Tarea de SQL en RMarkdown

```
library(DBI)
library(RSQLite)
# Tarea 1: Crear una tabla de alumnos en la misma base

# Crear la conexión a la base de datos
con <- dbConnect(RSQLite::SQLite(), "alumnos.db")
# Eliminar la tabla 'alumnos'
dbExecute(con, "DROP TABLE IF EXISTS alumnos")
```

```
## [1] 0
```

```
# Crear la tabla de alumnos
dbExecute(con, "
CREATE TABLE IF NOT EXISTS alumnos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre TEXT NOT NULL,
    calificacion INTEGER,
    promedio REAL,
    telefono INTEGER,
    email TEXT DEFAULT '@gmail.com'
);
")
```

```
## [1] 0
```

```
# Insertar los datos
dbExecute(con, "
INSERT INTO alumnos (nombre, calificacion, promedio, telefono, email)
VALUES
('Tomas Juri', 9 , 7.8 , 2612545422, 'tjuri02@gmail.com'),
('Matias Reyes', 9 , 9.8 , 2613831377, 'matireyes123@gmail.com'),
('Emilia Mujica', 8 , 9.3 , 2613401245, 'mujicaemilia0@gmail.com'),
('Victoria Mujica', 8 , 7.3 , 2613401241, 'mariavictoriamujicaa@gmail.com'),
('Mariano Garcia', 7 , 8.3 , 2616814048, 'marianogarciafabian98@gmail.com'),
('Candelaria Ortiz', 9 , 8.3 , 2615788288, 'ortiz.candelaria.2002@gmail.com'),
('Alvaro Buccolini', 9, 8.6, 2614709822, 'alvarobuccolini@gmail.com'),
('Ignacio Isgro', 8, 9.3, 2612449266, 'ignacioisgro12@gmail.com'),
('Lon Martinez', 6, 7.9, 2604638942, 'lonmartinez111@gmail.com'),
('Ivo Vazquez', 5, 4.6, 2616186593, 'ivovazquezbertani@gmail.com');
")
```

```
## [1] 10
```

```
# Consultar los datos
alumnos <- dbGetQuery(con, "SELECT * FROM alumnos;")
print(alumnos)
```

```
##    id           nombre calificacion promedio   telefono
## 1   1       Tomas Juri            9      7.8 2612545422
## 2   2     Matias Reyes            9      9.8 2613831377
## 3   3    Emilia Mujica            8      9.3 2613401245
## 4   4  Victoria Mujica            8      7.3 2613401241
## 5   5   Mariano Garcia            7      8.3 2616814048
## 6   6 Candelaria Ortiz            9      8.3 2615788288
## 7   7 Alvaro Buccolini            9      8.6 2614709822
## 8   8    Ignacio Isgro            8      9.3 2612449266
## 9   9     Lon Martinez            6      7.9 2604638942
## 10 10      Ivo Vazquez            5      4.6 2616186593
##                              email
## 1                tjuri02@gmail.com
## 2           matireyes123@gmail.com
## 3          mujicaemilia0@gmail.com
## 4   mariavictoriamujicaa@gmail.com
## 5  marianogarciafabian98@gmail.com
## 6  ortiz.candelaria.2002@gmail.com
## 7        alvarobuccolini@gmail.com
## 8         ignacioisgro12@gmail.com
## 9         lonmartinez111@gmail.com
## 10     ivovazquezbertani@gmail.com
```

```
# Desconectar de la base de datos
dbDisconnect(con)

# Tarea 2: Cuántas veces Jugó Godoy Cruz Antonio Tomba

# Leer el archivo CSV
partidos_arg <- read.csv("/cloud/project/ARG.csv")

# Verificar los nombres de las columnas
names(partidos_arg)
```

```
##  [1] "Country" "League"  "Season"  "Date"    "Time"    "Home"    "Away"
##  [8] "HG"      "AG"      "Res"     "PSCH"    "PSCD"    "PSCA"    "MaxCH"
## [15] "MaxCD"   "MaxCA"   "AvgCH"   "AvgCD"   "AvgCA"   "BFECH"   "BFECD"
## [22] "BFECA"
```

```
# Mostrar todos los valores únicos de una columna específica
lista_equipos <- unique(partidos_arg$Home)
print(lista_equipos)
```

```
##  [1] "Arsenal Sarandi"     "Velez Sarsfield"     "Racing Club"
##  [4] "Colon Santa FE"      "Quilmes"             "Newells Old Boys"
##  [7] "Godoy Cruz"          "San Lorenzo"         "River Plate"
## [10] "Tigre"               "San Martin S.J."     "Lanus"
## [13] "Estudiantes L.P."    "All Boys"            "Belgrano"
## [16] "Argentinos Jrs"      "Union de Santa Fe"   "Boca Juniors"
## [19] "Atl. Rafaela"        "Independiente"       "Gimnasia L.P."
## [22] "Rosario Central"     "Olimpo Bahia Blanca" "Defensa y Justicia"
## [25] "Banfield"            "Crucero del Norte"   "Sarmiento Junin"
## [28] "Huracan"             "Aldosivi"            "Nueva Chicago"
## [31] "Temperley"           "Patronato"           "Atl. Tucuman"
## [34] "Talleres Cordoba"    "Chacarita Juniors"   "San Martin T."
## [37] "Central Cordoba"     "Platense"            "Barracas Central"
## [40] "Colon Santa Fe"      "Instituto"           "Ind. Rivadavia"
## [43] "Dep. Riestra"
```

```
# Contar las veces que "Godoy Cruz" aparece en la columna "Home"
conteo_home <- sum(partidos_arg$Home == "Godoy Cruz")

# Contar las veces que "Godoy Cruz" aparece en la columna "Away"
conteo_away <- sum(partidos_arg$Away == "Godoy Cruz")

# Contar el total de veces que "Godoy Cruz" jugó
total_juegos <- conteo_home + conteo_away

# Imprimir los resultados
print(paste("Godoy Cruz jugó como local", conteo_home, "veces."))
```

```
## [1] "Godoy Cruz jugó como local 201 veces."
```

```
print(paste("Godoy Cruz jugó como visitante", conteo_away, "veces."))
```

```
## [1] "Godoy Cruz jugó como visitante 202 veces."
```

```
print(paste("Total de juegos de Godoy Cruz:", total_juegos))
```

```
## [1] "Total de juegos de Godoy Cruz: 403"
```

```
# Cuál fue su máxima cantidad de goles como visitante

# Filtrar los partidos en los que Godoy Cruz fue visitante
partidos_visitante <- subset(partidos_arg, Away == "Godoy Cruz")

# Obtener los goles que Godoy Cruz hizo como visitante
goles_visitante <- partidos_visitante$AG

# Encontrar la máxima cantidad de goles en un solo partido
max_goles <- max(goles_visitante, na.rm = TRUE)

# Imprimir el resultado
print(paste("La máxima cantidad de goles que Godoy Cruz hizo como visitante en un solo partido e
s:", max_goles))
```

```
## [1] "La máxima cantidad de goles que Godoy Cruz hizo como visitante en un solo partido es: 5"
```

```
# Tarea 3: Left joins entre tablas

# Dataset internos de R, podríamos verlos con el comando data()
data("AirPassengers") # Pasajeros
data("Titanic") # Sobrevivientes del Titanic

# Definir tablas de ejemplo
# Puedes reemplazar estos datasets con datos reales si los tienes
population <- data.frame(
  country = c("Brazil", "Germany", "Brazil", "Germany"),
  year = c(2000, 2000, 2001, 2001),
  population = c(170000000, 82000000, 171000000, 82500000)
)

who <- data.frame(
  country = c("Brazil", "Germany", "Brazil", "Germany"),
  year = c(2000, 2000, 2001, 2001),
  new_sp_m3544 = c(1000, 500, 1100, 550)
)

# Crear la conexión en memoria
conexion <- dbConnect(RSQLite::SQLite(), dbname = ":memory:")

# Almacenar datos de ejemplo en la base de datos
dbWriteTable(conn = conexion, name = "population", value = population)
dbWriteTable(conn = conexion, name = "who", value = who)

# Filtrar y unir tablas
query <- "
SELECT
  who.country,
  who.year,
  who.new_sp_m3544,
  population.population
FROM
  who
LEFT JOIN
  population
ON
  population.country = who.country
  AND population.year = who.year
WHERE
  who.country IN ('Brazil', 'Germany')
  AND who.year >= 2000
  AND who.year <= 2010
"

# Ejecutar la consulta y almacenar los resultados
M1_results <- dbGetQuery(conexion, query)
```

```
# Mostrar las primeras filas de los resultados
head(M1_results)
```

	country <chr>	year <dbl>	new_sp_m3544 <dbl>	population <dbl>
1	Brazil	2000	1000	1.70e+08
2	Germany	2000	500	8.20e+07
3	Brazil	2001	1100	1.71e+08
4	Germany	2001	550	8.25e+07

4 rows

Diagrama Entidad-Relación y Concepto de Normalización

Diagrama Entidad-Relación

Para generar el diagrama de entidad-relación en PDF, utilizamos el paquete `DiagrammeRsvg` y `rsvg` para convertir el gráfico a un formato compatible.

```
# Instalar los paquetes si es necesario
if(!require(DiagrammeR)) install.packages("DiagrammeR")
if(!require(DiagrammeRsvg)) install.packages("DiagrammeRsvg")
if(!require(rsvg)) install.packages("rsvg")

library(DiagrammeR)
library(DiagrammeRsvg)
library(rsvg)

# Crear el diagrama ER
gr <- grViz("
  digraph entidad_relacion {
    graph [layout = dot]

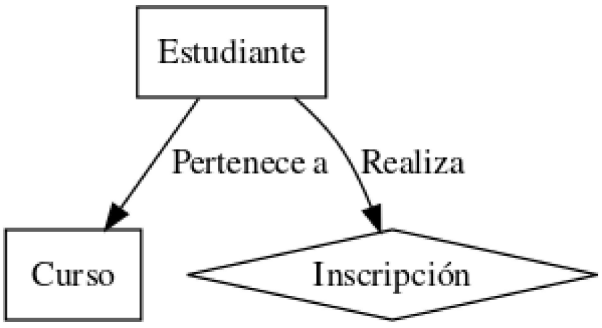
    node [shape = box]
    Estudiante [label = 'Estudiante']
    Curso [label = 'Curso']

    node [shape = diamond]
    Inscripcion [label = 'Inscripción']

    Estudiante -> Inscripcion [label = 'Realiza']
    Estudiante -> Curso [label = 'Pertenece a']
  }
")

# Exportar a SVG y luego convertir a PNG para compatibilidad con PDF
export_svg(gr) %>%
  charToRaw %>%
  rsvg_png("diagrama_entidad_relacion.png")

# Mostrar el diagrama en PDF
knitr::include_graphics("diagrama_entidad_relacion.png")
```



Concepto de Normalización

La **normalización** en bases de datos es un proceso que tiene como objetivo minimizar la redundancia y dependencia de los datos mediante la organización de los campos y tablas. La normalización divide las grandes tablas en tablas más pequeñas y las une utilizando relaciones. Este proceso garantiza que los datos se almacenen de forma eficiente y se eviten problemas como la duplicación de información.

El proceso de normalización se divide generalmente en varias formas normales (NF por sus siglas en inglés), donde cada forma normal tiene requisitos específicos:

- **Primera forma normal (1NF):** Asegura que los valores de cada columna sean atómicos (es decir, no divisibles) y que cada entrada en una tabla sea única.
- **Segunda forma normal (2NF):** Requiere que todos los atributos que no son clave dependan completamente de la clave principal.
- **Tercera forma normal (3NF):** Los atributos no clave deben depender únicamente de la clave primaria y no de otros atributos no clave (eliminando dependencias transitivas).

La normalización tiene como objetivo reducir la redundancia de los datos, mejorar la integridad de la base de datos y optimizar el rendimiento de las consultas.

Entendido. Aquí está el ejemplo de normalización que puedes añadir al final de tu documento RMarkdown:

```
## Ejemplo de Normalización

Supongamos que tenemos una tabla no normalizada sobre pedidos de productos:

| PedidoID | Cliente | Producto  | Cantidad | PrecioUnitario | DirecciónCliente       |
|----------|---------|-----------|----------|----------------|------------------------|
| 1        | Juan    | Televisor | 2        | 500            | Calle A, Ciudad X      |
| 2        | Ana     | Lavadora  | 1        | 300            | Calle B, Ciudad Y      |
| 3        | Juan    | Microondas| 1        | 200            | Calle A, Ciudad X      |

Esta tabla tiene redundancia de datos, como la repetición de la dirección del cliente. Para norm
alizar esta tabla, la descomponemos en varias tablas siguiendo las formas normales.

### Primera Forma Normal (1NF)

Aseguramos que todos los campos contengan valores atómicos y que cada fila sea única. La tabla o
riginal ya está en 1NF.

### Segunda Forma Normal (2NF)

Eliminamos las dependencias parciales. En este caso, separamos los datos del cliente de los dato
s del pedido. Creamos dos tablas: `Clientes` y `Pedidos`.


```r
# Crear las tablas normalizadas
clientes <- data.frame(
  ClienteID = c(1, 2),
  Nombre = c("Juan", "Ana"),
  Direccion = c("Calle A, Ciudad X", "Calle B, Ciudad Y")
)

pedidos <- data.frame(
  PedidoID = c(1, 2, 3),
  ClienteID = c(1, 2, 1),
  Producto = c("Televisor", "Lavadora", "Microondas"),
  Cantidad = c(2, 1, 1),
  PrecioUnitario = c(500, 300, 200)
)
```

Tercera Forma Normal (3NF)

Eliminamos las dependencias transitivas, asegurando que los atributos no clave dependan únicamente de la clave primaria. En este ejemplo, ya hemos alcanzado la 3NF ya que la tabla `Pedidos` no tiene atributos dependientes de otros atributos no clave.

Tablas Normalizadas

Clientes
```

ClienteID	Nombre	Direccion
1	Juan	Calle A, Ciudad X
2	Ana	Calle B, Ciudad Y

Pedidos

PedidoID	ClienteID	Producto	Cantidad	PrecioUnitario
1	1	Televisor	2	500
2	2	Lavadora	1	300
3	1	Microondas	1	200

Con estas dos tablas, hemos eliminado la redundancia y mejorado la organización de los datos, cumpliendo con la 3NF.