Healthcare Data Analysis

Project Overview: This project uses generated healthcare data, similiar to what one might find on MyAvatar, to create an example analysis using SQL and R. This analysis will cover calling data from an in-memory database, cleaning data, and ultimately aggregating and drawing insight from the data.

1. Load Libraries and Data Because we are working in R Markdown and we want to use SQL, we'll need to load SQL libraries and also tidyverse so we can visualize the data.

```
knitr::opts_chunk$set(echo = TRUE)
library(DBI)
library(RSQLite)
library(glue)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Define file paths
data_dir <- "C:/Users/isabe/Downloads/BHSN Interview Project/"
finance_path <- glue("{data_dir}finance_metrics.csv")
client_demo_path <- glue("{data_dir}client_demo.csv")
```

We generate an in-memory SQLite database for ease of use writing in SQL. Our data sets have client demographics and financial metrics Then we merge the data by "Service Type" which is a column both data sets have in common.

```
# Connect to an SQLite database; this database resides purely in memory
ourDatabase <- dbConnect(RSQLite::SQLite(), ":memory:")


# Load data from CSV files
finance_metrics <- read.csv(finance_path)
client_demo <- read.csv(client_demo_path)

# Write data into the SQLite database
dbWriteTable(ourDatabase, "FinanceMetrics", finance_metrics)
```

```
dbWriteTable(ourDatabase, "ClientDemographics", client_demo)

# Merging the datasets on ServiceType
dbExecute(ourDatabase, "
  CREATE TABLE ConsolidatedData AS
  SELECT f.*, c.Age, c.Gender, c.Ethnicity, c.PrimaryDiagnosis, c.TreatmentOutcome
  FROM FinanceMetrics f
  JOIN ClientDemographics c ON f.ServiceType = c.ServiceType
")
```

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

```
ConsolidatedData <- dbGetQuery(ourDatabase, "SELECT * FROM ConsolidatedData")
write.csv(ConsolidatedData, "{data_dir}cleanData.csv", row.names = FALSE)
```

2. Clean the Data Here we'll clean the data, which means we'll replace missing values with identity values, and remove duplicates using the distinct function.

```
# SQL query to handle missing values and remove duplicates
dbExecute(ourDatabase, "
  UPDATE ConsolidatedData
  SET Age = COALESCE(Age, 0),
      Gender = COALESCE(Gender, 'Unknown'),
      Ethnicity = COALESCE(Ethnicity, 'Unknown'),
      PrimaryDiagnosis = COALESCE(PrimaryDiagnosis, 'Not Specified'),
      TreatmentOutcome = COALESCE(TreatmentOutcome, 'Not Specified')
")
```

```
## [1] 3668
```

```
# Creating a clean table with unique records
dbExecute(ourDatabase, "
  CREATE TABLE CleanData AS
  SELECT DISTINCT * FROM ConsolidatedData
")
```

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

3. Aggregate and Visualize Data We'll pull the data into R for visualizations. For SQL/R integration we need to create R-friendly data-frames from the tables we have in SQL.

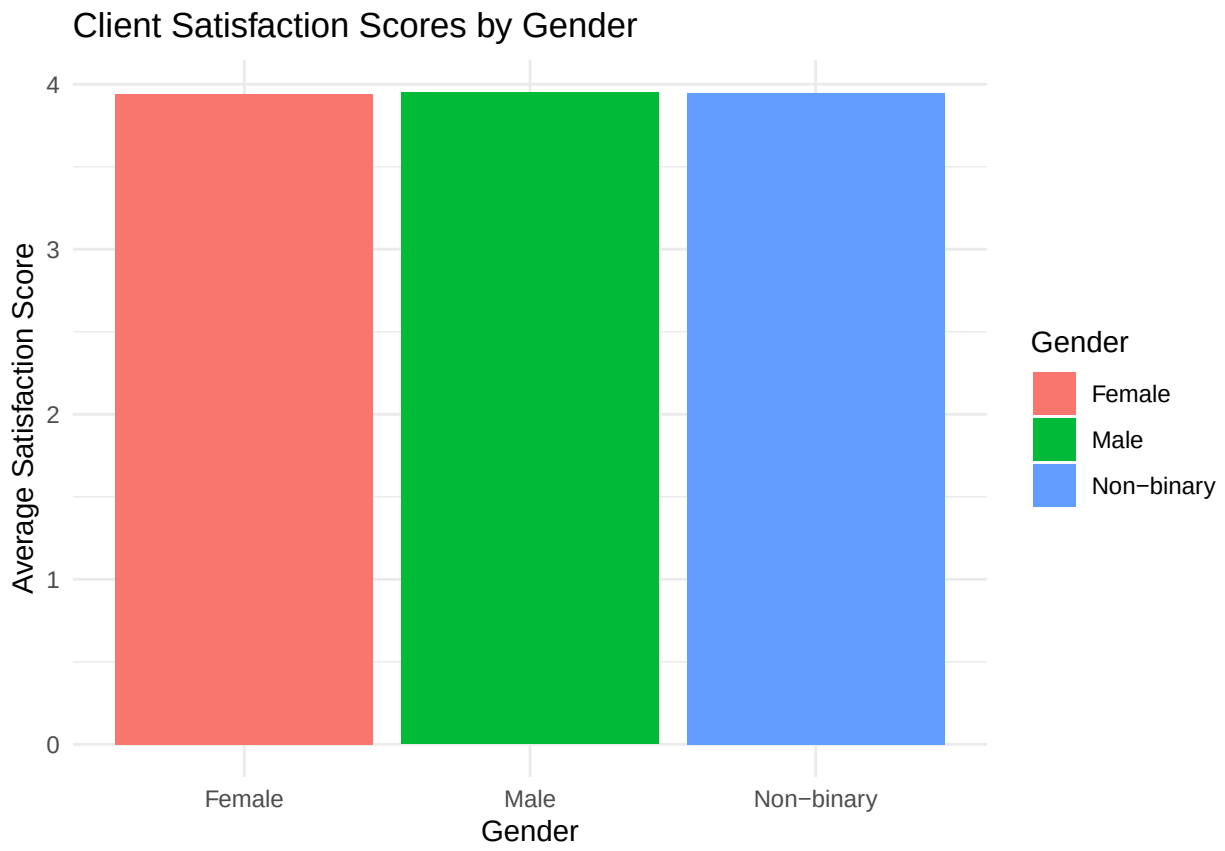
```
# Load clean data from SQL for visualization in R
cleanEHR <- dbReadTable(ourDatabase, "CleanData")
```

We'll create some aggregations with SQL. For now we'll focus on Client Satisfaction Scores by Gender, Outcome by Age, and Billing Accuracy and Claim Denial Rates.

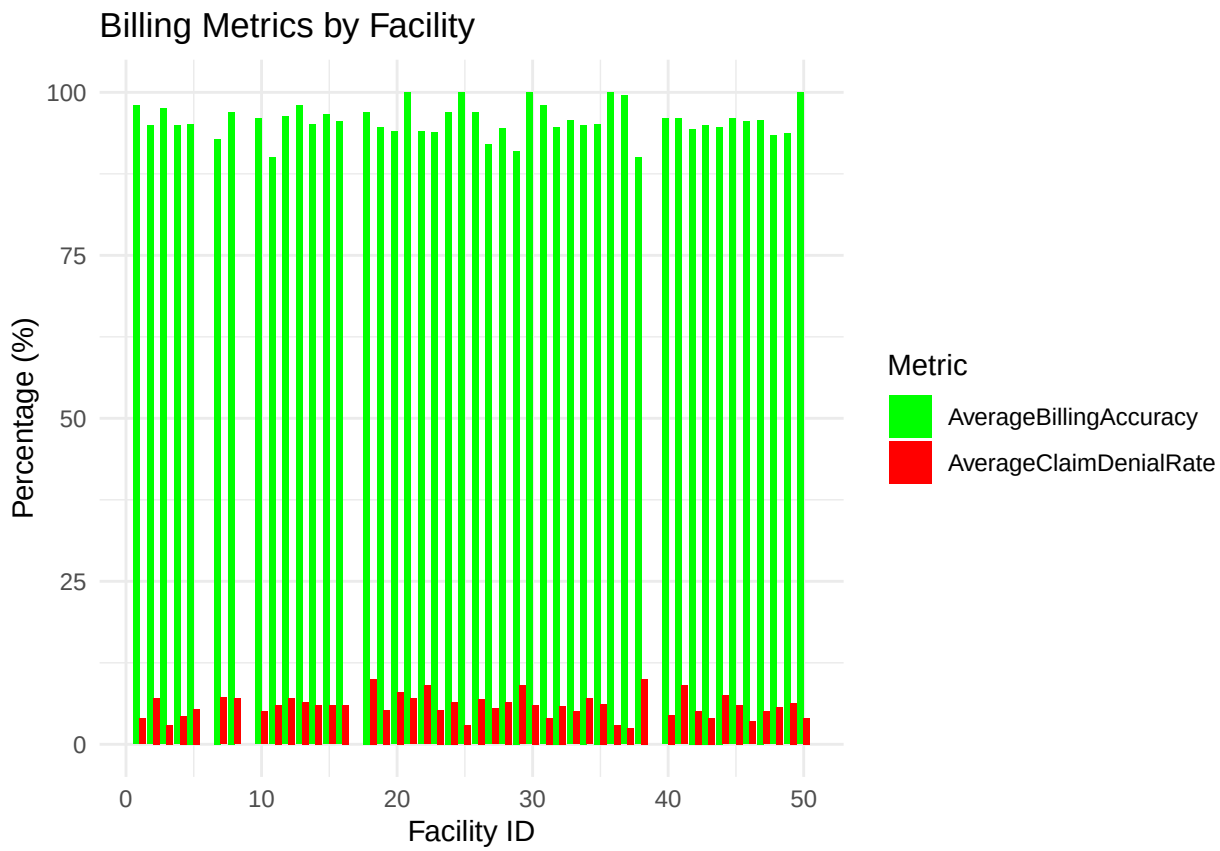
```
# Execute SQL queries and fetch results into data frames
satisfaction_by_gender <- dbGetQuery(ourDatabase, "
  SELECT Gender, AVG(ClientSatisfactionScore) AS AverageSatisfaction
  FROM CleanData
  GROUP BY Gender
")

# For the Age Outcome data we'll split up the ages into 'bins'
# which will help with visualizations later.

age_outcome_data <- dbGetQuery(ourDatabase, "
  SELECT
    CASE
      WHEN Age BETWEEN 18 AND 25 THEN '18-25'
      WHEN Age BETWEEN 26 AND 35 THEN '26-35'
      WHEN Age BETWEEN 36 AND 45 THEN '36-45'
      WHEN Age BETWEEN 46 AND 55 THEN '46-55'
      WHEN Age BETWEEN 56 AND 65 THEN '56-65'
      WHEN Age BETWEEN 66 AND 75 THEN '66-75'
      WHEN Age > 75 THEN '76+'
      ELSE 'Unknown' END AS AgeGroup,
    TreatmentOutcome,
    COUNT(*) AS Count
  FROM CleanData
  GROUP BY AgeGroup, TreatmentOutcome
  ORDER BY AgeGroup, TreatmentOutcome
")

billing_accuracy_data <- dbGetQuery(ourDatabase, "
  SELECT FacilityID, AVG(BillingAccuracyRate) AS AverageBillingAccuracy,
  AVG(ClaimDenialRate) AS AverageClaimDenialRate
  FROM CleanData
  GROUP BY FacilityID
")
billing_data_long <- pivot_longer(billing_accuracy_data, cols
                                  = c(AverageBillingAccuracy, AverageClaimDenialRate),
                                  names_to = "Metric", values_to = "Value")

on.exit(dbDisconnect(ourDatabase), add = TRUE)
```

Using ggplot from the tidyverse package, we can visualize our aggregations with simple bar plots.

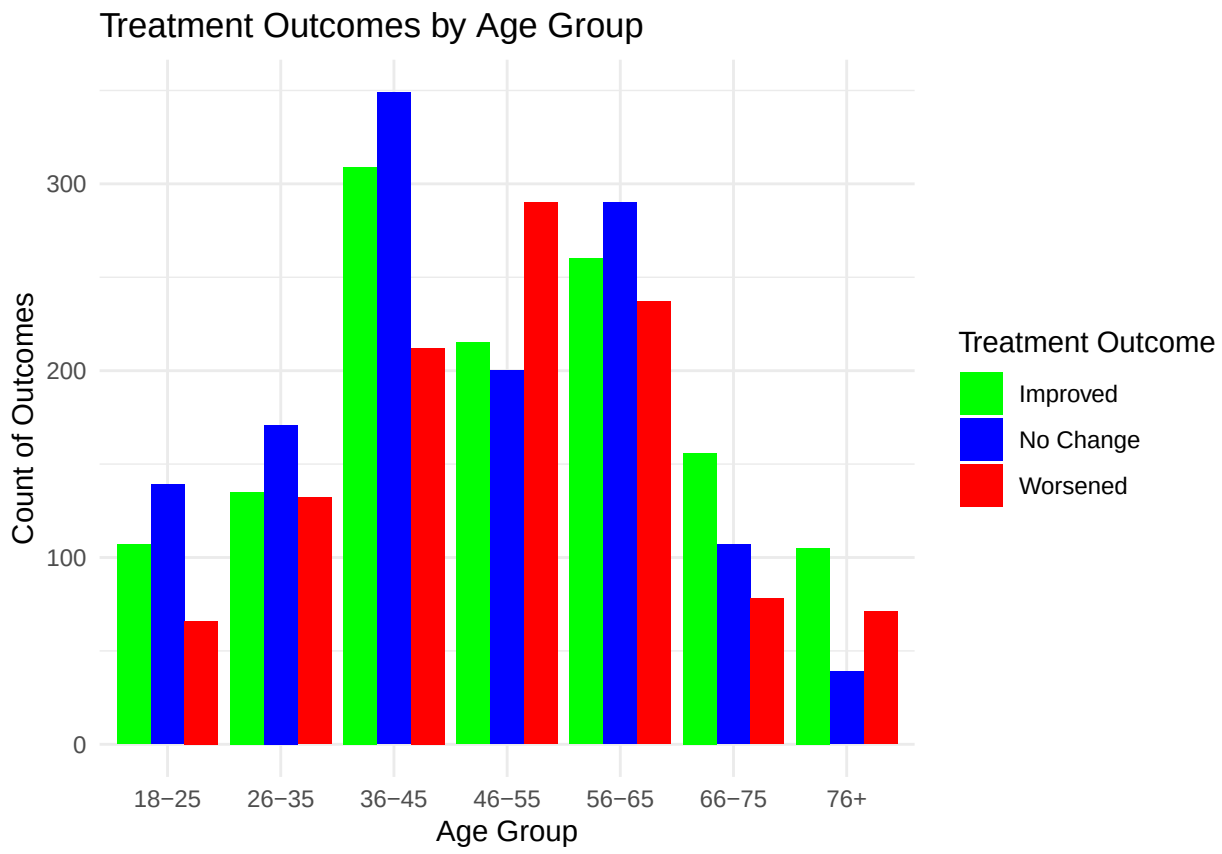
```
# Visualize Client Satisfaction Scores by Gender
ggplot(satisfaction_by_gender, aes(x = Gender, y = AverageSatisfaction, fill
                                   = Gender)) +
  geom_bar(stat = "identity") +
  labs(title = "Client Satisfaction Scores by Gender", y = "Average Satisfaction Score",
       x = "Gender") +
  theme_minimal()
```




```r
# Visualize Billing Accuracy and Claim Denial Rates by Facility
ggplot(billing_data_long, aes(x = FacilityID, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("green", "red")) +
  labs(title = "Billing Metrics by Facility", y = "Percentage (%)", x = "Facility ID") +
  theme_minimal()
```




```r
# Visualize Outcome by Age
ggplot(age_outcome_data, aes(x = AgeGroup, y = Count, fill = TreatmentOutcome)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("Improved" = "green", "Worsened" = "red",
                               "No Change" = "blue")) +
  labs(title = "Treatment Outcomes by Age Group", y = "Count of Outcomes",
       x = "Age Group", fill = "Treatment Outcome") +
  theme_minimal()
```



Summary

In this example project, we successfully demonstrated several key data management techniques within an integrated R and SQL environment:

- **Data Cleaning:** Implemented basic data cleaning operations, including the replacement of missing values with default identifiers and the removal of duplicate records to ensure data integrity.
- **Database Interaction:** Demonstrated the capability to effectively interact with an in-memory SQLite database, showcasing the loading, updating, and querying of data.
- **R/SQL Integration:** Showcased seamless integration between R and SQL, utilizing the strengths of both to perform and visualize comprehensive data analysis tasks.