

# Resumen comandos para la Ciencia de Datos

Gabriel Pinilla

26 de octubre de 2023



# Índice general

<b>1. SQL en Postgres</b>	<b>1</b>
1.1. Notas previas...	1
1.2. Comandos básicos	1
1.2.1. Estructura básica de una consulta	1
1.2.2. Where vs Having	2
1.2.3. Crear una base de datos	2
1.2.4. Crear una tabla	2
1.2.5. Insertar datos a una tabla	2
1.2.6. Ver todos los datos	2
1.3. Limpieza, orden y transformación de datos	3
1.3.1. Manipulación de filas y columnas	3
1.3.2. Orden	4
1.3.3. Conversión de tipos	4
1.4. Análisis y visualización	5
1.4.1. Cálculos matemáticos básicos	5
1.4.2. Número de caracteres de cada fila	5
1.4.3. Extraer un substring de un string	6
1.4.4. Devolver valores nulos	6
1.4.5. Subconsulta	6
1.4.6. Consultas en múltiples tablas	6
<b>2. Terminal Windows</b>	<b>9</b>
2.1. Comandos comunes	9
<b>3. Excel y Google Sheets</b>	<b>11</b>
3.1. Notas previas...	11
3.2. Comandos básicos	11
3.3. Limpieza, orden y transformación de datos	11
3.3.1. Filtrado	11
3.3.2. Orden	12
3.3.3. Quitar duplicados	12

3.3.4.	Funciones lógicas . . . . .	12
3.3.5.	Manipulación de strings . . . . .	12
3.3.6.	Formatos de fechas . . . . .	13
3.3.7.	BURSCARV y BURSCARH . . . . .	13
3.3.8.	Formato de datos . . . . .	14
3.3.9.	Formato condicional . . . . .	14
3.4.	Análisis y visualización . . . . .	14
3.4.1.	Cálculos matemáticos básicos . . . . .	14
3.4.2.	Cálculos con múltiples criterios . . . . .	15
3.4.3.	Tablas dinámicas . . . . .	15
3.4.4.	Creación de gráficos . . . . .	15
3.5.	Importación de datos . . . . .	15
3.6.	Errores . . . . .	15
<b>4.</b>	<b>Python</b>	<b>17</b>
4.1.	Notas previas... . . . .	17
4.2.	Librerías . . . . .	17
4.2.1.	Importación de datos . . . . .	17
4.3.	Conceptos previos . . . . .	18
4.3.1.	Manipulación de strings . . . . .	18
4.3.2.	Estructuras de datos . . . . .	19
4.4.	Limpieza, orden y transformación de datos . . . . .	21
4.4.1.	Bibliotecas . . . . .	21
4.4.2.	Tipos de datos de Pandas . . . . .	22
4.4.3.	Resúmenes de datos . . . . .	22
4.4.4.	Manipulación de datos . . . . .	22
4.5.	Análisis . . . . .	22
4.6.	Visualización . . . . .	22
4.7.	Análisis y visualización . . . . .	22
4.7.1.	Cálculos matemáticos . . . . .	22
<b>5.</b>	<b>R</b>	<b>25</b>
5.1.	Notas previas... . . . .	25
5.2.	Paquetes . . . . .	25
5.3.	Limpieza . . . . .	26
5.3.1.	Resúmenes de datos . . . . .	26
5.4.	Orden . . . . .	26
5.5.	Transformación de datos . . . . .	27
5.6.	Cálculos matemáticos . . . . .	27
5.7.	Visualización . . . . .	27
5.7.1.	Suavizado . . . . .	29

5.7.2.	Estética y facetas . . . . .	29
5.7.3.	Guardar gráficas . . . . .	29
<b>6.</b>	<b>Markdown</b>	<b>31</b>
<b>7.</b>	<b>Git y GitHub</b>	<b>33</b>
7.1.	Git . . . . .	33
7.1.1.	Repositorios . . . . .	33
7.1.2.	.gitignore . . . . .	34
7.1.3.	Ramas . . . . .	34
<b>8.</b>	<b>Tableau Public</b>	<b>37</b>
8.1.	Creación de visualizaciones . . . . .	37
8.1.1.	Importación de datos y conexiones . . . . .	37
8.1.2.	Campos calculados . . . . .	38
8.1.3.	Conceptos y elementos de Tableau . . . . .	40
8.1.4.	Filtros . . . . .	41
8.2.	Creación de visualizaciones básicas . . . . .	43
8.3.	Diseño y personalización de paneles . . . . .	49
8.3.1.	Funciones de agregación . . . . .	50
8.3.2.	Funciones LOD . . . . .	50
8.4.	Dashboards . . . . .	51
8.4.1.	Creación de un dashboard . . . . .	51
<b>9.</b>	<b>Glosario de términos de programación</b>	<b>53</b>



# Capítulo 1

## SQL en Postgres

### 1.1. Notas previas...

1. Al finalizar una consulta se debe terminar con ;.
2. El texto se debe escribir entre comillas simples o dobles, dependiendo del programa a utilizar.
3. En SQL, el primer caracter de un string inicia con ' y se escriben entre comillas simples.
4. Para insertar comentarios se debe usar --

### 1.2. Comandos básicos

#### 1.2.1. Estructura básica de una consulta

La siguiente consulta muestra la estructura básica y el orden que deben seguir los comandos:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE condicion
4 GROUP BY columnas
5 HAVING condicion
6 ORDER BY columnas DESC
7 LIMIT numero;
```

Si se requieren todas las columnas de la tabla, se debe agregar un \* después del comando **SELECT**.

### 1.2.2. Where vs Having

En esta consulta, la cláusula **WHERE** filtra y después el **GROUP BY** agrupa:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE condicion
4 GROUP BY columnas;
```

Sin embargo, en la siguiente consulta el **GROUP BY** agrupa y luego el **HAVING** filtra:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 GROUP BY columnas
4 HAVING condicion;
```

### 1.2.3. Crear una base de datos

Para crear una base de datos:

```
1 CREATE DATABASE nom-base-de-datos;
```

### 1.2.4. Crear una tabla

Para crear una tabla:

```
1 CREATE TABLE nom-tabla (
2     col1 tipo-de-variable,
3     col2 tipo-de-variable,
4 );
```

El comando **CREATE TABLE IF NOT EXISTS** crea una tabla si es que dicha tabla no existe en la base de datos.

### 1.2.5. Insertar datos a una tabla

Insertar datos a una tabla ya creada:

```
1 INSERT INTO tabla (col-sep-por-coma)
2 VALUES (valores-sep-por-coma);
```

### 1.2.6. Ver todos los datos

Para ver la tabla completa:

```
1 SELECT *
2 FROM conj-de-datos.tabla;
```



## 1.3. Limpieza, orden y transformación de datos

### 1.3.1. Manipulación de filas y columnas

Para eliminar registros:

```
1 DELETE FROM tabla
2 WHERE nom-col = condicion;
```

Agregar una columna:

```
1 ALTER TABLE tabla
2 ADD columna tipo-de-dato;
```

Eliminar una datos de una columna:

```
1 DELETE conj-de-datos.tabla
2 WHERE condicion;
```

Eliminar una columna:

```
1 ALTER TABLE tabla
2 DROP COLUMN columna;
```

Actualizar datos:

```
1 UPDATE conj-de-datos.tabla
2 SET nom-col = valor-a-cambiar
3 WHERE condicion;
```

Eliminar espacios en blanco:

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 TRIM (columna)=criterio;
```

Borrar una base de datos:

```
1 DROP DATABASE base-de-datos;
```

Borrar una tabla:

```
1 DROP TABLE tabla;
```

Evitar selecciones duplicadas:

```
1 SELECT
2 DISTINCT (columna)
3 FROM conj-de-datos.tabla
4 WHERE condicion;
```

Para contar valores distintos:

```
1 SELECT COUNT(DISTINCT(col))
2 FROM conj-de-datos.tabla ;
```

¿Cuántas veces aparece cada elemento?

```
1 SELECT columna, COUNT (*)
2 FROM conj-de-datos.tabla
3 GROUP BY columnas;
```

### 1.3.2. Orden

Ordenar datos:

```
1 SELECT columnas
2 FROM conj-de-datos.tabla
3 ORDER BY columna DESC;
```

En las consultas SQL, la cláusula **WHERE** filtra los datos según la condición que se le otorgue. Se pueden ordenar los datos en orden ascendente (**ASC**) o descendente (**DESC**)

### 1.3.3. Conversión de tipos

Para convertir los datos de un tipo a otro se usa la función **CAST**:

```
1 SELECT
2 CAST(columna AS INT) \\
3 FROM conj-de-datos.tabla; \\
```

Los tipos de datos más comunes son:

1. **SMALLINT**
2. **INTEGER** o **INT**
3. **BIGINT**
4. **SERIAL**: Número entero que se autoincrementa.
5. **NUMERIC** o **DECIMAL**
6. **VARCHAR()**: Cadena de caracteres de longitud variable.
7. **TEXT**: Cadena de caracteres de longitud variable que no tiene límite.
8. **DATETIME**
9. **DATE**: Año, mes, día (formato standar).
10. **TIME**: Hora, minuto, segundo.
11. **TIMESTAMP**: Fecha y hora.

12. **BOOLEAN**: TRUE o FALSE.

Si la consulta anterior falla, se puede usar la siguiente consulta:

```
1 SELECT
2 SAFE_CAST(columna AS TIPO-DE-DATO)
3 FROM conj-de-datos.tabla;
```

## 1.4. Análisis y visualización

### 1.4.1. Cálculos matemáticos básicos

Suma (**SUM**) y promedio (**AVG**):

```
1 SELECT col, SUM(col2), AVG(col2)
2 FROM conj-de-datos.tabla
3 GROUP BY col;
```

Mínimo y máximo: Devuelve el valor mínimo de una columna con el nombre de min-columna. Lo mismo para el máximo.

```
1 SELECT MIN(columna) AS min-columna,
2 MAX(col2) AS max-columna
3 FROM conj-de-datos.tabla;
```

Conteo de registros:

```
1 SELECT COUNT (*)
2 FROM conj-de-datos.tabla;
```

La consulta anterior contará todos los registros de la tabla. Si se le agrega la cláusula **WHERE** al final, cuenta los registros dependiendo de la condición que tenga.

### 1.4.2. Número de caracteres de cada fila

Para que devuelva el número de caracteres:

```
1 SELECT
2 LENGTH(col-para-comprobar-su-largo)
3 FROM conj-de-datos.tabla;
```

También se puede escribir como:

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 LENGTH(columna)=criterio;
```

En algunos programas usan **LEN**.

Ejemplo:

```
1 SELECT columna1
2 FROM conj-de-datos.tabla
3 WHERE
4 LENGTH(columna1)=5;
```

Esta consulta devolverá todos los registros de la columna 1 que tengan 5 caracteres.

### 1.4.3. Extraer un substring de un string

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE
4 SUBSTRS (col, inicio, num-de-letras-a-extraer) + criterio;
```

### 1.4.4. Devolver valores nulos

```
1 SELECT columna
2 FROM conj-de-datos.tabla
3 WHERE columna IS NULL;
```

### 1.4.5. Subconsulta

Es una consulta dentro de otra. Se ejecutan desde la más interna hacia la más externa y van entre paréntesis:

```
1 SELECT col-sep-por-coma
2 FROM conj-de-datos.tabla
3 WHERE col > (SELECT col-sep-por-coma
4              FROM conj-de-datos.tabla);
```

### 1.4.6. Consultas en múltiples tablas

Para hacer consultas en múltiples tablas, se puede hacer de dos maneras. La primera es usar **SELECT** y **FROM**:

```
1 SELECT tabla1.*, tabla2.*
2 FROM tabla1, tabla 2
3 WHERE tabla1.columna = tabla2.columna;
```

La segunda es usando **JOIN**, los cuales se usan para unir dos tablas bajo la siguiente sintaxis:

```
1 SELECT *  
2 FROM conj-de-datos.tabla1  
3 INNER JOIN tabla2  
4 ON tabla1.col= tabla2.col;
```

La primera tabla o tabla izquierda siempre irá junto al **FROM**, luego irá la segunda tabla o tabla derecha. Se lee *Tabla 1 hará un cruce con la tabla 2*.

Existen 4 formas de cruzar datos:

1. **INNER JOIN**: Devuelve los registros con valores coincidentes de ambas tablas. No muestra valores nulos.



Figura 1.1: Diagrama de Venn de INNER JOIN

2. **LEFT JOIN**: Devuelve todos los registros de la tabla izquierda y solo los registros coincidentes de la tabla derecha. Tanto en el **LEFT JOIN** como el **RIGHT JOIN** van a mostrar valores nulos. Si hace el cruce con columnas que tienen valores nulos, no cruza esos valores.



Figura 1.2: Diagrama de Venn de LEFT JOIN

3. **RIGHT JOIN**: Devuelve todos los registros de la segunda tabla y solo los coincidentes de la primera tabla.



Figura 1.3: Diagrama de Venn de RIGHT JOIN

4. **FULL JOIN**: Une todos los registros de las dos tablas, sean coincidentes o no.



Figura 1.4: Diagrama de Venn de FULL JOIN

5. **CROSS JOIN**: Combina cada uno de los registros de una tabla con los registros de la otra. No se hace sobre una clave, es decir, no va el **ON**. La tabla resultante tendrá un número de filas igual al producto entre los números de filas de cada tabla, repitiéndose los datos de estas. Debido a esto, ocupa más recursos.

# Capítulo 2

## Terminal Windows

### 2.1. Comandos comunes

1. \d tabla: muestra información de una tabla y los tipos de datos de cada columna.
2. \c base-de-datos: conectarse a una base de datos.
3. \dt: da una lista de todas las bases de datos.
4. \q: salir de la terminal.
5. nom-programa version: verifica la versión instalada del programa en el sistema.
6. clear: limpia la pantalla.





# Capítulo 3

## Excel y Google Sheets

### 3.1. Notas previas...

- Algunas funciones de Excel son parecidas a las de Google Sheets.
- En Google Sheets se usan ; y en Excel ,.
- Para los rangos se usa C:C, por ejemplo: =MAX(A3:A9).

### 3.2. Comandos básicos

La sintaxis de una función en Excel y Google Sheets es =NOM-FUNCION(argumento1; argumento2;...), donde los argumentos son los valores que se usan como entrada para la función.

Se puede definir la hoja de cálculo con 'nom-hoja'!rango, por ejemplo: =CONTAR.SI('hoja 1'!G:G). En este ejemplo toma toda la columna G de la hoja 1.

### 3.3. Limpieza, orden y transformación de datos

#### 3.3.1. Filtrado

=FILTRAR(rango; fila-o-col = filtro;"" ): Devolverá todos los registros del rango, y si no hay devuelve una cadena vacía ("" ). Se pueden crear filtros en Excel: Datos > Filtro.

### 3.3.2. Orden

Para ordenar datos en hojas de cálculo, se debe seleccionar Datos > Ordenar. También se puede hacer mediante la función =ORDENAR().

En el siguiente comando, FALSO (o -1 en Excel) indica que el orden es descendiente. Por otro lado, VERDADERO (o 1 en Excel) señala orden ascendente.  
=ORDENAR(rango-para-ordenar;segun-columna;FALSO)

Usando =ORDENARPOR():

=ORDENARPOR(rango;col-1; FALSO; col-2;VERDADERO)

### 3.3.3. Quitar duplicados

Para quitar duplicados en Excel: Datos > Quitar espacios duplicados

### 3.3.4. Funciones lógicas

Las funciones lógicas utilizadas en Excel y Google Sheets son:

- Y(expresion1;expresion2;...)
- O(expresion1;expresion2;...)
- NO(valor-logico)
- =SI(expresion;valor-si-verdadero;valor-si-falso)
- =SI.ERROR(valor;valor-si-error)

### 3.3.5. Manipulación de strings

1. =ESPACIOS(valores): Elimina espacios del texto.
2. =MAYUSC(celda): Cambia de minúscula a mayúscula.
3. =MINUSC(celda): Cambia de mayúscula a minúscula.
4. =NOMPROPIO(celda): Cambia a nombre propio.
5. =CONCATENAR(): Une dos o más cadenas de texto en una celda.
6. =ENCONTRAR(): Busca una cadena de texto y devuelve su posición.
7. =DERECHA(): Devuelve el número de caracteres iniciando desde la derecha.

8. =IZQUIERDA(): Devuelve el número de caracteres iniciando desde la izquierda.
9. =EXTRAE(): Devuelve un número específico de caracteres de una posición.
10. =LARGO(): Devuelve la longitud de una cadena de texto.
11. =REEMPLAZAR(): Reemplaza una cadena de texto por otra en una celda.

### 3.3.6. Formatos de fechas

1. =HOY(): Devuelve la fecha actual.
2. =AHORA(): Devuelve fecha y hora actual.
3. =FECHA(): Crea una fecha. Ej: =FECHA(2023,6,10).
4. =DIAS360(): Calcula el número de días entre dos fechas en un año de 365 días.
5. =DIAS.LAB(): Calcula el número de días laborales entre dos fechas.
6. =DIAS(): Calcula el número de días entre dos fechas.
7. =MES(): Devuelve el número de mes de una fecha.
8. =AÑO(): Devuelve el año de una fecha.
9. =DIASEM(): Devuelve el número de día de la semana de 1 a 7 para una fecha.
10. =DIAS.LAB.INTL(): Calcula el número de días laborales entre dos fechas usando una definición personalizada de días laborales.
11. =FIN.MES(): Devuelve la fecha del último día del mes antes o después de un número determinado de meses.

### 3.3.7. BURSCARV y BURSCARH

Para relacionar tablas y buscar datos de una tabla a partir de una clave de búsqueda, se usan los comandos BUSCARV y BUSCARH. La principal diferencia entre ellos es que BUSCARV solo realiza la búsqueda en una columna, mientras que BUSCARH lo hace con las filas.

Si se escribe FALSO, la coincidencia será exacta. Si se escribe VERDADERO, la coincidencia será cercana. En Excel 1 es verdadero y 0 es falso:

=BUSCARV(valor-buscado; rango-de-busqueda; num-para-indicar-col-de-busqueda; FALSO)

### 3.3.8. Formato de datos

Se usa la función =CONVERTIR() se usa para transformar un número de un sistema de medición a otro.

=CONVERTIR(num-a-convertir;unidad-del-num;unidad-resultado)

### 3.3.9. Formato condicional

El formato condicional es una herramienta que se utiliza para identificar tendencias resaltándolas con colores. Se le añade una condición para que cambie el aspecto de la celda.

Se puede aplicar el formato condicional de la siguiente forma:

Excel: Se debe seleccionar el rango de celdas donde se quiera aplicar el formato > Inicio > Formato condicional > Reglas para resaltar celdas.

Google Sheets: Se debe seleccionar el rango de celdas donde se quiera aplicar el formato > Formato > Formato condicional.

## 3.4. Análisis y visualización

### 3.4.1. Cálculos matemáticos básicos

1. =SUMA(rango)
2. =PROMEDIO(rango)
3. =MIN(rango)
4. =MAX(rango)
5. =RESIDUO(): Da como resultado el resto cuando al dividir dos números.
6. =CONTAR.SI(rango; "criterio"): Cuenta el número de celdas que cumplen con un criterio.
7. =SUMAR.SI(rango; "criterio"): Suma los valores de un rango si cumplen con el criterio.
8. =SUMAPRODUCTO(matriz1;matriz2;...): Multiplica las matrices y muestra el resultado de la suma de esos productos.

### 3.4.2. Cálculos con múltiples criterios

1. =SUMAR.SI.CONJUNTO(rango-suma;rango-criterio1;criterio1;rango-criterio2;criterio2;...)
2. =CONTAR.SI.CONJUNTO(rango-criterio1;criterio1;rango-criterio2;criterio2;...)
3. =MAX.SI.CONJUNTO(rango-max;rango1;criterio1;rango2;criterio2;...)

### 3.4.3. Tablas dinámicas

Una tabla dinámica o Pivot Table es una tabla que resume, calcula y analiza datos para observar tendencias entre ellos.

Para crear una tabla dinámica tanto en Excel como en Google Sheets, se debe seleccionar Insertar > Tabla dinámica, luego seleccionar los datos preferentemente limpios y con columnas.

### 3.4.4. Creación de gráficos

Para crear un gráfico tanto en Excel como en Google Sheets, se debe seleccionar Insertar > Gráfico. Se puede personalizar cambiando los colores, dándole nombre a los ejes y variar el tipo de gráfico.

## 3.5. Importación de datos

Si se desea importar datos de otras hojas de cálculo:

Excel: Datos>Obtener datos> Desde archivo > Desde libro> seleccionar archivo > Importar> seleccionar en el navegador la hoja de trabajo que se quiere importar > Cargar o Transformar datos.

Google Sheets: =IMPORTRANGE(), el cual permite especificar un rango de celdas en la otra hoja de cálculo para duplicarlo en la hoja que se esté trabajando.

## 3.6. Errores

- #DIV/0!: Fórmula que intenta dividir por cero un valor en una celda o por una celda vacía. Se soluciona con: =SI.ERROR(valor; valor-si-hay-error).
- #ERROR!: Error que solo devuelve Google Sheets. Señala que la fórmula no se puede interpretar tal como se ingresa, es decir, hay un error en la fórmula.

- #N/A: Indica que la hoja de cálculo no puede encontrar los datos de la fórmula. Ocurren generalmente cuando se usa la función =BUSCARV().
- #NAME? o #NOMBRE?: ocurre cuando el nombre de una fórmula no se reconoce.
- #NUM!: Señala que el cálculo de una fórmula no se puede realizar según lo especificado por los datos, como por ejemplo una fecha negativa.
- #VALUE!: Señala un problema con la fórmula o con las celdas con las que hace referencia.
- #REF!: Aparece cuando las celdas de una fórmula se han eliminado.

# Capítulo 4

## Python

### 4.1. Notas previas...

1. Python es un lenguaje de programación orientado a objetos de alto nivel.
2. Los tipos de datos en Python son int (números enteros), float (decimales), string (cadenas de texto) y booleanos.
3. Los strings se escriben entre comillas simples o dobles.
4. Los comentarios se hacen con el numeral (#). Los comentarios que contengan más de una línea se hacen en un bloque de tres comillas simples:

```
1      # Esto es un comentario
2
3      '''
4      Esto
5      es un
6      comentario
7      '''
```

### 4.2. Librerías

En ciencia de datos, las librerías más usadas son NumPy, Matplotlib y Pandas.

#### 4.2.1. Importación de datos

Para importar las librerías, se utiliza el comando `import` y, usualmente, se le asigna un alias:

```
1      import libreria as alias
```

Si se quiere importar solo una función de una librería:

```
1 from libreria import funcion as alias
```

Ejemplo:

```
1 # Funcion que genera un numero aleatorio entre 0 y 10
2 import random
3 num = random.randint(0, 10)
4 print('El numero es ', num)
```

En Google Colaboratory, se importan los datos de drive con Pandas:

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

Luego, ir a Archivos > Drive > Copiar ruta de acceso. Se debe importar la librería Pandas con un alias:

```
1 import pandas as pd
```

Finalmente, para leer un archivo en formato csv:

```
1 nom-ds = pd.read_csv('ruta-del-archivo.csv')
```

## 4.3. Conceptos previos

### 4.3.1. Manipulación de strings

Los strings son cadenas de textos que contienen una secuencia de caracteres, como palabras. Se definen usando comillas simples o dobles y se pueden realizar operaciones con ellas.

Para interactuar con el usuario, se usan los comandos `print` y `input`. El primero imprime un mensaje en pantalla, mientras que el segundo permite que el usuario ingrese datos.

```
1 # Imprimir texto en pantalla
2 print('Hola mundo')
3
4 # Toma informacion y la puede guardar en una variable
5 nombre=input('Cual es tu nombre?')
6
7 # Sustituir variables en un string
8 print(f'Tu nombre es: {nombre}')
9
10 # Concatenacion
11 print('hola' + ' ' + 'mundo')
12 # salida: hola mundo
13
14 # Repeticion
```



```
15     print('Ja' * 4)
16     # salida: JaJaJaJa
```

Para transformar un número a string:

```
1     str(15)
2     # salida: 15
3
4     print('Quiero ' + str(15) + ' panes')
```

Y de un string a número:

```
1     edad = int(input('Cual es tu edad?'))
```

Operaciones con strings:

```
1     texto = 'Hola a todos'
2     # 1. lower(): convierte los caracteres a minusculas
3     texto_1 = texto.lower()
4     print(texto_1)
5     # salida: hola a todos
6
7     # 2. upper(): convierte los caracteres a mayusculas
8     texto_2 = texto.upper()
9     print(texto_2)
10    # salida: HOLA A TODOS
11
12    # 3. replace(sub, new_sub): reemplaza texto por uno
nuevo
13    texto_3 = texto.replace('todos', 'Marta')
14    print(texto_3)
15    # salida: Hola a Marta
16
17    # 4. split(sep): divide una cadena en una lista con un
separador
18    textonuevo = 'Hola, Josefa'
19    lista = textonuevo.split(',')
20    print(lista)
21    # salida: ['Hola', 'Josefa']
22
23    # 5. strip(): elimina los espacios en blanco al inicio y
final de una cadena
24    otrotexto = '    Hola    '
25    texto_4 = otrotexto.strip()
26    print(texto_4)
27    # salida: Hola
```

### 4.3.2. Estructuras de datos

Listas o arreglos: son secuencias de elementos ordenados por un índice, empezando por el 0.

```

1      # Crear una lista
2      lista_1 = ['elemento0', 'elemento1', 'elemento2']
3      print(lista_1) # Imprime la lista
4
5      # Para acceder a los elementos de la lista:
6      print(lista_1[0])
7      # salida: elemento0
8
9      print(lista_1[2]) # o print(lista_1[-1])
10     # salida: elemento2
11
12     # Para agregar un elemento al final de la lista
13     lista_1.append('elemento3')
14     print(lista_1)
15     # salida: ['elemento0', 'elemento1', 'elemento2', '
16     elemento3']
17
18     # Para obtener la longitud
19     longitud = len(lista_1)
20     print(longitud)
21     # salida: 4

```

Diccionarios: es una colección de par clave-valor. Su sintaxis es *{clave : valor}*:

```

1      diccionario_1={'nombre':'Marcelo',
2                    'apellido':'Soto',
3                    'ciudad':'Santiago',
4                    'edad':38,
5                    'profesion':'Presidente'}
6
7      # Imprimir un valor
8      print({diccionario_1['profesion']})
9      # salida: Presidente
10
11     # Agregar un nuevo par clave-valor
12     diccionario_1['estado civil'] = 'Soltero'

```

Tuplas: las tupla no se pueden modificar después de haber sido creada.

```

1      tupla_1=('elemento0', 'elemento1', 'elemento3')
2
3      # Para que devuelva 'elemento0'
4      tupla_1[0]

```

Set: no permite tener elementos repetidos. Guardará solo los datos que no se repiten.

```

1      set_1 = {'banana', 'manzana', 'pera'}

```

## 4.4. Limpieza, orden y transformación de datos

### 4.4.1. Bibliotecas

Las bibliotecas que más se usan en Python para la ciencia de datos son Pandas, NumPy, Scikit-learn y Matplotlib.

NumPy se especializa en el cálculo y análisis de datos, la cual permite manejar datos de forma rápida por medio de los NumPy Arrays o ndarray, una estructura de datos que puede ser de una dimensión (vector), dos dimensiones (matriz), tres (cubo) o más.

Por lo general, se importa con el alias np:

```
1 import numpy as np
```

Por otro lado, la biblioteca Pandas permite manipular y analizar estructuras de datos. Se basa en NumPy y proporciona los siguientes tipos de datos:

1. Series: Estructuras de una dimensión.
2. DataFrame: Estructura de dos dimensiones.
3. Panel: Estructura de tres dimensiones.

Se importa bajo el alias de pd:

```
1 import pandas as pd
```

Scikit-learn es una biblioteca de aprendizaje automático que proporciona acceso a algoritmos comunes.

Matplotlib se especializa en la generación de gráficos en dos dimensiones y personalización de estos. Se pueden crear diagramas de barras, mapas de calor, histogramas y más.

Por lo general, se importa con el alias plt:

```
1 import matplotlib.pyplot as plt
```

#### 4.4.2. Tipos de datos de Pandas

#### 4.4.3. Resúmenes de datos

#### 4.4.4. Manipulación de datos

Transformación de datos

Búsqueda de valores

Quitar duplicados, espacios en blanco...

Filtros

Orden

### 4.5. Análisis

### 4.6. Visualización

Para mostrar n número de filas del conjunto de datos:

```
1 nom-ds.head(n)
```

Nombres de las columnas del conjunto de datos:

```
1 nom-ds.columns
```

Renombrar columnas, primero se crea un diccionario y luego se utiliza el comando **rename**:

```
1 columnas= {'nom-col-1': 'nom-col-nueva-1', ...}  
2 nom-ds=nom-ds.rename(columns=columnas)
```

Para visualizar una muestra del dataset:

```
1 nom-ds.sample(num-de-registros)
```

Información del dataset:

```
1 nom-ds.info()
```

Localizar índice:

```
1 nom-ds.iloc[indice]
```

### 4.7. Análisis y visualización

#### 4.7.1. Cálculos matemáticos

Dimensión del conjunto de datos:

```
1 nom-ds.shape
```



# Capítulo 5

## R

### 5.1. Notas previas...

- 1.

### 5.2. Paquetes

Los paquetes en R incluyen funciones, documentación sobre estas mismas, muestras de conjuntos de datos y pruebas para verificar el código. R incluye un conjunto de paquetes denominados Base R.

Existe una colección de paquetes llamado Tidyverse que contiene los siguientes paquetes:

1. ggplot2: se usa para visualizar datos y crear visualizaciones.
2. tidyr: se usa para la limpieza de datos.
3. readr: se usa para importar datos.
4. dplyr: ofrece funciones que ayudan a la manipulación de datos.

Otros paquetes:

1. here
2. skimr: facilita el resumen de los datos.
3. janitor

Para ver los paquetes instalados:

`installed.packages()`

Para cargar paquetes:

```
library(paquete)
```

## 5.3. Limpieza

### 5.3.1. Resúmenes de datos

Funciones para obtener resúmenes de los marcos de datos:

```
#Muestra el tipo de datos  
str(df)  
#Devuelve el min, max, media, mediana, 1er y 3r cuartil. skimr  
summary(df)  
# skimr  
skim_without_charts(df)  
# Skimr  
skim(df)  
# Devuelve el numero de filas y columnas.  
glimpse(df)  
# Muestra solo las primeras seis filas.  
head(df)
```

## 5.4. Orden

```
# Se usa para elegir qué variable se quiere ordenar.  
arrange()  
# Para ordenar de forma descendente.  
arrange(-)  
# Agrupa.  
group_by()  
# Filtra los datos.  
filter()  
# Excluye valores NA.  
drop_na()
```



## 5.5. Transformación de datos

```
# Divide datos en columnas separadas.
separate(df, columna-a-separar, into=c('nom-col-nueva',
  ↪ 'nom-col-nueva-2'), sep= 'tipo-de-separador')
# Permite fusionar columnas entre sí.
unite(df, 'nom-col', col-a-combinar, col-a-combinar-2,
  ↪ sep='separador')
# Se puede usar para añadir columnas con cálculos.
mutate()
```

## 5.6. Cálculos matemáticos

```
# Media aritmética.
mean()
# Valor máximo.
max()
# Desviación estándar.
sd()
# Correlación.
cor(x=var-x, y=var-y, method='pearson o kendall o spearman')
```

## 5.7. Visualización

Se utiliza la función ggplot para crear gráficas:

```
ggplot((data=df) + geom_función(mapping=aes(x=var-x,
  ↪ y=var-y, color=var1, shape=var2, size=var3,
  ↪ alpha=var4))), (color= color-para-todos-los-puntos) +
  ↪ labs() + annotate(text)
```

Conceptos de la función ggplot:

```
# Se inicia un diagrama y se le pueden agregar capas con +.
ggplot()
# Dataset.
data
# Usa puntos para crear diagramas (figura geométrica).
geom_función()
# Define cómo se aplican las variables a las propiedades
  ↪ visuales.
```

```

mapping # Va junto a la función aes()
# Especifica qué variables aplicar a los ejes x e y, como el
  ↪ color (color),
# color de relleno (fill), forma de puntos (shape), tipo de
  ↪ línea (linetype)
# y tamaño (size).
aes()
# Grado de transparencia del color.
alpha()
# Se pueden introducir etiquetas como title="", subtitle="",
  ↪ caption="".
labs
# Se utiliza para agregar texto dentro de la cuadrícula.
annotate(text)
# Más información en
  ↪ https://ggplot2.tidyverse.org/reference/annotate.html

```

Funciones geom:

```

# Crea diagramas de dispersion.
geom_point()
# Crea graficos de barra.
geom_bar()
# Gráfico de líneas.
geom_line()
# Suavizado. Crea una línea de tendencia.
geom_smooth()
# Crea un histograma.
geom_histogram()
# Crea un diagrama de dispersión y agrega una pequeña cantidad de ruido
# aleatorio para lidiar con la superposición de puntos.
geom_jitter()

```

Cuando se usa la función `geom_bar`, R cuenta automáticamente cuantas veces aparece cada valor `x` en los datos y muestra los recuentos en el eje `y`. Por esta razón, se puede poner en el código solo el eje `x`. Se puede usar `fill` para llenar de color las barras.

En el caso de `geom_histogram`, los argumentos de esta función son `bins=numero`, el cual hace referencia al número de intervalos y `binwidth=numero` es la amplitud de los intervalos.

### 5.7.1. Suavizado

El suavizado permite detectar una tendencia de datos aun cuando no se pueda notar con facilidad una tendencia en los puntos de datos graficados. En ggplot2, suma una línea de suavizado como otra capa en un diagrama. Ayuda a que los diagramas sean más legibles.

1. El suavizado LOESS es óptimo para suavizar diagramas con menos de 1000 puntos.

```
ggplot(data, aes(x=, y=))+ geom_point() +  
  ↪ geom_smooth(method="loess")
```

2. El suavizado GAM es útil para suavizar diagramas con un gran numero de puntos.

```
ggplot(data, aes(x=, y=))+ geom_point() +  
  ↪ geom_smooth(method="gam", formula = y ~ s(x))
```

### 5.7.2. Estética y facetas

Una faceta es una cara o sección de un objeto. Sirve para comparar datos. Permiten mostrar grupos más pequeños, o subconjuntos, de datos.

```
# Facetar el diagrama con una variable.  
facet_wrap(~variable)  
# Facetar el diagrama con dos variables.  
facet_grid(var1~var2)
```

### 5.7.3. Guardar gráficas

Para guardar una gráfica se puede exportar desde RStudio o usar la función:

```
ggsave(\nombre-gráfico.tipo-archivo", width=ancho,  
  ↪ height=largo) # del paquete ggplot2. Ejemplo:  
ggsave(\Pingüinos.png")
```



# Capítulo 6

## Markdown

Markdown es un lenguaje usado en documentos donde existen dos tipos de celdas: una de texto y otra de código. Las celdas de texto están formateadas con este lenguaje, lo que lo hace fácil y simple de escribir.

Tanto Colab como R permiten insertar ecuaciones usando la notación de LaTeX y personalizar la escritura con código HTML.

Se le puede dar formato a la escritura con los siguientes comandos:

1. *Cursiva*: `*texto*`
2. **Negrita**: `**texto**`
3. Tachado: `texto`
4. [Enlaces](#): `[Clic aquí](aquíVaElEnlace)`
5. Monoespaciado: `'texto'`
6. Imagen: `![Una imagen](linkDeLaImagen)`

Los encabezados en Markdown se inician con un numeral. Mientras más numerales tenga, más pequeño es el encabezado.

# Capítulo 1: **Capítulo 1:**

## Sección 1: **Sección 1:**

### Subsección: **Subsección:**

Se pueden insertar bloques de código en las celdas de texto usando las comillas

“.

“python

import pandas as pd “

De esta forma, el código en la celda tendrá un formato especial:

```
1 import pandas as pd
```

Se pueden crear listas ordenadas:

1. Propiedad 1
2. Propiedad 2
3. Propiedad 3

Y listas desordenadas con guiones (-) o asteriscos (\*):

- Propiedad 1
- Propiedad 2
- Propiedad 3

Es posible insertar líneas horizontales con tres asteriscos (\*\*\*) o tres guiones (—).

# Capítulo 7

## Git y GitHub

### 7.1. Git

Se puede hacer correr desde la cmd con el comando `git`.

Lista de comandos:

- `git config --global user.name "nom-usuario"`: se configura el nombre de usuario que va entre las comillas. Para configurar el email se debe poner `git config --global user.email` seguido del email entre comillas.
- `git config user.name`: para ver el nombre de usuario. Para ver el email es `git config user.email`

#### 7.1.1. Repositorios

Para abrir Git en el proyecto, se debe hacer clic derecho Git Bash Here en la carpeta donde se ubica dicho proyecto. Una vez abierto, se puede crear un nuevo repositorio con `git init` en la rama Master o main.

Con el comando `git add nom-archivo.extension` se puede enviar archivos al repositorio. Si se quieren subir todos los archivos, se debe usar el comando `git add ..`

Una vez que se hayan realizado todos los cambios en los archivos, se le puede añadir un comentario con `git commit -m "Mensaje"` y se suben con `git push origin nom-rama`.

Si los archivos fueron modificados en GitHub, se pueden bajar al computador con `git pull`.

Lista de comandos:

1. `git status`: ver el estado del proyecto.

2. `git commit -m "mensaje"`: se utiliza para tomar una instantánea del proyecto y dejar un mensaje con los cambios realizados en este.
3. `git log`: para ver todos los mensajes y cambios que han realizado los autores.
4. `git log --stat`: sirve para visualizar los commits y muestra un pequeño resumen de lo modificado, como la cantidad de líneas e inserciones.
5. `git log --oneline`: muestra un código que identifica al commit.
6. `git checkout codigo-commit`: actualiza el archivo para que coincida con el commit señalado en el código-commit. Una vez que se vuelve a un commit anterior, los otros que están en medio se borran. Por esta razón se recomienda trabajar con ramas.
7. `git rm --cached nom-archivo.extension`: deja de rastrear un archivo sin eliminarlo del directorio de trabajo. Si se quiere hacer con todos los archivos: `git rm --cached ..`
8. `git clone direccion-del-repositorio`: clonar un repositorio en el computador.
9. `touch nom_archivo.extension`: crea un archivo.
10. `clear`: limpiar la pantalla.

### 7.1.2. .gitignore

El archivo `.gitignore` se crea con fines de que Git ignore ciertos archivos. Para crearlo, se puede hacer mediante la consola de Git con `touch .gitignore`. Luego de crearlo, se puede abrir con cualquier aplicación y escribir los archivos que se desean ignorar. Se deben escribir los nombres de los proyectos, como `mi_proyecto.txt`, o se pueden ignorar todas las extensiones, como `*.txt`.

### 7.1.3. Ramas

Una rama en Git es una versión del código de un proyecto, las cuales ayudan a mantener el orden en el control de versiones. El control de versiones es la práctica de gestionar los cambios que se realizan sobre un código. Cuando se habla del estado en el que se encuentra, se refiere a la versión, revisión o edición de este. Cada repositorio en Git comienza con una rama principal predeterminada llamada `master`.



Git almacena una serie de instantáneas al ejecutar el comando commit junto con los metadatos de quién lo haya modificado.

Comandos:

- `git branch`: devuelve la cantidad de ramas que tiene un proyecto.
- `git checkout rama`: cambia de rama.
- `git checkout -b nombre-rama`: crear una rama.
- `git merge nombre-rama`: unir ramas. El código de nombre-rama se une al de la rama en la que se está trabajando.
- `git branch -d nombre-rama`: eliminar una rama solo si la rama se ha fusionado. De lo contrario, usar `-D` que elimina la rama independientemente de su estado de fusión.
- `git branch -d nombre-remoto/nombre-rama`: eliminar una rama remota de VSCode. Generalmente, `nombre-remoto` es `origin`.



# Capítulo 8

## Tableau Public

Tableau Public es una plataforma para crear y compartir visualizaciones de datos en línea.

Una visualización de datos es una representación gráfica de los mismos para comunicar información de manera efectiva. Se usa principalmente para identificar patrones que pueden ser difíciles de detectar en una tabla de datos, explorar datos, comunicar información, ayudar a tomar decisiones informadas, hacer seguimiento a procesos, identificar problemas de procesos y más.

### 8.1. Creación de visualizaciones

#### 8.1.1. Importación de datos y conexiones

Para importar datos, se pueden cargar archivos desde el equipo o conectarse con Google Drive. Si se desean agregar más de un archivo, se debe seleccionar el signo + al lado de Conexiones.

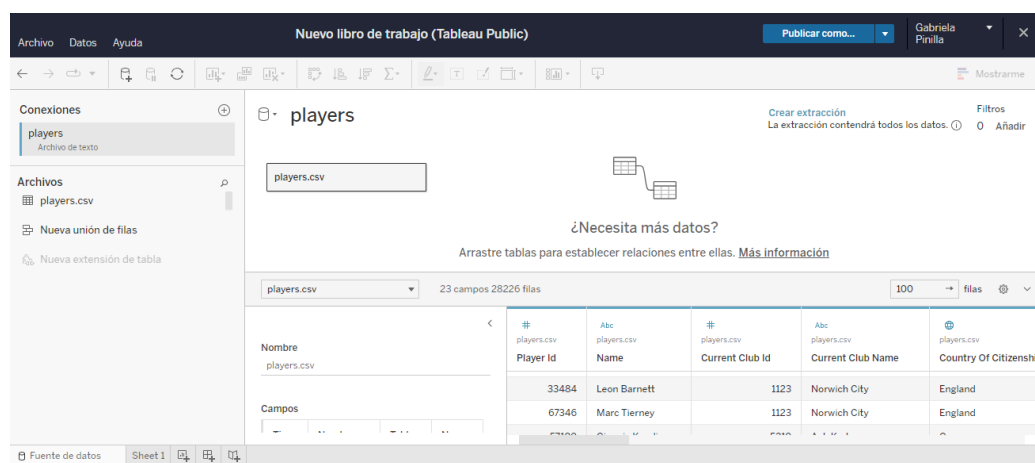


Figura 8.1: Fuente de datos

En la esquina inferior derecha se muestran los datos en una tabla. Hacia la izquierda en Conexiones, se muestran las fuentes de los datos, los cuales se pueden conectar unos con otros arrastrándolos hacia el panel que dice ¿Necesita más datos?, quedando de la siguiente forma:

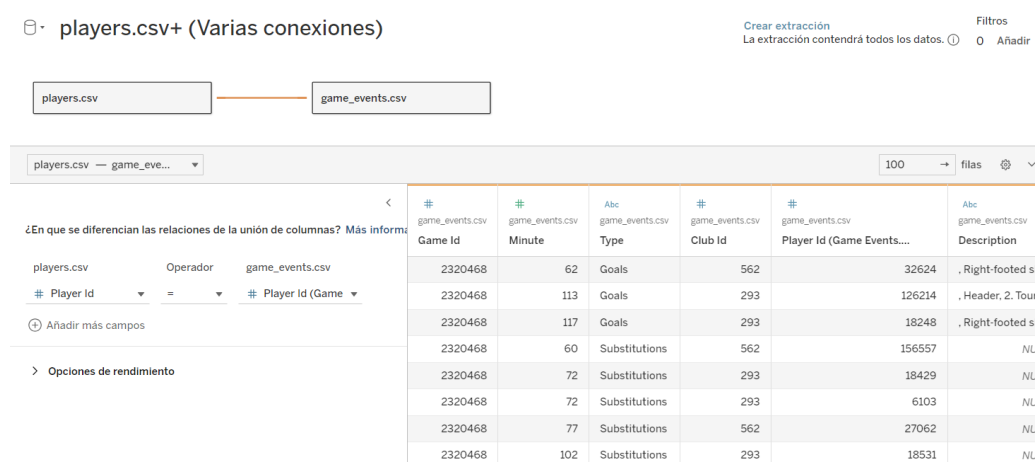


Figura 8.2: Campos en común

Tableau detecta si hay un campo en común entre ambos archivos csv la cual muestra bajo la conexión de las fuentes.

### 8.1.2. Campos calculados

Desde la misma hoja de Fuentes de datos se pueden realizar cálculos en las tablas. Para hacerlo, se debe seleccionar la tabla, luego la flecha de la columna en

la que se desea crear un campo calculado, > Crear > Campo calculado...

Se le puede asignar un nombre al campo calculado y escribir la fórmula para calcular los valores del campo seleccionado con los nombres de los campos entre paréntesis de corchetes. Ejemplo: [Campo 1]\*5.

En Tableau existen funciones básicas que se pueden utilizar para las fórmulas de los campos calculados:

1. CONTAINS: Retorna verdadero si una cadena contiene una subcadena. Ejemplo: CONTAINS("Hola a todos", "todos"), devolverá verdadero.
2. LTRIM: Elimina los espacios en blanco del lado izquierdo de una cadena. Ejemplo: LTRIM("Hola"), devuelve "Hola".
3. TRIM: Elimina los espacios en blanco de ambos lados de una cadena. Ejemplo: TRIM("Hola "), retorna "Hola"
4. SPLIT(): Divide un campo según un separador especificado y un token. Ejemplo: SPLIT([Nombre completo], ' ', 2), en el caso de que el nombre sea Juan Soto, devolverá Soto. Si el token es 1, devolverá Juan.
5. LEN: Devuelve el largo de una cadena de texto. Ejemplo: LEN("Hola a todos"), devuelve 12.
6. LEFT: Devuelve los primeros caracteres de una cadena según una cantidad especificada. Ejemplo: LEFT("Hola a todos", 4), devuelve "Hola".
7. RIGHT: Devuelve los caracteres iniciando desde la derecha hacia la izquierda. Ejemplo: RIGHT("Hola a todos", 5), devuelve "todos".
8. LOWER: Convierte el texto en minúsculas. Ejemplo: LOWER("HOLA"), devuelve hola.
9. UPPER: Convierte el texto en mayúsculas. Ejemplo: UPPER("hola"), devuelve "HOLA".
10. MAX: Retorna el valor máximo de una expresión. Ejemplo: MAX([Ventas]), retornará el valor máximo de la columna Ventas.
11. REPLACE: Reemplaza una subcadena de texto por otra. Ejemplo: REPLACE("Hola a todos", "todos", "todas las mujeres presentes aquí"), devuelve "Hola a todas las mujeres presentes aquí".
12. COUNT: Muestra la cantidad de filas de un campo. Ejemplo: COUNT([Libros]), devolverá la cantidad de libros.

13. CEILING: Devuelve el número entero más pequeño mayor o igual que un número especificado. Ejemplo: CEILING(5,13), devuelve 5.
14. FLOOR: Devuelve el número entero más grande menor o igual que un número especificado. Ejemplo: FLOOR(5,16), devuelve 3.
15. ROUND: Redondea un número con decimales especificados. Ejemplo: ROUND("5,1634546", 2), devuelve 5,16.
16. LOG: Devuelve el logaritmo en base e de un número.
17. DATE: Devuelve la fecha. Ejemplo: DATE(01/05/2023 00:00:00), devuelve 01/05/2023.
18. DATEDIFF: Devuelve la diferencia entre dos fechas. Ejemplo: DATE-DIFF(01/09/2023, 10/09/2023, DAY), devolverá la cantidad de días entre ambas fechas.
19. DATENAME: Devuelve el nombre de un mes, día de la semana o del año.
20. DATEPART: Devuelve una parte específica de una fecha. Ejemplo: DATE-PART(01/09/2023, YEAR), devolverá 2023.

### 8.1.3. Conceptos y elementos de Tableau

Al momento de crear un dashboard, se debe tener en consideración ciertos términos y elementos de la plataforma.

Una hoja de trabajo es un espacio donde se puede construir la visualización de datos. Sus elementos principales son:

- Filas y columnas: aquí es donde el usuario puede arrastrar los campos de datos para definir las dimensiones y medidas de la visualización.
- Dimensiones y medidas: en el panel de datos se pueden encontrar los datos divididos por una línea donde los de arriba representan las dimensiones y los de abajo las medidas.
  - Las dimensiones son los datos categóricos. Su fin es clasificar la información. Los tipos de dimensiones son: fechas, cadenas de texto, booleanos (verdadero o falso) y rol geográfico. Ejemplo: género, estado civil.
  - Las medidas son datos de origen cuantitativos que asignan valores a las dimensiones. Los tipos de medidas son: números y rol geográfico (latitud y longitud). Ejemplo: cantidad de habitantes.

En este mismo panel, se encuentran datos de color azules, los cuales corresponden a datos discretos. Se tratan como finitos. Por el contrario, los verdes son datos continuos. Se tratan como un intervalo infinito.

- **Marcas:** en la tarjeta de marcas se configura la forma en que las dimensiones y medidas serán representadas. Sus elementos son: Tipo de marca: el cual indica la forma en la que se van a graficar los datos; Color: permite cambiar los colores de la visualización; Tamaño: varía el tamaño de la visualización; Texto: permite definir el texto de la etiqueta a mostrar en la gráfica; Detalle: permite agregar información a las marcas; Descripción emergente: permite que al pasar el mouse por una marca se muestre una descripción emergente con información.

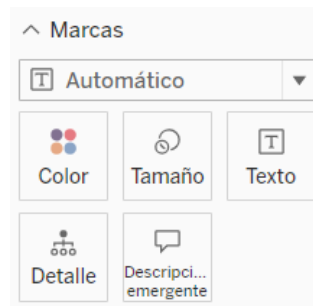


Figura 8.3: Tarjeta de marcas

- **Filtros:** aplica filtros al arrastrar los campos a este estante.
- **Páginas:** se pueden arrastrar campos para crear páginas en la visualización.

#### 8.1.4. Filtros

1. **Filtros de extracción:** se usan para limitar el conjunto de datos desde la Fuente de datos. Al aplicarlo, se abrirá una ventana para añadir los filtros que se deseen.

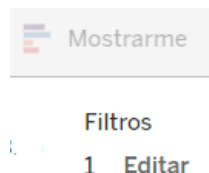


Figura 8.4: Filtro de extracción

2. Filtros de contexto: se usan para definir un contexto específico en los cálculos. En lugar de aplicar un filtro para el conjunto de datos, crea un subconjunto de datos para hacer cálculos. Para crear un filtro de contexto, se debe arrastrar un campo al estante de filtros en la hoja de trabajo, luego seleccionar la flecha de la cápsula del campo y seleccionar Añadir a contexto.

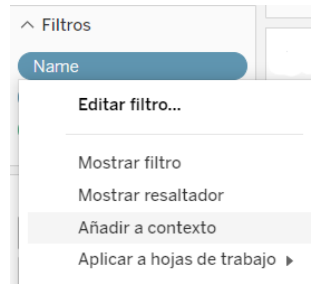


Figura 8.5: Filtro de contexto

3. Filtro de dimensión: se usan para filtrar datos de dimensiones discretas.

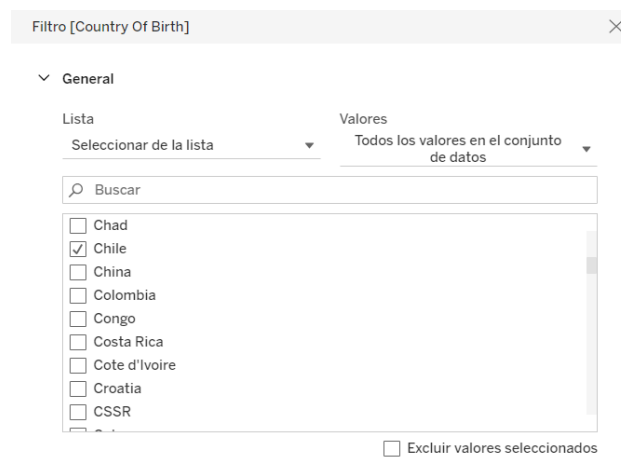


Figura 8.6: Filtro de dimensión

4. Filtro de medida: se usan para filtrar datos en función de una medida continua.





Figura 8.7: Filtro de medida

5. Filtro en gráfico: se pueden filtrar datos en un gráfico.

## 8.2. Creación de visualizaciones básicas

La creación de visualizaciones en Tableau se hace arrastrando las tablas de datos hacia los estantes de filas, columnas, marcas y filtros.

En el siguiente ejemplo se utilizó la base de datos del siguiente enlace de Kaggle [Uhttps://www.kaggle.com/datasets/ishikajohari/shazam-global-top-200-per-week?resource=download](https://www.kaggle.com/datasets/ishikajohari/shazam-global-top-200-per-week?resource=download), el cual muestra datos del top 200 por semana de Shazam, una aplicación para identificar música y otros desde los dispositivos móviles.

Con la base de datos de la primera semana (Week1 -22-Jul-to-28-Jul-2023.csv), se construyó la siguiente tabla:

The screenshot shows the Tableau Public interface. On the left, the 'Datos' pane lists data sources: 'Week1 - 22-Jul-to-28-Ju...'. The 'Análisis' pane shows a search bar and a list of tables: 'Artist', 'Title', 'Nombres de medidas', 'Rank', 'Week1 - 22-Jul-to-28-Ju...', and 'Valores de medidas'. The 'Filtros' pane is empty. The 'Marcas' pane shows the 'Automático' mark type selected, with 'Color', 'Tamaño', and 'Texto' options. The 'SUMA(Rank)' function is selected in the 'Texto' section. The main view shows 'Sheet 1' with a table of data.

Artist	Title	Rank
A.V.G	Я плачу	56
Aaron Smith	Dancin (feat. Luvli) [Krono...	118
Adele	Set Fire to the Rain	45
Aerosmith	Dream On	98
Akon	Lonely	194
	Right Now (Na Na Na)	84
Alec Benjamin	Let Me Down Slowly	117
Aqua	Barbie Girl	120
Asake	Lonely At The Top	33
AYLIVA	Aber sie	170
Ayra Starr	Rush	42
Bad Bunny	WHERE SHE GOES	60
Bad Gyal, Tokíscha & YOUN...	Chulo pt. 2 (Pt. 2)	167
Bakar	Hell N Back	198
Beastie Boys	No Sleep Till Brooklyn	105
Becky G.	Arranca (feat. Omega)	173
Ben E. King	Stand By Me	146
Benny Benassi & The Biz	Satisfaction (Radio Edit)	100
Bibi Babydoll & Dj Brunin ..	Automotivo Bibi Fogosa	5
Billie Eilish	headline (edit)	104

Figura 8.8: Tabla de datos creada en Tableau

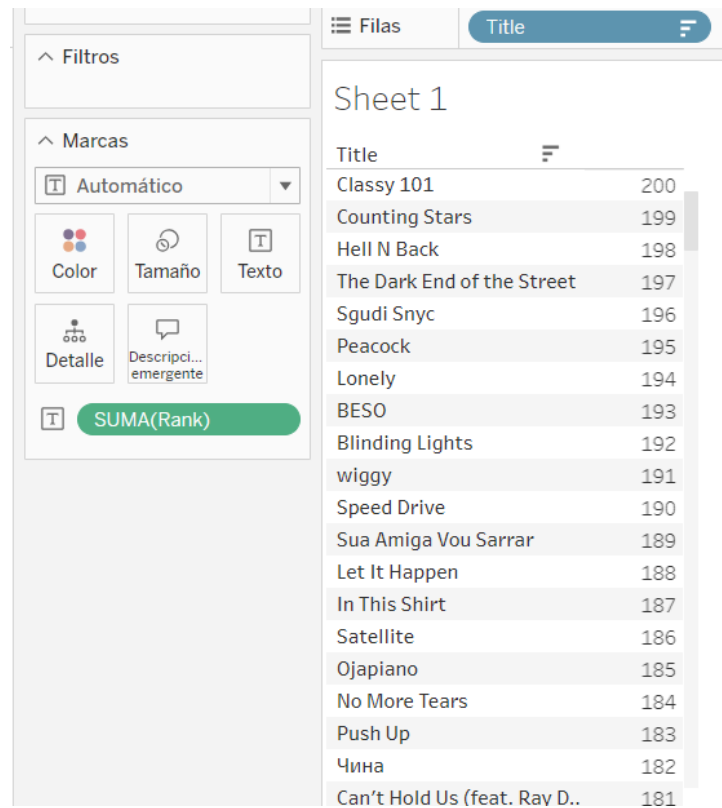
En el estante de filas se añadieron Artist (dimensión) y Title (dimensión), mientras que Rank (medida) se añadió a Texto en Marcas, mostrando en la tabla el rankin de la canción y al artista a quien le pertenece. Por defecto, Tableau añade una función de agregación, que en el caso del ejemplo es SUMA, sin embargo, como se trata de un rankin, solo corresponde a un número y no a una suma.

En la misma tabla, se pueden ordenar los datos por el rankin, como de mayor a menor, sin embargo, al tener dos dimensiones en las filas no mostrará cuál canción es menos escuchada en el rankin, sino que mostrará un conjunto de canciones con su rankin perteneciente al mismo artista.

Artist	Title	
Travis Scott	FE!N (feat. Playboi Carti)	131
	FRANCHISE (feat. Young T..	133
	I KNOW ?	88
	MELTDOWN (feat. Drake)	39
	MY EYES	83
	TELEKINESIS (feat. SZA & ..	29
OneRepublic	Counting Stars	199
	I Ain't Worried	172
	RUNAWAY	123
Lana Del Rey	Radio	138
	Say Yes To Heaven	180
	Summertime Sadness	153
The Weeknd	After Hours	99
	Blinding Lights	192
	Save Your Tears	127
Taylor Swift	cardigan	169
	Cruel Summer	64
	seven	161

Figura 8.9: Tabla de datos creada en Tableau

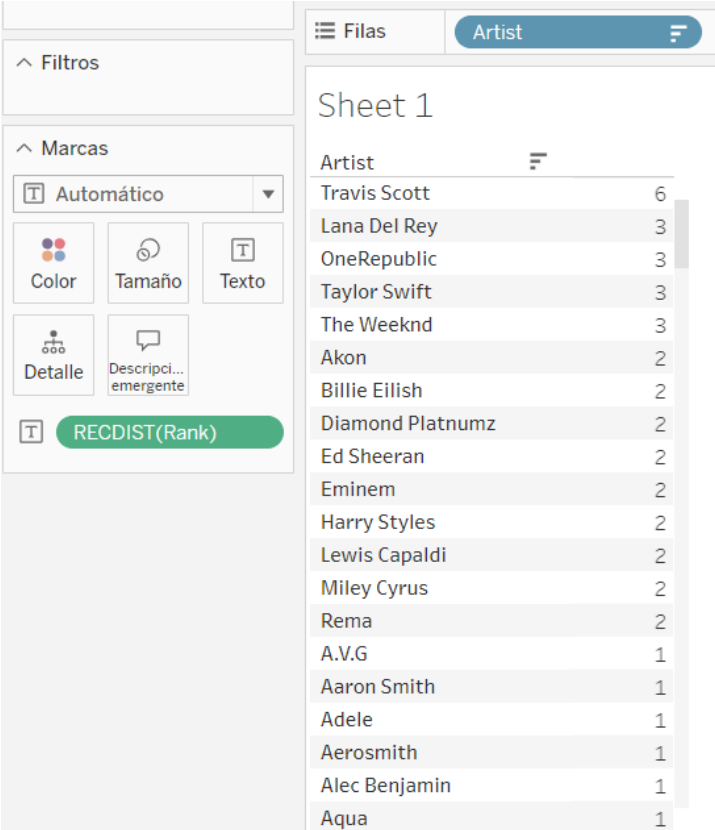
Para descubrir cuál es la canción menos escuchada, se debe dejar solo una dimensión en filas, como Title, y luego ordenarlas. Si se quiere saber cuál es el artista menos escuchado, se debe hacer lo mismo con la dimensión Artist.



Title	
Classy 101	200
Counting Stars	199
Hell N Back	198
The Dark End of the Street	197
Sgudi Snyc	196
Peacock	195
Lonely	194
BESO	193
Blinding Lights	192
wiggy	191
Speed Drive	190
Sua Amiga Vou Sarrar	189
Let It Happen	188
In This Shirt	187
Satellite	186
Ojapiano	185
No More Tears	184
Push Up	183
Чина	182
Can't Hold Us (feat. Ray D..	181

Figura 8.10: Tabla de datos creada en Tableau

También, se puede modificar la medida Rank para que cuente la cantidad de canciones con la que los artistas aparecen en el rankin.



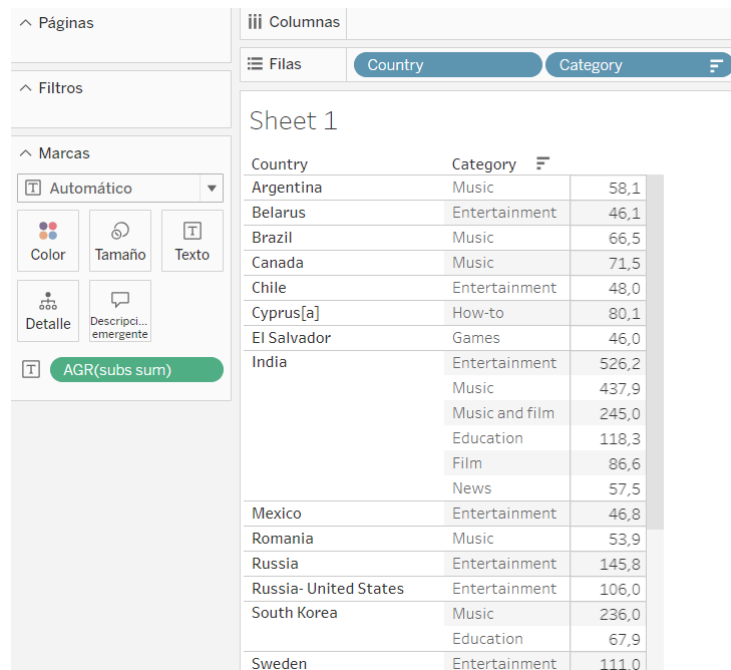
The screenshot shows the Tableau interface. On the left, the 'Marcas' (Marks) shelf contains a green pill labeled 'RECDIST(Rank)'. The main view is a table titled 'Sheet 1' with the column 'Artist' selected. The table lists 20 artists and their corresponding values in the 'RECDIST(Rank)' field.

Artist	RECDIST(Rank)
Travis Scott	6
Lana Del Rey	3
OneRepublic	3
Taylor Swift	3
The Weeknd	3
Akon	2
Billie Eilish	2
Diamond Platnumz	2
Ed Sheeran	2
Eminem	2
Harry Styles	2
Lewis Capaldi	2
Miley Cyrus	2
Rema	2
A.V.G	1
Aaron Smith	1
Adele	1
Aerosmith	1
Alec Benjamin	1
Aqua	1

Figura 8.11: Tabla de datos creada en Tableau

Para el siguiente ejemplo, se tomó la base de datos mensual de las 50 principales cuentas de redes sociales de <https://www.kaggle.com/datasets/amyrmahdy/monthly-top-50-social-media-accounts-dataset>.

Con la base de datos del top 50 de Youtube (youtube\_top\_50\_2023-07-03), se construyó la siguiente visualización:



Country	Category	AGR(subs sum)
Argentina	Music	58,1
Belarus	Entertainment	46,1
Brazil	Music	66,5
Canada	Music	71,5
Chile	Entertainment	48,0
Cyprus[a]	How-to	80,1
El Salvador	Games	46,0
India	Entertainment	526,2
	Music	437,9
	Music and film	245,0
	Education	118,3
	Film	86,6
	News	57,5
Mexico	Entertainment	46,8
Romania	Music	53,9
Russia	Entertainment	145,8
Russia- United States	Entertainment	106,0
South Korea	Music	236,0
	Education	67,9
Sweden	Entertainment	111,0

Figura 8.12: Tabla de datos creada en Tableau

Esta tabla muestra la categoría (Category) de cada país (Country) ordenado de mayor a menor por la suma de los subscriptores (subs sum) de cada categoría. Esta suma se realizó en el campo calculado de la Fuente de datos con el código SUM([Subscribers (millions)]).

Sin embargo, estos datos también se pueden representar de forma geoespacial por medio de un mapa.

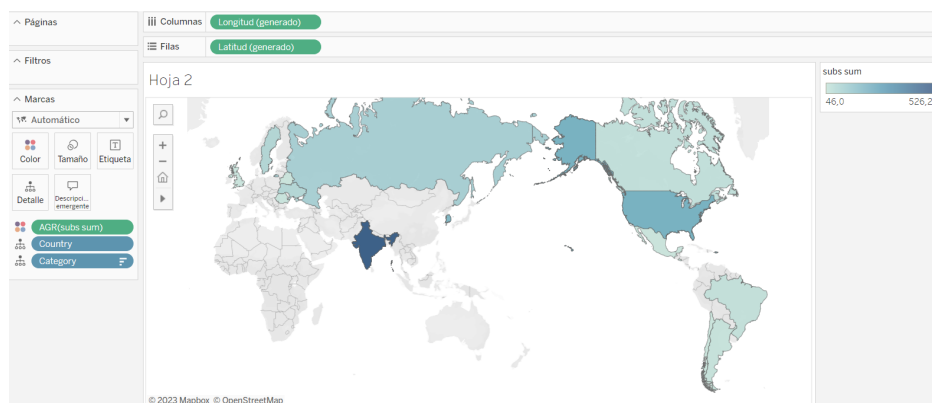


Figura 8.13:

Ordenando los subscriptores de forma descendente, se obtiene un mapa que

muestra la categoría con más subscriptores en los países, donde el color azul indica un número más alto que el color celeste claro.

Para hacer esto, se agregó la tabla de subscriptores a la marca de Color, de esta forma Tableau puede hacer un mapa de calor indicando qué países tienen una mayor cantidad de subscriptores. La categoría y el país se arrastraron hacia la marca de Detalles, por lo que al pasar el mouse por encima de alguno de estos países mostrará el nombre del país, la categoría y la cantidad de subscriptores. La latitud y longitud en filas y columnas se generó a partir de la tabla Country.

### **8.3. Diseño y personalización de paneles**

Al graficar, se debe tener en consideración el tipo de datos que presenta la base de datos dado que no todos los gráficos sirven para representar cualquier tipo de datos.

Para elegir un gráfico que va a representar un conjunto de datos, es importante saber qué es lo que se quiere mostrar con la visualización, conocer los tipos de gráficos y saber cuál es la audiencia a quienes se les va a presentar.

A continuación, se presentan algunos tipos de gráficos:

1. Gráfico de columnas: Compara categorías usando barras verticales.
2. Gráfico de barras: Compara categorías usando barras horizontales.
3. Gráfico circular: Proporciona las categorías en relación al total.
4. Gráfico de líneas: Muestra las tendencias a lo largo del tiempo usando líneas.
5. Gráfico de área: Muestra las tendencias a lo largo del tiempo usando un área sombreada bajo la línea.
6. Gráfico de dispersión: Relaciona dos conjuntos de datos usando puntos.
7. Gráfico de burbujas: Relaciona dos conjuntos de datos usando burbujas de diferentes tamaños para mostrar la magnitud de un tercer conjunto de datos.
8. Histograma: Distribuye los datos en un rango continuo.
9. Gráfico de radar: Expone los datos en un formato circular con categorías que se extienden radialmente.

### 8.3.1. Funciones de agregación

Las funciones de agregación son una funciones que operan sobre un conjunto de datos que devuelven un único valor. Se pueden obtener el promedio de un conjunto, máximo, mínimo, recuento o suma.

Las funciones de agregación más comunes en Tableau son:

- Promedio
- Suma
- Promedio o media
- Mediana
- Recuento: devuelve la cantidad de filas. Ejemplo:  $\text{Rec}(a,a,b,b)=4$ .
- Recuento (distintos): devuelve la cantidad de valores distintos de una columna. Ejemplo:  $\text{RecDis}(a,a,b,b)=2$ .
- Máximo y mínimo
- Desviación estándar y desviación estándar poblacional
- Varianza y varianza poblacional

Para agregar una función de agregación, se debe seleccionar Medida > Suma, Promedio, Mediana...

Por defecto, Tableau agrega la suma para medidas numéricas y el recuento para las no numéricas.

### 8.3.2. Funciones LOD

Las funciones LOD (Level of Detail) en Tableau permite realizar cálculos en un nivel de detalle específico en los datos. Son útiles cuando se requiere realizar cálculos que no se pueden lograr con las funciones de agregación.

Tableau tiene tres tipos de funciones LOD:

1. **FIXED**: permite definir un nivel de detalle específico para un cálculo. Ejemplo: Para calcular la suma de ventas por categoría: `FIXED [Categoría] : SUM([Ventas])`.
2. **INCLUDE**: permite incluir un nivel de detalle específico a un cálculo sin afectar el nivel de detalle general de la visualización. Ejemplo: Para calcular la suma de ventas por categorías e incluir la ciudad: `INCLUDE [Categoría],[Ciudad] : SUM([Ventas])`.



3. EXCLUDE: permite excluir un nivel de detalle específico a un cálculo sin afectar el nivel de detalle general de la visualización.

## 8.4. Dashboards

Un dashboard consiste en una pantalla que presenta una colección de visualizaciones que resumen información importante.

Uno de los propósitos de un dashboard es monitorear el rendimiento de una empresa por medio de KPI, presentando la información de forma clara para que el público pueda entenderla.

Un KPI (Key Performance Indicator o Indicador Clave de Rendimiento) es una medida que evalúa el rendimiento de una empresa o proceso con respecto al objetivo empresarial. Los KPI siguen la metodología S.M.A.R.T., un acrónimo que explica las características básicas de un objetivo SMART. Estos deben ser específicos (Specific), medibles (Measurable), alcanzables (Attainable), realistas (Realistic) y con un tiempo límite (Time).

Ejemplos de métricas que se pueden usar para los KPI:

- ROI
- Número de clientes nuevos
- Tasa de satisfacción
- Tasa de rotación del personal
- Porcentaje de clics en una página web
- Cantidad de carritos de compras abandonados
- IPC

### 8.4.1. Creación de un dashboard

Para crear un dashboard, se debe seleccionar el icono de Nuevo dashboard. Luego, arrastrar las vistas desde la lista de Hojas hacia el dashboard.



## Capítulo 9

# Glosario de términos de programación

La siguiente lista muestra las traducciones al español de los términos de programación en inglés más usados.

1. Array: Arreglo, listas.
2. Dataset: Conjunto de datos.
3. Debugging: Depuración.
4. Default: Predeterminado.
5. Delete: Borrar.
6. Drop: Borrar.
7. Loop: Bucle.
8. Return: Retornar.
9. String: Cadena.
10. Database: Base de datos.
11. Null: Nulo.
12. Fork: Bifurcación, clonación de un programa.



# Bibliografía

- [1] <https://www.javatpoint.com/postgresql-tutorial>
- [2] <https://git-scm.com/doc>
- [3] <https://git-scm.com/book/es/v2>
- [4] [https://help.tableau.com/current/pro/desktop/es-es/dashboards\\_create.htm](https://help.tableau.com/current/pro/desktop/es-es/dashboards_create.htm)
- [5] <https://docs.python.org/es/3/index.html>