




































<https://github.com/yoyusD/Prueba-tcnica-Data-Engineer>

[illegible]

Inherited from table(s)

| Columns   |  |   |                  |       |   |   |                      |  |
|---|--|---|------------------|-------|---|---|----------------------|--|
|   | Name   | Data type   | Length/Precision | Scale | Not NULL?   | Primary key?  | Default              |  |
|  |  id           | character varying    | 40               |       |  |  | <input type="text"/> |  |
|  |  company_name | character varying    | 130              |       |  |  | <input type="text"/> |  |
|  |  company_id   | character varying    | 40               |       |  |  | <input type="text"/> |  |
|  |  amount       | numeric              | 37               | 2     |  |  | <input type="text"/> |  |
|  |  status       | character varying    | 30               |       |  |  | <input type="text"/> |  |
|  |  created_at   | date                 |                  |       |  |  | <input type="text"/> |  |
|  |  updated_at   | date                 |                  |       |  |  | <input type="text"/> |  |

*Figura 2 Parámetros de las columnas.*

## 1.2 Extracción

Para la extracción de la información de la base de datos. Se escogió el lenguaje de programación Python, junto con la librería de Pandas, la cual es ideal para el procesamiento de datos.

Escogí este lenguaje por mi experiencia previa trabajando con este, junto con librerías de análisis de datos.

## 1.3 Transformación

En esta parte se presentaron algunos retos. Como, por ejemplo, cambiar las columnas “updated\_at” y “created\_at” en formato datetime en formato ISO. Esto con la finalidad de que PostgreSQL pueda manejar el tipo de dato *date*.

Se encontraron los siguientes datos en la base de datos. Estos fueron tomados en cuenta para el análisis, definimos así un límite de (37,2) de la columna “amount” en tipo de dato numérico. Permitiendo así trabajar con la cantidad máxima que se presenta en nuestra base de datos.

| id                                       | company_name | company_id                               | amount   | status          | created_at | updated_at |
|--|--------------|--|----------|-----------------|------------|------------|
| a07c65559e91a7b08a25f6c7e9f130cdfb727a42 | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 3.00E+34 | voided          | 17/04/2019 |            |
| 37fd630d05e0b4c9c179eea6f88541e27adfb07  | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 2.13E+18 | pending_payment | 27/03/2019 |            |
| a98245b3f1e8484f2e65f245ef35051822d0f4fb | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 2.13E+16 | pending_payment | 09/05/2019 |            |

Establecemos “id” y “company\_id” con una extensión de 40 caracteres, dado que esta es la longitud estándar que podemos encontrar en los datos de la columna id. Y se encontraron 3 valores vacíos en la columna de “id”.

| id | company_name | company_id                               | amount | status          | created_at | updated_at |
|----|--------------|--|--------|-----------------|------------|------------|
|    | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 66.16  | pending_payment | 14/03/2019 |            |
|    | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 55.71  | paid            | 13/02/2019 | 13/02/2019 |
|    | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 89.36  | paid            | 17/04/2019 | 17/04/2019 |

Además, se identificaron 3 datos donde “created\_at” tiene valores *NULL*.

| id                                       | company_name | company_id                               | amount | status          | created_at | updated_at |
|--|--------------|--|--------|-----------------|------------|------------|
| 870f7a34b425bd0bf44660377637a40bc8df65df | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 55.75  | paid            |            | 21/01/2019 |
| 5300670ad03ce15b984a03e5ab49442d61f70c01 | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 55.75  | paid            |            | 28/02/2019 |
| e5902356495f62a5f8c6f96190a9dd5f0e10c780 | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 91.27  | pending_payment |            |            |

También encontramos 3 valores *NULL* en la columna de “company\_name”.

| id                                       | company_name | company_id                               | amount | status | created_at | updated_at |
|--|--------------|--|--------|--------|------------|------------|
| ec79a21ef969c7fc6beef080ff56baf0aeeca8b5 |              | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 112.52 | paid   | 14/02/2019 | 14/02/2019 |
| 4740cf2624c3b929d9944cdfcb5c87e71a82ddc5 |              | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 118.78 | paid   | 05/03/2019 | 06/03/2019 |
| 6a6ac16d53a02ba7948bff0a534e45404e716c5b |              | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 244.88 | voided | 09/05/2019 |            |

En “status” se encontró un valor con números hexadecimales.

| id                                       | company_name | company_id                               | amount | status | created_at | updated_at |
|--|--------------|--|--------|--------|------------|------------|
| 8ee1ad3b1736f801145e18ecf54e346c006e43d5 | MiPasajefy   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 146.74 | 0xFFFF | 03/02/2019 | 03/02/2019 |

Y por último encontramos, en “company\_name”, nombres que no están en concordancia con los demás, diferenciándose con un patrón de números hexadecimales 0xFFFF.

| id                                       | company_name     | company_id                               | amount | status | created_at | updated_at |
|--|------------------|--|--------|--------|------------|------------|
| 891732cfc6980d71a24ccba4138a3af2f14413fb | <b>MiP0xFF</b>   | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 47.64  | voided | 17/03/2019 |            |
| 60366f3cfb5245ec62143984eef4ba93cfbe3c38 | <b>MiPas0xFF</b> | cbf1c8b09cd5b549416d49d220a40cbd317f952e | 134.99 | paid   | 25/01/2019 | 25/01/2019 |

Estos valores de las tablas se optaron por omitirlos en la transformación de datos de prueba, con la finalidad de poder realizar consultas en la base de datos sin el mayor de los problemas.

**El código de transformación se presenta en el repositorio de github.**

## 1.4 Dispersión de la información

Creamos las tablas con el siguiente query:

```
1 CREATE TABLE companies (  
2   company_id VARCHAR(40) PRIMARY KEY,  
3   company_name VARCHAR(130)  
4 );  
5  
6 CREATE TABLE charges (  
7   id VARCHAR(40) PRIMARY KEY,  
8   company_id VARCHAR(40) NOT NULL,  
9   amount DECIMAL(37,2) NOT NULL,  
10  status VARCHAR(30) NOT NULL,  
11  created_at TIMESTAMPTZ,  
12  updated_at TIMESTAMPTZ,  
13  CONSTRAINT fk_company  
14  FOREIGN KEY (company_id)  
15  REFERENCES companies(company_id)  
16 );
```

Importamos los datos que procesamos en Python previamente.

*data\_company.csv*

*data\_charge.csv*

Una vez importados en PostgreSQL. Utilizamos ERD Tool para poder visualizar las tablas creadas. Además, creamos la relación entre las tablas.

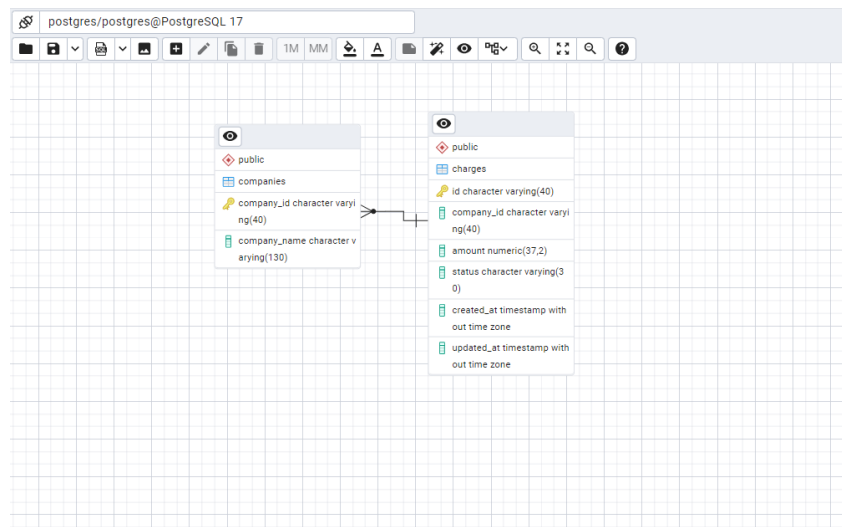


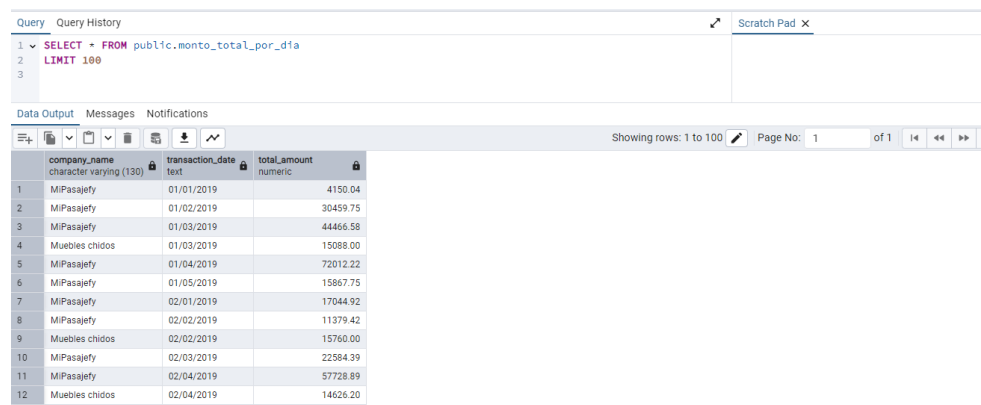
Figura 3 Vista de las tablas en ERD Tool

## 1.5 SQL

Diseñamos una vista. Este diseño se hizo para ver el monto total transaccionado por día para las diferentes compañías. (Se escogió el formato DD/MM/YYYY, para la visualización de las fechas)

```
1 CREATE OR REPLACE VIEW monto_total_por_dia AS SELECT
2   c.company_name, to_char(ch.created_at, 'DD/MM/YYYY') AS transaction_date, SUM(ch.amount) AS total_amount
3   FROM charges ch JOIN companies c ON ch.company_id = c.company_id
4   GROUP BY c.company_name, to_char(ch.created_at, 'DD/MM/YYYY')
5   ORDER BY transaction_date, c.company_name;
6
```

Ejecutamos la vista nombrada “monto\_total\_por\_dia”. Visualizamos las 100 primeras columnas.



|    | company_name<br>character varying (130) | transaction_date<br>text | total_amount<br>numeric |
|----|---|--------------------------|-------------------------|
| 1  | MI Pasajefy                             | 01/01/2019               | 4150.04                 |
| 2  | MI Pasajefy                             | 01/02/2019               | 30459.75                |
| 3  | MI Pasajefy                             | 01/03/2019               | 44466.58                |
| 4  | Muebles chidos                          | 01/03/2019               | 15088.00                |
| 5  | MI Pasajefy                             | 01/04/2019               | 72012.22                |
| 6  | MI Pasajefy                             | 01/05/2019               | 15867.75                |
| 7  | MI Pasajefy                             | 02/01/2019               | 17044.92                |
| 8  | MI Pasajefy                             | 02/02/2019               | 11379.42                |
| 9  | Muebles chidos                          | 02/02/2019               | 15760.00                |
| 10 | MI Pasajefy                             | 02/03/2019               | 22584.39                |
| 11 | MI Pasajefy                             | 02/04/2019               | 57728.89                |
| 12 | Muebles chidos                          | 02/04/2019               | 14626.20                |

Figura 4 Ejecución de la vista.

## Sección 2: Scala

Con forme al problema y las especificaciones siguientes:

*Problema:* Calcular el numero faltante de un conjunto de los primeros 100 números naturales del cual se extrajo uno.

*Especificaciones:*

- La aplicación debe de implementarse en el lenguaje Scala
- Se debe de implementar una clase que represente al conjunto de los primero 100 números
- La clase implementada debe de tener el método Extract para extraer un cierto numero deseado
- La clase implementada debe de poder calcular que numero se extrajo y presentarlo
- Debe de incluir validación del input de datos (numero, número menor de 100)
- La aplicación debe de poder ejecutarse con un argumento introducido por el usuario que haga uso de nuestra clase y muestre que pudo calcular que se extrajo ese número

Desarrollamos el programa en Scala.

Para ello fue necesario descargar el sdk en el bash. Utilizamos SWL de Windows.

```
curl -s "https://get.sdkman.io"  
source "$HOME/.sdkman/bin/sdkman-init.sh"
```

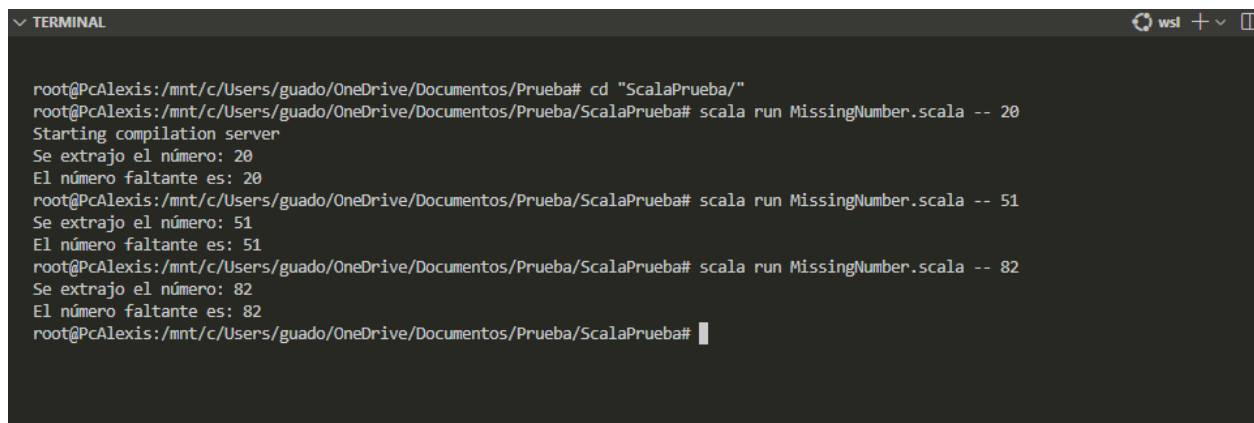
Una vez instalado el sdk, proseguimos por la instalación de Scala.

```
sdk install scala
```

Con Scala ya instalado podremos compilar y ejecutar aplicaciones con extensión scala.

### **El programa de Scala se encuentra en el repositorio de github.**

Los resultados obtenidos del programa se presentan a continuación:

A screenshot of a terminal window titled "TERMINAL" with a "wsl" icon in the top right corner. The terminal shows a series of commands and their outputs. The user navigates to a directory and runs a Scala program named "MissingNumber.scala" three times with different arguments: 20, 51, and 82. Each run produces the same output: "Starting compilation server", "Se extrajo el número: [argument]", and "El número faltante es: [argument]".

```
root@PcAlexis:/mnt/c/Users/guado/OneDrive/Documentos/Prueba# cd "ScalaPrueba/"  
root@PcAlexis:/mnt/c/Users/guado/OneDrive/Documentos/Prueba/ScalaPrueba# scala run MissingNumber.scala -- 20  
Starting compilation server  
Se extrajo el número: 20  
El número faltante es: 20  
root@PcAlexis:/mnt/c/Users/guado/OneDrive/Documentos/Prueba/ScalaPrueba# scala run MissingNumber.scala -- 51  
Se extrajo el número: 51  
El número faltante es: 51  
root@PcAlexis:/mnt/c/Users/guado/OneDrive/Documentos/Prueba/ScalaPrueba# scala run MissingNumber.scala -- 82  
Se extrajo el número: 82  
El número faltante es: 82  
root@PcAlexis:/mnt/c/Users/guado/OneDrive/Documentos/Prueba/ScalaPrueba#
```

Utilizamos `scala run "Programa.scala" - - "Argumento"`. Para poder ejecutar nuestro programa haciendo uso, precisamente, de argumentos.