LIMPIEZA DE DATOS

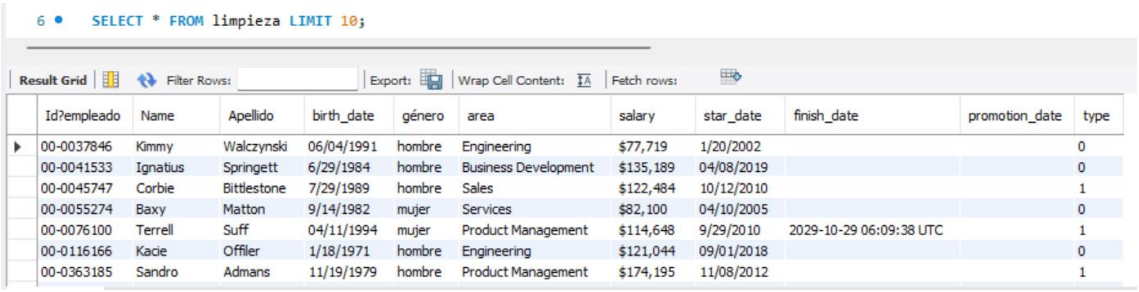
Francisco Olascoaga

Descripción del Proyecto

Objetivo: El objetivo de este proyecto es demostrar habilidades avanzadas en la limpieza y transformación de datos usando SQL, aplicadas a una tabla de empleados ficticia con diversos problemas de calidad de datos.

Este proyecto muestra un caso práctico de limpieza y transformación de datos en una base de datos SQL. La base de datos contiene información de empleados, incluyendo sus nombres, apellidos, fechas de nacimiento, géneros, áreas de trabajo, salarios, fechas de inicio y fin de contrato, fechas de promoción y tipo de trabajo. El objetivo de este proyecto es limpiar y estandarizar los datos para que sean consistentes y fáciles de analizar.

Inicialmente, los datos se encontraban como se muestran a continuación:



```
6 •  SELECT * FROM limpieza LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Id?empleado	Name	Apellido	birth_date	género	area	salary	star_date	finish_date	promotion_date	type
00-0037846	Kimmy	Walczynski	06/04/1991	hombre	Engineering	$77,719	1/20/2002			0
00-0041533	Ignatius	Springett	6/29/1984	hombre	Business Development	$135,189	04/08/2019			0
00-0045747	Corbie	Bittlestone	7/29/1989	hombre	Sales	$122,484	10/12/2010			1
00-0055274	Baxy	Matton	9/14/1982	mujer	Services	$82,100	04/10/2005			0
00-0076100	Terrell	Suff	04/11/1994	mujer	Product Management	$114,648	9/29/2010	2029-10-29 06:09:38 UTC		1
00-0116166	Kacie	Offiler	1/18/1971	hombre	Engineering	$121,044	09/01/2018			0
00-0363185	Sandro	Admans	11/19/1979	hombre	Product Management	$174,195	11/08/2012			1

En el análisis inicial de los datos, se identificaron inconsistencias significativas tanto en los nombres de las columnas como en los registros. Estas inconsistencias pueden obstaculizar el análisis preciso y la generación de informes fiables, ya que los datos no están uniformemente estructurados ni estandarizados.

Para abordar estos problemas y optimizar el proceso de limpieza y preparación de datos, he decidido implementar un procedimiento almacenado (STORE PROCEDURE). El procedimiento almacenado permite automatizar las tareas repetitivas y reducir el riesgo de errores humanos, facilitando así una preparación de datos más eficiente y confiable para el análisis subsiguiente.

```sql
-- CREO UN STORE PROCEDURE PARA EVITAR HACER SIEMPRE LA MISMA LLAMDA

DELIMITER //
CREATE PROCEDURE limp()
BEGIN
    select * from limpieza;
END //

DELIMITER ;

CALL limp();
```

Mediante un DESCRIBE obtengo las columnas que no son consistentes con su tipo de registro:

```sql
6 •  DESCRIBE limpieza;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Apellido	text	YES		NULL	
birth_date	text	YES		NULL	
género	text	YES		NULL	
area	text	YES		NULL	
salary	text	YES		NULL	
star_date	text	YES		NULL	
finish_date	text	YES		NULL	
promotion_date	text	YES		NULL	
type	int	YES		NULL	

Desactivo el modo de actualizaciones seguras para poder realizar cambios:

```sql
SET SQL_SAFE_UPDATES = 0;
```

Ahora, realizo los siguientes cambios:

```sql
ALTER TABLE limpieza CHANGE COLUMN `Id?empleado` `id_emp` VARCHAR(20) NULL;
ALTER TABLE limpieza CHANGE COLUMN `género` `gender` varchar(20) null;

ALTER TABLE limpieza ADD COLUMN temp_genero VARCHAR(20); -- creo una columna en forma de copia de seguridad
UPDATE limpieza SET temp_genero = género;

ALTER TABLE limpieza DROP COLUMN género;
ALTER TABLE limpieza CHANGE COLUMN `temp_genero` `gender` VARCHAR(20) NULL;
ALTER TABLE limpieza CHANGE COLUMN Apellido last_name varchar(50) null;
ALTER TABLE limpieza CHANGE COLUMN star_date start_date varchar(50) null;
ALTER TABLE limpieza CHANGE COLUMN Name name varchar(50) null;
```

Al no poder cambiar el nombre de la columna genero procedí por crear una columna adicional con el nombre 'temp_genero' para posteriormente mediante un SET traspasar los datos que se encontraban en la columna 'genero'. Elimino la columna genero y asigno la temporal como permanente con su correspondiente nombre 'gender'.

Duplicados:

En cuanto al análisis de encontrar registros duplicados procedí de la siguiente forma:

```sql
-- cantidad de duplicados

SELECT DISTINCT id_emp, count(*) as duplicados
FROM limpieza
GROUP BY id_emp
HAVING count(*)>1;

SELECT count(*) as cantiad_duplicados
```

Uso de una CTE:

```sql
-- En lugar de usar una subquery utilizo una CTE.
WITH total_duplicados AS (
    SELECT id_emp, COUNT(*) AS duplicados
    FROM limpieza
    GROUP BY id_emp
    HAVING COUNT(*) > 1
)
SELECT COUNT(*) AS cantidad_duplicados
FROM total_duplicados;
```

Al encontrar con registros duplicados procedo de la siguiente manera: renombro la tabla ‘limpieza’ a ‘conduplicados’ y luego creo una tabla temporal ‘sinduplicados’ mediante un DISTINCT. En base a esto, hago una comparación entre ambas tablas (conduplicados y sinduplicados) para finalmente crear otra vez una tabla ‘limpieza’ y mediante un

```sql
CREATE TABLE limpieza as
select * from templimpieza;
```

se conviente en la tabla principal eliminado la tabla ‘conduplicados’ mediante un ‘DROP TABLE conduplicados’;

La sintaxis de lo dicho es como sigue:

```sql
rename table limpieza to conduplicados;
-- creo una tabla temporal sin duplicados
create temporary table  templimpieza AS
SELECT DISTINCT * FROM conduplicados;

select count(*) as original FROM conduplicados;
-- 22223
SELECT COUNT(*) AS copia FROM templimpieza;
-- '22214'

CREATE TABLE limpieza as
select * from templimpieza;

-- finalmente elimino la tabla con duplicados
DROP TABLE conduplicados;
```

Limpieza de Datos de Texto

Mediante un SELECT verifico las columnas cuyos registros contienen inconsistencias y luego las cambio permanentemente:

```sql
SELECT TRIM(name) as name from limpieza;
SELECT TRIM(last_name) as last_name from limpieza;

SELECT name, trim(name) FROM limpieza
WHERE length(name) - length(trim(name)) > 0;

-- con esta CTE confirmo que la query anterior es correcta ya que no
-- me arroja espacios en el nombre, ahora puedo cambiar permanentemente la columna
with espacios as (
SELECT TRIM(name) as name from limpieza)
SELECT name FROM espacios
WHERE length(name) - length(trim(name)) > 0

-- cambio permanentenmente la columna name

UPDATE limpieza SET name = trim(name);
```

Procedo de la misma forma con la columna ‘last_name’.

```sql
-- hago lo mismo con la columna apellidos

SELECT last_name, trim(last_name)
FROM limpieza
WHERE length(last_name) - length(trim(last_name)) > 0;

-- verifico con una CTE que la query funcione para cambiar la tabla

WITH espacios_last_name AS (
SELECT TRIM(last_name) as last_name FROM limpieza)
SELECT length(last_name) - length(trim(last_name)) > 0 AS espacios
FROM espacios_last_name;

UPDATE limpieza SET last_name = trim(last_name);
```

A fines prácticos inserto espacios en la columna ‘área’ para posteriormente eliminarlos mediante la siguiente sintaxis:

```
-- ¿ que pasa si tengo espacios entre dos nombres en una columna?
-- introduzco espacios para poder corregirlos

UPDATE limpieza SET area = replace(area,' ', '       ');
CALL limp();

-- REGEXP '\\s{2,}': La función REGEXP busca patrones que coincidan con la expresión regular.
-- \\s: Representa un espacio en blanco (espacio, tabulación, etc.).
-- {2,}: Indica que debe haber al menos dos espacios en blanco.

-- nos muestra todos los registros que tienen dos o mas espacios
SELECT area from limpieza
WHERE area regexp '\\s{2,}';

SELECT area, trim(regexp_replace(area,'\\s{2,}',' ')) as ensayo
FROM limpieza;

UPDATE limpieza SET area = trim(regexp_replace(area,'\\s{2,}',' '));
```

Estandarizo datos de las columnas 'id_emp', 'gender' , 'type' y ''salary':

La columna 'id_emp' contiene ceros delante del número que son innecesarios y no aportan información útil. Estos ceros pueden causar inconsistencias y dificultar la manipulación y análisis de los datos. Por lo tanto, se procede a limpiarlos de la siguiente manera:

```
SELECT SUBSTRING(id_emp, 5) AS id_emp FROM limpieza;

UPDATE limpieza SET id_emp = SUBSTRING(id_emp, 5);


-- nuestro set de datos en gender esta en español, pero lo paso a ingles

SELECT gender,
CASE
    WHEN gender = 'hombre' THEN 'male'
    WHEN gender = 'mujer' THEN 'female'
    ELSE 'Other'
END AS gender1
FROM limpieza;

UPDATE limpieza SET gender = CASE WHEN gender = 'hombre' THEN 'male'
    WHEN gender = 'mujer' THEN 'female'
    ELSE 'Other'
END;
```

```sql
ALTER TABLE limpieza modify column type TEXT;

SELECT type,
CASE
    WHEN type = 1 THEN 'remote'
    WHEN type = 0 THEN 'hybrid'
    ELSE 'other'
END AS ejemplo
FROM LIMPIEZA;


update limpieza
SET type =
CASE
    WHEN type = 1 THEN 'remote'
    WHEN type = 0 THEN 'hybrid'
    ELSE 'other'
end;

call limp();
```

```sql
-- columna salario
-- esta consulta asi me arroja los valores que quiero pero en diferentes columnas
-- por lo tanto, debe ser anidada
select salary,
            replace(salary, '$',''),
            replace(salary, ',',''),
            TRIM(salary) as salario
FROM limpieza;

-- asi quedaria:

SELECT salary,
             CAST(TRIM(REPLACE(REPLACE(salary,'$',''),',','')) AS DECIMAL(15,2)) AS salary_new
FROM limpieza;

-- AHORA SI, CAMBIO LA COLUMNA PERMANENTEMENTE

UPDATE limpieza SET salary =  CAST(TRIM(REPLACE(REPLACE(salary,'$',''),',','')) AS DECIMAL(15,2));

alter table limpieza modify column salary int null;
```

SQL trabaja con el formato de fecha "año-mes-día" (YYYY-MM-DD), que es el formato de fecha estándar. Este formato es ampliamente utilizado porque es claro, no ambiguo y se ordena cronológicamente de forma natural.

La columna ‘birth_date’ se encuentra en varios formatos como “mes/día/año” (MM/DD/YYYY) y “mes-día-año” (MM-DD-YYYY), lo que es inconsistente y dificulta la realización de cálculos y comparaciones con fechas. Este tipo de inconsistencia puede causar errores en los análisis y reportes.

Para solucionar este problema, se utilizan dos funciones de SQL: STR_TO_DATE y DATE_FORMAT.

STR_TO_DATE: Esta función convierte una cadena de texto en una fecha según el formato especificado. Es útil cuando se tienen fechas almacenadas como texto y se necesita convertirlas a un tipo de dato DATE para realizar operaciones con ellas.

DATE_FORMAT: Esta función convierte una fecha en una cadena de texto según el formato especificado. Es útil cuando se necesita presentar una fecha en un formato específico para reportes o interfaces de usuario.

```sql
select birth_date from limpieza;

-- voy a darle formato año, mes, dia.
-- uso dos funciones str_to_date, date_format

SELECT birth_date, CASE
        WHEN birth_date like '%/%' THEN date_format(str_to_date(birth_date, '%m/%d/%y'),'%y-%m-%d')
         WHEN birth_date like '%-%' THEN date_format(str_to_date(birth_date, '%m-%d-%y'),'%y-%m-%d')
         ELSE null
    END AS new_birth_date
FROM limpieza;

UPDATE limpieza SET birth_date = CASE
            WHEN birth_date like '%/%' THEN date_format(str_to_date(birth_date, '%m/%d/%y'),'%y-%m-%d')
            WHEN birth_date like '%-%' THEN date_format(str_to_date(birth_date, '%m-%d-%y'),'%y-%m-%d')
            ELSE null
END;
```

Verifico que la query sea correcta para cambiar permanentemente la columna de la siguiente manera:

```
    UPDATE limpieza
SET birth_date = CASE
    WHEN birth_date LIKE '%/%' THEN
        DATE_FORMAT(STR_TO_DATE(birth_date, '%m/%d/%Y'), '%Y-%m-%d')
    WHEN birth_date LIKE '%-%' THEN
        DATE_FORMAT(STR_TO_DATE(birth_date, '%m-%d-%Y'), '%Y-%m-%d')
    ELSE
        NULL
END;

ALTER TABLE limpieza modify column birth_date date;
CALL limp();
```

Hago el mismo procedimiento en la columna ‘start_date’

```
UPDATE limpieza
SET start_date = CASE
    WHEN start_date LIKE '%/%' THEN
        DATE_FORMAT(STR_TO_DATE(start_date, '%m/%d/%Y'), '%Y-%m-%d')
    WHEN start_date LIKE '%-%' THEN
        DATE_FORMAT(STR_TO_DATE(start_date, '%m-%d-%Y'), '%Y-%m-%d')
    ELSE
        NULL
END;

ALTER TABLE limpieza modify column start_date date;
```

Problemas con la Columna ‘finish_date’

En cuanto a la columna ‘finish_date,’ me encuentro con dos problemas principales:

Formato Fecha-Hora: Los registros están en formato fecha-hora, lo que incluye tanto la fecha como la hora en un solo campo.

Espacios Vacíos: La columna contiene registros con espacios vacíos, los cuales arrojarían errores al intentar realizar consultas y cálculos de fechas.

Inicialmente se encuentra así:

```
10 •    SELECT finish_date FROM limpieza;
```

Result Grid | Filter Rows: | Export:

finish_date
2029-10-29 06:09:38 UTC
2008-12-05 01:21:48 UTC

Proceso a realizar la limpieza:

```
-- copia de seguridad de finish_date

ALTER TABLE limpieza ADD COLUMN date_backup text;

UPDATE limpieza set date_backup = finish_date;

-- ya tenemos nuestra copia de respaldo

SELECT finish_date, str_to_date(finish_date, '%Y-%m-%d %H:%i:%s') AS fecha from limpieza;

UPDATE limpieza SET finish_date = str_to_date(finish_date, '%Y-%m-%d %H:%i:%s')
WHERE finish_date <> '';

UPDATE limpieza
SET finish_date = STR_TO_DATE(REPLACE(finish_date, ' UTC', ''), '%Y-%m-%d %H:%i:%s')
WHERE finish_date <> '';
call limp();
```

```
ALTER TABLE limpieza
    add column fecha date,
    add column hora time;
```

```sql
UPDATE limpieza SET
fecha = date(finish_date),
hora = time(finish_date)
WHERE finish_date is not null and finish_date <>'';


-- como en nuestra columna tenemos espacios en blanco cuando hagamos una actualizacion nos arrojara error
-- convierto los espacios en blancos en valores nulos

 UPDATE limpieza SET finish_date = null where finish_date = '';

ALTER TABLE limpieza modify column finish_date datetime;

DESCRIBE limpieza;
```

Finalmente, hago un calculo con las fechas para insertar una nueva columna ‘age’. Para ello utilizo CURDATE, es una función en SQL que se utiliza para obtener la fecha actual del sistema en el formato YYYY-MM-DD.

```sql
-- calculos con fechas
ALTER TABLE limpieza ADD column age Int;

SELECT birth_date, start_date, timestampdiff(year,birth_date, start_date) AS edad_ingreso FROM limpieza;

-- sacamos la edad de los empleados


UPDATE limpieza SET age = timestampdiff(year, birth_date, curdate());

SELECT name, birth_date, age FROM limpieza;
```

Por último, creo una ultima columna ‘correo’ el cual lo realizo mediante un CONCAT y SUBSTRING.

```sql
-- creo un correo electronico que se componga del primer nombre del empleado
-- luego un guion bajo y luego las dos primeras letras del apellido
-- uso subtreing y concat


SELECT CONCAT(SUBSTRING_INDEX(name, ' ', 1), '_', SUBSTRING(last_name, 1, 2), '.', SUBSTRING(type, 1, 1), '@consulting.com') AS email
FROM Limpieza;

alter table limpieza ADD COLUMN email varchar(100);

UPDATE limpieza SET
email = CONCAT(SUBSTRING_INDEX(name, ' ', 1), '_', SUBSTRING(last_name, 1, 2), '.', SUBSTRING(type, 1, 1), '@consulting.com');


SELECT email from limpieza;
```