calidad de datos, ya sea: - Con Python (pandas o pytest) - O con Great Expectations, dbt tests, o el framework que prefieras.

2. Los tests deben validar automáticamente las reglas definidas en la parte 1.

3. Cada test debe fallar si la regla no se cumple y generar un reporte o salida legible

aqui lo desarrollé en el archivo ***test\_data\_quality\_suite.py***