WORKING WITH DATABASES USING R

1. Installing libraries and connecting to the database

```
install.packages ("dplyr")
install.packages ("RSQLite")

library(dplyr)
```

2. Running the library and creating an SQLite connection

```
# Tạo kết nối đến cơ sở dữ liệu SQLite
conn <- DBI::dbConnect(RSQLite::SQLite(), "C:/Users/Lewis
Do/Documents/Semester 5/DSR301m/Lab1/chinook.db")

# kiểm tra kết nối
if (!is.null (conn)) {
message("Kết nối đến cơ sở dữ liệu thành công!")
} else {
message("Kết nối đến cơ sở dữ liệu thất bại!") }
```

3. Querying to view data in tables

```
# Truy vấn để lấy thông tin về các bảng trong cơ sở dữ liệu SQLite
query <- "SELECT name FROM sqlite_master WHERE type='table'"
# Thực hiện truy vấn
result <- dbGetQuery(conn, query)
# In kết quả
print(result)
```

4. Data Querying

```
# Lấy tất cả thông tin từ bảng albums
albums_query <- "SELECT * FROM albums"
albums_data <- dbGetQuery(conn, albums_query)
print (albums_query)
print (albums_data)

# Lấy tên và địa chỉ email của khách hàng
customers_query <- "SELECT FirstName, LastName, Email FROM customers"
customers_data <- dbGetQuery(conn, customers_query)
print (customers_query)
print(customers_data)
```

5. Data Checking

```
# Kiểm tra số lượng bản ghi
nrow (albums_data)
# Kiểm tra cấu trúc dữ liệu
str(albums_data)
# Kiểm tra thông tin tóm tắt
summary (customers_data)
```

6. Finding duplicate values

```
# Tìm các giá trị trùng lặp trong cột AlbumId
duplicate_values <- albums_data$AlbumId[duplicated
(albums_data$AlbumId)]
# In ra các giá trị trùng lặp
print (duplicate_values)
```

7. Removing records with NA values

```
# Loại bỏ các bản ghi có giá trị NA
cleaned_customers <- na.omit (customers_data)
print (cleaned_customers)

# Chuyển đổi kiểu dữ liệu
cleaned_customers$Email <- as.character(cleaned_customers$Email)
print(cleaned_customers $Email)
```

8. Performing SQL queries

```
install.packages ("dplyr")

library(dplyr)

# Lấy sô´lượng album theo nghệ sĩ
artist_album_count_query <-
"
SELECT artists.Name, COUNT(albums.AlbumId) AS AlbumCount
FROM artists
JOIN albums ON artists.ArtistId = albums.ArtistId
GROUP BY artists. Name
ORDER BY AlbumCount DESC
"
artist_album_counts <- dbGetQuery(conn, artist_album_count_query)
print(artist_album_counts)

# Vẽ biểu đồ`sô´lượng album theo nghệ sĩ
library(ggplot2)

ggplot(artist_album_counts, aes(x = reorder(Name, -AlbumCount), y =
AlbumCount)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Sô´lượng album theo nghệ sĩ", x = "Nghệ sĩ", y = "Sô´
lượng album")
```

9. Running Totals Query

```
# Truy vấn tính tổng doanh thu lũy kê´cho môĩ tháng với tên bảng đúng

running_total_query <- "
SELECT Month, Revenue, SUM (Revenue) OVER (ORDER BY Month) AS
RunningTotal
FROM (
SELECT strftime('%Y-%m', InvoiceDate) AS Month,
```

```
SUM (Total) AS Revenue
FROM invoices
GROUP BY Month
)
"
running_total_data <- dbGetQuery (conn, running_total_query)
# In kết quả
print (running_total_data)
```

10. Checking table structure

```
# Kiểm tra cấu trúc bảng invoice items để xác định các cột
columns_query <- "PRAGMA table_info(invoice_items)"
columns_info <- dbGetQuery(conn, columns_query)
print(columns_info)
```

11. Combining Queries

```
# Truy vấn tính tổng số lượng và doanh thu cho từng nghệ sĩ với cột
Unit Price
combined_artist_query <- "
SELECT artists.NamE,
       COUNT (tracks. TrackId) AS TrackCount,
       SUM(invoice_items.Quantity * invoice_items.Unit_Price) AS
TotalRevenue FROM artists
JOIN albums ON artists.ArtistId = albums.ArtistId
JOIN tracks ON albums.AlbumId = tracks.AlbumId
JOIN invoice_items ON tracks.TrackId = invoice_items.TrackId
JOIN invoices ON invoice_items.InvoiceId = invoices.InvoiceId
GROUP BY artists.Name
ORDER BY TotalRevenue DESC
"
combined_artist_data <- dbGetQuery (conn, combined_artist_query)
# In kết quả
print (combined_artist_data)
```

12. Segmentation Analysis

```
# Truy vấn phân khúc khách hàng dựa trên số lượng đơn hàng và tổng chi
tiêu
segmentation_query <- "
SELECT
  CASE
    WHEN OrderCount > 5 AND TotalSpent > 1000 THEN 'VIP'
    WHEN OrderCount BETWEEN 3 AND 5 AND TotalSpent BETWEEN 500 AND
1000 THEN 'Regular'
    ELSE 'New'
  END AS CustomerSegment,
  COUNT(CustomerId) AS Count
FROM (
  SELECT
    customers.CustomerId,
    COUNT(invoices.InvoiceId) AS OrderCount,
```

```
    SUM(invoices.Total) AS TotalSpent
  FROM customers
  JOIN invoices ON customers.CustomerId = invoices.CustomerId
  GROUP BY customers.CustomerId
)
GROUP BY CustomerSegment;
"

segmentation_data <- dbGetQuery (conn, segmentation_query)
# In kết quả
print (segmentation_data)
```

13. Statistical Analysis Query Tính toán các chỉ số thống kê như trung bình, độ lệch chuẩn, và phân phối.

```
# Truy vấn tính doanh thu theo tháng với tên cột chính xác
monthly_revenue_query <- "
SELECT strftime ('%Y-%m', InvoiceDate) AS Month, SUM (Total) AS
Revenue
FROM invoices
GROUP BY Month
ORDER BY Month
"
monthly_revenue_data <- dbGetQuery (conn, monthly_revenue_query)
# In kết quả
print (monthly_revenue_data)
```

14. Trend Analysis Query

```
# Truy vấn tính số´lượng đơn hàng theo tháng với tên cột
monthly_orders_query <- "
SELECT strftime('%Y-%m', InvoiceDate) AS Month,
COUNT (InvoiceId) AS OrderCount
FROM invoices
GROUP BY Month
ORDER BY Month
"
monthly_orders_data <- dbGetQuery (conn, monthly_orders_query)
# In kết quả
print (monthly_orders_data)
```

15. Window Functions Query

```
# Tính tổng doanh thu và doanh thu trung bình theo từng khách hàng
window_function_query <- "
SELECT CustomerId, Total, SUM (Total) OVER (PARTITION BY CustomerId)
AS CustomerTotal, AVG (Total) OVER (PARTITION BY CustomerId) AS
CustomerAverage
FROM Invoices
"
window_data <- dbGetQuery (conn, window_function_query)
print (window_data)
```

16. Calculating aggregate indicators for groups

```
# Tính tổng số´tiền chi tiêu của từng khách hàng và sắp xếp theo số´
tiền chi tiêu
customer_spending_query <- "
SELECT customers.CustomerId,
      customers.FirstName,
      customers.LastName,
      SUM(Invoices.Total) AS TotalSpent
FROM customers
JOIN Invoices ON customers.CustomerId = Invoices.CustomerId
GROUP BY customers.CustomerId
ORDER BY TotalSpent DESC
"
customer_spending_data <- dbGetQuery (conn, customer_spending_query)
print (customer_spending_data)
```

17. Time Series Analysis Query

```
# Kiểm tra cấú trúc bảng invoices để xem tên các cột
columns_query <- "PRAGMA table_info (invoices)"
columns_info <- dbGetQuery(conn, columns_query)
print (columns_info)

# Tính doanh thu theo tháng sử dụng cột InvoiceDate
monthly_revenue_query <-
"
SELECT strftime('%Y-%m', InvoiceDate) AS Month,
       SUM (Total) AS Revenue
FROM invoices
GROUP BY Month
ORDER BY Month
"
monthly_revenue_data <- dbGetQuery (conn, monthly_revenue_query)
# In kết quả
print (monthly_revenue_data)
```

18. Monthly Revenue Chart

```
library (dplyr)

#Tính phần trăm doanh thu cho mỗi tháng
monthly_revenue_data <- monthly_revenue_data %>%
  mutate(Revenue = (Revenue/ sum(Revenue)) * 100)

# Vẽ biểu đồ`tròn
ggplot (monthly_revenue_data, aes(x="", y=Revenue, fill=Month)) +
  geom_bar(stat="identity", width=1, color='white') +
  coord_polar(theta="y") +
  labs(title = "Biểu đồ`tròn: Doanh thu theo tháng") +
  theme_void() + # xóa nền để tạo biểu đồ`tròn rõ ràng hơn
  theme(legend.position = "right") # đưa thêm chú thích bên phải
```

19. Query to Calculate Total Quantity and Revenue

```
# Truy vấn tình tổng số´lượng và doanh thu cho từng nghệ sĩ với cột
Unit Price
combined_artist_query <- "
SELECT artists.Name,
       COUNT (tracks.TrackId) AS TrackCount,
       SUM(invoice_items.Quantity * invoice_items.Unit_Price) AS
TotalRevenue
FROM artists
JOIN albums ON artists.Artistid = albums.ArtistId
JOIN tracks ON albums.AlbumId = tracks.AlbumId
JOIN invoice_items ON tracks.TrackId = invoice_items.TrackId
JOIN invoices ON invoice_items.InvoiceId = invoices.InvoiceId
GROUP BY artists.Name
ORDER BY TotalRevenue DESC
"
combined_artist_data <- dbGetQuery(conn, combined_artist_query)
# In kết quả
print (combined_artist_data)
```

20. Querying to view data information of a table

```
# Truy vấn dữ liệu từ bảng invoices
invoices_data <- tbl(conn, "invoices")

# Hiển thị thông tin bảng invoices
result <- dbGetQuery(conn, "SELECT * FROM invoices")
print(result)
```

21. Conditional filtering

```
# Lọc các hàng có giá trị trong cột Total lớn hơn 15
invoices_data_filtered <- invoices_data %>% filter(Total > 15)

# In ra kết quả
print(invoices_data_filtered)
```

22. Sorting data

```
invoices_data <- invoices_data %>%
  arrange(BillingCountry, desc(Total))
invoices_data
```

23. Summarizing data

```
summary_data <- invoices_data %>%
  group_by(BillingCountry) %>%
  summarize(avg_total = mean(Total, na.rm = TRUE))

summary_data
```

24. Column merging

```
print(invoices_data)

invoices_data <- invoices_data %>%
```

```
  mutate(BillingLocal = paste(BillingAddress, BillingCity, sep=", "))
invoices_data
```

25. Adding columns

```
dbExecute(conn, "ALTER TABLE invoices ADD COLUMN Ghi_chu TEXT")

result <- dbGetQuery(conn, "SELECT * FROM invoices")
print(result)
```

26. Renaming columns

```
dbExecute(conn, "ALTER TABLE invoices RENAME COLUMN Ghi_chu to Notes")

result <- dbGetQuery(conn, "SELECT * FROM invoices")
print(result)
```

27. Deleting columns

```
dbExecute(conn, "ALTER TABLE invoices DROP COLUMN Notes")

result <- dbGetQuery(conn, "PRAGMA table_info('invoices')")

print(result)
```

28. Filtering rows with null or NA

    Thực hiện lọc ra những dòng có cột null hoặc NA trong cột BillingPostalCode, thực hiện tạo cột mới tên là payment, thực hiện thêm dữ liệu tự động cho cột payment là “pay in cash” và in ra kết quả

```
# Truy vấn dữ liệu từ bảng invoices
invoices_data <- tbl(conn, "invoices")

# Lọc ra các dòng có giá trụ null hoặc NA
filtered_data <- invoices_data %>%
  filter(is.na(BillingPostalCode) | BillingPostalCode == "")

print(filtered_data)

# Thêm cột mới "payment"
filtered_data <- invoices_data %>%
  mutate(payment = "pay in cash")

print(filtered_data)
```

29. Calculating column totals

```
# Truy vấn
invoices_data <- tbl(conn, "invoices")

summary_data <- invoices_data %>%
  group_by(BillingCountry) %>%
```

```
  summarise(total_amount = sum(Total))

print(head(summary_data))

print(summary_data)
```

30. Drawing charts

```
# Vẽ biểủ đô`thống kê
ggplot (summary_data, aes (x = BillingCountry, y = total_amount)) +
geom_bar (stat = "identity", fill = "skyblue") +
labs (x = "Billing Country", y = "Total Amount", title = "Total Amount
by Billing Country") +
theme_minimal() +
theme (axis.text.x = element_text (angle = 90, vjust= 0.5, hjust=1))
```

31. Practicing statistical analysis

```
# Truy vấń dữ liệu từ bảng invoices
invoices_data <- tbl (conn, "invoices")

# Tổng cộng của cột "Total"
total_sales <- invoices_data %>%
  summarise (total_sales = sum (Total))
# In ra tổng doanh thu
print (total_sales)

# Sô´lượng hóa đơn môĩ quôć gia
invoices_per_country <- invoices_data %>%
  count(BillingCountry)
# In ra sô´lượng hóa đơn môĩ quôć gia print (invoices_per_country)

# Tổng doanh thu theo quôć gia
sales_per_country <- invoices_data %>%
  group_by(BillingCountry) %>%
  summarise (total_sales = sum (Total))
# In ra tổng doanh thu theo quôć gia
print(sales_per_country)

# Tổng doanh thu trung bình môĩ hóa đơn
average_invoice_total <- invoices_data %>%
  summarise (average_total = mean (Total))
# In ra tổng doanh thu trung bình môĩ hóa đơn
print(average_invoice_total)

# Sô´lượng hóa đơn theo tháng
invoices_per_month <- invoices_data %>%
  mutate (InvoiceMonth = format (as.Date (InvoiceDate), "%Y-%m"))

# In ra sô´lượng hóa đơn theo tháng
print(invoices_per_month)
```

```
# thống kê khách hàng của từng quốc gia nghe nhạc thể loại nào, Tính
tổng doanh thu cho từng thể loại
# Truy vấn dữ liệu từ các bảng (invoices, invoice_items, customers,
tracks, genres)
invoices_data <- tbl (conn, "invoices")
invoice_items_data <- tbl (conn, "invoice_items")
customers_data <- tbl (conn, "customers")
tracks_data <- tbl (conn, "tracks")
genres_data <- tbl (conn, "genres")

# Kết hợp dữ liệu từ các bảng để lấy thông tin về quốc gia, thể loại
nhạc và doanh thu
customer_genre_revenue <- invoices_data %>%
  inner_join (invoice_items_data, by = "InvoiceId") %>%
  inner_join(customers_data, by = "CustomerId") %>%
  inner_join (tracks_data, by = "TrackId") %>%
  inner_join (genres_data, by = "GenreId") %>%
  group_by (BillingCountry, GenreName) %>%
  summarise (total_revenue = sum (Unit_Price *Quantity))

# In ra dữ liệu thống kê
print (customer_genre_revenue)
{r}
# Đóng kết nối
DBI: dbDisconnect (conn)
```