Desafío 2. SQL
Recursos: SQLite

Perfil Junior
Dataset: Ciclovías

1. Crear una sentencia para consultar los 10 principales barrios ordenados por largo de la calle (campo long).

```
SELECT *
FROM  ciclovias
ORDER BY long desc
   limit 10
```

2. Crear una sentencia para insertar la siguiente línea: MULTILINESTRING((-8.429218163232 -34.5508222858,-58.42963323879 -34.5505234549563)), 899, 16033, OBLIGADO RAFAEL, Av.Costanera, 6182, 6260, 6181, 6199 ,COSTANERA SUR, AV.COSTANERA RAFAEL OBLIGADO, AVENIDA ,52.38, DOBLE, 2 ,No Aplica, Ciclovías ,Mano derecha, Vereda, Construcción Año 2014, Ciclovía ,VÍA DISTRIBUIDORA PRINCIPAL ,SI, 13, 13, 13, PALERMO S, PALERMO S, PALERMO S

```
INSERT INTO ciclovias
VALUES ('MULTILINESTRING((-58.429218163232 -34.5508222858,-8.42963323879 -34.5505234549563))', 899, 16033, 'OBLIGADO RAFAEL, Av.Costanera', 6182, 6260, 6181, 6199,'COSTANERA SUR', 'AV.COSTANERA RAFAEL OBLIGADO', 'AVENIDA', 52.38, 'DOBLE', 2, 'No Aplica', 'Ciclovías','Mano derecha','Vereda', 'Construcción Año 2014', 'Ciclovía', 'VÍA DISTRIBUIDORA PRINCIPAL', 'SI', 13, 13, 13,'PALERMO S', 'PALERMO S', 'PALERMO S')
```

3. Agrupar los barrios por el campo “COMUNA”.

```
SELECT BARRIO
FROM ciclovias
GROUP BY COMUNA
```

4. Crear una sentencia para modificar el nombre del campo CICLO_OBSE por CICLO_OBSERVACION.

```
ALTER TABLE ciclovias
RENAME COLUMN ciclo_obse TO ciclo_observacion
```

Perfil SemiSenior
Dataset: (Ciclovías; Bocas de Subte)

1. Si tuvieras que crear una sola tabla con ambos datasets, ¿qué campos te parecen interesantes y porque?
   Los campos interesantes de cada dataset son relacionados a continuación:
   Ciclovias (id, CODIGO, NOMOFICIAL, NOMANTER, NOM_MAPA, SENTIDO, RED_TP, COMUNA, BARRIO)
   Bocas de Subte (id, línea, estación, destino_bo, lineas_de_, cierra_fin, calle, calle2, barrio, comuna, dom_norma)
   Dichas variables servirían luego para tirar *insight* y ayudar al usuario a conectar los viajes en metro y bicicletas como transporte alternativo. Incluso podrían sugerirse mejores rutas para llegar más rápido al destino o de manera más segura. Esto ayudaría a que más personas optaran por usar las bicicletas.

2. Basándonos en el punto anterior, ¿Cómo armarías la tabla?. Colocar la sentencia y la tabla final.
   Para armar la tabla usaría el campo comuna para hacer el join, debido a que una ciclovía puede tener varias bocas pero una boca puede estar en una única ciclovía creamos un relacionamiento de ciclovias para boca (1:n) y hacemos uso de SELECT DISTINCT para no repetir los valores.

```
CREATE TABLE Conexion AS
SELECT DISTINCT id, codigo, nomoficial, nom_mapa, sentido, red_tp, barrio,
id_boca, linea, estacion, destino_bo, lineas_de_, cierra_fin, calle,
calle2, barrio_boca, comuna, dom_norma
FROM ciclovias
INNER JOIN bocas_subte
ON bocas_subte.comuna_boca = ciclovias.COMUNA
```

Nome	Tipo	Esquema
Conexion		CREATE TABLE Conexion(id INT, codigo INT, nomoficial TEXT, nom_mapa TEXT, sentido TEXT, red_tp TEXT, BARRIO TEXT, id_boca INT, linea TEXT, estacion TEXT, destino_bo TEXT, lineas_de_ TEXT, cierra_fin TEXT, calle TEXT, calle2 TEXT, barrio_boca TEXT, COMUNA TEXT, dom_norma TEXT)
id	INT	"id" INT
codigo	INT	"codigo" INT
nomoficial	TEXT	"nomoficial" TEXT
nom_mapa	TEXT	"nom_mapa" TEXT
sentido	TEXT	"sentido" TEXT
red_tp	TEXT	"red_tp" TEXT
BARRIO	TEXT	"BARRIO" TEXT
id_boca	INT	"id_boca" INT
linea	TEXT	"linea" TEXT
estacion	TEXT	"estacion" TEXT
destino_bo	TEXT	"destino_bo" TEXT
lineas_de_	TEXT	"lineas_de_" TEXT
cierra_fin	TEXT	"cierra_fin" TEXT
calle	TEXT	"calle" TEXT
calle2	TEXT	"calle2" TEXT
barrio_boca	TEXT	"barrio_boca" TEXT
COMUNA	TEXT	"COMUNA" TEXT
dom_norma	TEXT	"dom_norma" TEXT

3. ¿Te falta algún otro dato? ¿Te interesaría sumar otro dataset? ¿porque?
   Sería interesante tener también las estaciones de bicicletas que hay cerca de las bocas por lo que podríamos sumar la tabla de estaciones, para eso nos haría falta algún id para hacer el relacionamiento. Esta unión nos permitiría tirar conclusiones sobre nuevas estrategias de expansión ubicando estaciones en lugares estratégicos para hacer conexiones con los subte. Además podríamos ampliar las sugerencias de rutas para los usuarios transportarse.

4. ¿Cuál de las 2 sentencias demanda menos recursos y por qué?
   a. SELECT COUNT(BOCAS_NUMBER) FROM BOCAS_SUBTE;
   b. SELECT COUNT(DISTINCT BOCAS_NUMBER) FROM BOCAS_SUBTE

   En este caso la función SELECT COUNT DISTINCT demanda menos recursos porque solo vamos a mostrar las salidas sin repeticiones. Pero la diferencia es mínima ya que lo que puede hacer más costosa la consulta es que existan muchos datos diferentes.

5. Agrupar los barrios por el campo “COMUNA” (tabla bocas de subte)

```
SELECT BARRIO
FROM bocas_subte
GROUP BY COMUNA
```

6. Listar las comunas que tengan más de 5 barrios (tabla bocas de subte)

```
SELECT comuna,
   Count(DISTINCT barrio) qtd_bairros
FROM bocas_subte
GROUP BY comuna HAVING qtd_bairros > 5
```

Nota: No existen comunas con más de 5 barrios en la tabla bocas de subte

```
SELECT comuna,
       count(DISTINCT barrio) qtd_bairros
FROM   bocas_subte
GROUP  BY comuna
ORDER  BY qtd_bairros DESC
```

Output:

```
1   SELECT comuna,
2          count(DISTINCT barrio) qtd_bairros
3   FROM   bocas_subte
4   GROUP  BY comuna
5   ORDER  BY qtd_bairros DESC
6
```

	comuna	qtd_bairros
1	Comuna 1	4
2	Comuna 4	3
3	Comuna 15	3
4	Comuna 13	3
5	Comuna 7	2
6	Comuna 5	2
7	Comuna 3	2
8	Comuna 6	1
9	Comuna 2	1
10	Comuna 14	1
11	Comuna 12	1
12	*NULL*	0

```
Execução finalizada sem erros.
Resultado: 12 linhas retornadas em 13 ms
Na linha 1:
SELECT comuna,
```

7. Crear una sentencia para actualizar el campo “numero_de_” del dataset bocas de subte, con ID = 5. El nuevo número de boca debe ser 30.

```
UPDATE  bocas_subte
SET numero_de_= 30
WHERE id = 5
```

8. Crear una sentencia para eliminar de la tabla CICLOVIAS al registro con ID = 100.

```
DELETE FROM ciclovias
```

```
WHERE id=100
```

9. Crear una sentencia para modificar el nombre del campo ESCALERA_M por ESCALERA_MECANICA de la tabla.

```
ALTER TABLE bocas_subte
RENAME COLUMN escalera_m TO escalera_mecanica
```

10. Crear una sentencia para modificar el nombre del campo numero_de_ por numero_boca de la tabla.

```
ALTER TABLE bocas_subte
RENAME COLUMN numero_de_ TO numero_boca
```

Perfil Senior

1. Crear una sentencia para generar la tabla VEHICULO, la cual va a contener los campos:
   Patente: primary key, alfanumérico, no nulo
   Marca: texto, no nulo
   Modelo: texto, no nulo
   Capacidad carga kg: decimal y no nulo
   Valor: decimal, puede ser nulo

```
CREATE TABLE Vehiculo (
   Patente CHAR(50) PRIMARY KEY NOT NULL,
   Modelo TEXT NOT NULL,
   Capacidad_carga_kg REAL NOT NULL,
   Valor REAL
)
```

2. Crear una sentencia para insertar un registro en la tabla SERVICIO_TECNICO_VEHICULO, sabiendo que el ID es un número entero, el vehículo y descripción son campos de texto, la fecha es un campo de tipo de dato fecha y en formato YYYYMMDD y por último, tanto los kilómetros del vehículo como el costo son numéricos con decimales.

```
INSERT INTO Servicio_tecnico_vehiculo (Id_servicio, Vehiculo, Fecha, Descripcion, Km, Costo)
VALUES(49, "Patente2", strftime('%Y-%m-%d', '2022-10-16'), "Pagamento a vista", 35000, 500);
```

3. Listar los vehículos en los que se gastó más en servicio técnico (reparaciones totales) que su valor de mercado.

```
SELECT Patente, Modelo, Valor, SUM(Costo) As Reparaciones_totales
FROM Vehiculo
LEFT JOIN Servicio_tecnico_vehiculo ON Servicio_tecnico_vehiculo.Vehiculo
= Vehiculo.Patente
GROUP BY Servicio_tecnico_vehiculo.Vehiculo
HAVING NOT SUM(Costo) < Valor
```

4. ¿Cuál es la carga no aprovechada total para cada mes del año actual? (Ej: en un viaje con un camión de 90kg de capacidad que transporta 40kg se tiene una carga no aprovechada de 50kg en ese viaje)

   SELECT Vehiculo.Capacidad_carga_kg – Transporte.Carga As Result
   FROM Vehiculo
   LEFT JOIN Transporte ON Transporte.ID_Patente = Vehiculo.Patente
   AND Transporte.Fecha_transporte > ‘2022-01-01’

5. Obtener la carga total enviada (por conductor) a destinos de entre 100 y 300 habitantes, desde orígenes que tengan más de 2000 habitantes. Considerar sólo los casos en los que el destino NO esté entre los preferidos del conductor (cada conductor tiene sus preferidos en la tabla PREFERENCIA_DESTINO). Ordenar los resultados en forma de ranking.

   SELECT * FROM Transporte
   WHERE id_origen = (SELECT id_origen
   FROM Lugar
   WHERE num_Habitantes = 2000)
   AND Destino = (SELECT id_origen
   FROM Lugar
   WHERE num_Habitantes >= 100
   AND num_Habitantes <= 300)
   Falta unir con la tabla Preferencia_Destino por documento del conductor…
   Falta SELECT ID_Patente, SUM(Carga) As Carga_total
   Falta usar ORDER By para ordenar por Carga_Total.

6. Crear una sentencia para actualizar el número de habitantes del lugar con ID = 5. El nuevo número de habitantes debe ser 300.

```
UPDATE Lugar
SET numero_habitantes = 300
WHERE Id_origen=5
```

7. Crear una sentencia para eliminar de la tabla SERVICIO_TECNICO_VEHICULO al registro con ID = 100.

```
PRAGMA foreign_keys = 0;
DELETE from Servicio_tecnico_vehiculo where Id_servicio = 100;
PRAGMA foreign_keys = 1;
```

8. Crear una sentencia para modificar el nombre de la tabla CONDUCTOR por EMPLEADO.

```
ALTER TABLE Conductor RENAME TO Empleado;
PRAGMA foreign_key_check;
```

9. ¿Cuál de las 2 sentencias demanda menos recursos y por qué?
   a. SELECT COUNT(DOCUMENT_NUMBER) FROM TRANSPORTE;
   b. SELECT COUNT(DISTINCT DOCUMENT_NUMBER) FROM TRANSPORTE

   La función COUNT DISTINCT devuelve el número de valores únicos en la columna. O sea, un COUNT (clave DISTINCT) básicamente tiene que ordenar previamente sus datos por clave y la clasificación puede ser (muy) costosa, especialmente en datos altamente cardinales (con muchos valores distintos). Un COUNT (*) simple solo tiene que contar el número de filas, sin necesidad de clasificación, por lo que siempre demandará menos recursos que COUNT (DISTINCT).

10. ¿Qué tipo de relaciones se pueden encontrar en el DER?

    Lugar -> Transporte (1: n)
    Vehiculo -> Transporte (1: n)
    Conductor -> Transporte (1: n)
    Preferencia_destino -> Transporte (1: n)
    Servicio_tecnico_vehiculo -> Transporte (1: n)
    Conductor -> Preferencia_destino (n: n)
    Vehiculo -> Servicio_tecnico_vehiculo (n: n)

11. Obtener la suma de las cargas por conductor (nombre, apellido)

```
SELECT Nombre, Apellido, SUM(Carga) As Carga_total
FROM Conductor
LEFT JOIN Transporte ON Transporte.Documento_conductor=
Conductor.Documento_conductor
GROUP By Transporte.Documento_conductor
```