Question 1(a) – Python Code

```
import pandas as pd

from sqlalchemy import create_engine, URL

# Set up the sqlalchemy engine

url_object = URL.create(

'mysql+pymysql',

username='root',

password='anl503',

host='localhost',

database='anl503jul24')

engine = create_engine(url_object)

# Read in excel data into Pandas DataFrame

file_path = "ECA_data.xlsx"  #excel file

sheet_name = "ECA_data"  #excel file sheet name

df = pd.read_excel(file_path, sheet_name=sheet_name)

# View DataFrame summary info

print(df.info())

# Upload DataFrame to MySQL as 'shopper_profile' table.

df.to_sql('shopper_profile', engine,

if_exists='replace', index=False)
```

The following is the summary information of `df.info()`after the excel file is loaded into Pandas DataFrame. We are required to check if the total entries are the same as the excel file and all the required variables are included in the dataframe.

```
>>> print(df.info())

<class 'pandas.core.frame.DataFrame'>
```

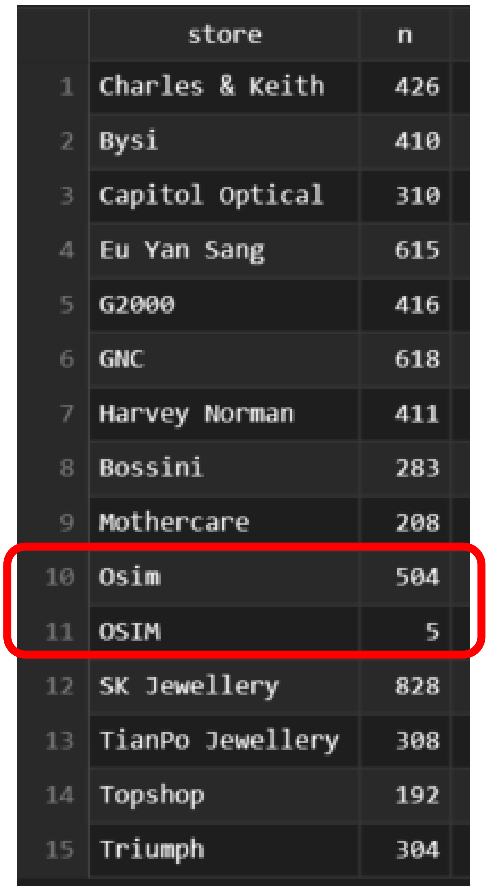
```
RangeIndex: 5838 entries, 0 to 5837
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   mall         5838 non-null   object
 1   store        5838 non-null   object
 2   citizenship  5838 non-null   object
 3   age          5838 non-null   int64
 4   gender       5838 non-null   object
 5   race         5838 non-null   object
dtypes: int64(1), object(5)
memory usage: 273.8+ KB
None
```

Question 1(b)

When we run the requested MySQL statement, we observed that there is an error in one of the store names, “OSIM”. From **Figure 1**, the characters of the store name were not uniform. To standardize the store name, we will run the below MySQL statement to change related store names to “OSIM”.

Figure 1

MySQL result



	store	n
1	Charles & Keith	426
2	Bysi	410
3	Capitol Optical	310
4	Eu Yan Sang	615
5	G2000	416
6	GNC	618
7	Harvey Norman	411
8	Bossini	283
9	Mothercare	208
10	Osim	504
11	OSIM	5
12	SK Jewellery	828
13	TianPo Jewellery	308
14	Topshop	192
15	Triumph	304

SQL Code:

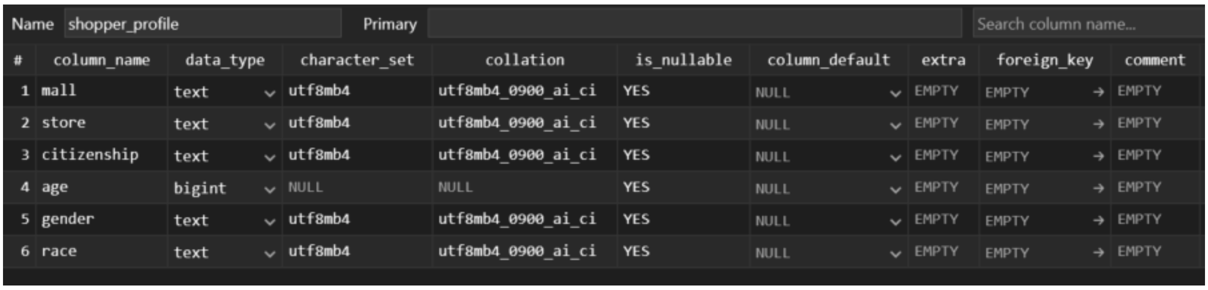
```
-- Question 1(b)
-- List all stores in the dataset and number of records'
count.

SELECT store COLLATE utf8mb4_bin AS store,

        COUNT(*) AS n

 FROM  shopper_profile

GROUP BY 1;

-- Rename "Osim" to standardize name "OSIM".

UPDATE shopper_profile

SET store = 'OSIM'

WHERE store = 'Osim';
```

Question 1(c)

To improve the performance on the database, we should try to limit the bytes used for each record. Based on Stablepoint (n.d.), we are able to reduce 1 extra storage space when we changed from text to `varchar()` type. In addition, we could also restrict the number of characters for each of the variables that we knew that it cannot be exceptionally long. For example, in gender and race, it is impossible for the variables' characters to reach more than 10 characters for gender and 20 characters for race. As for age, we could reduce the integer type to `tinyint` because it is not possible for anyone to live until 255 years. Hence, we applied the new data types using the below SQL code and the result is shown in **Figure 2**.

Figure 2

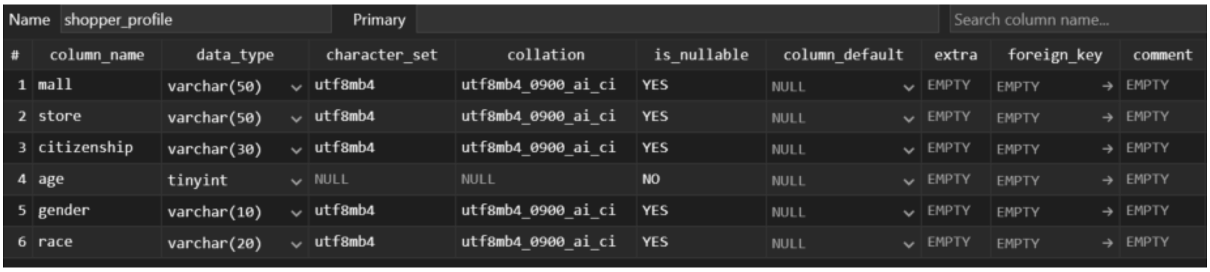
`shopper_profile` data structure table (Before Change)



Name shopper_profile Primary

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	mall	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
2	store	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
3	citizenship	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
4	age	bigint	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
5	gender	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
6	race	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY

`shopper_profile` data structure table (After Change)



Name shopper_profile Primary

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	mall	varchar(50)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
2	store	varchar(50)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
3	citizenship	varchar(30)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
4	age	tinyint	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
5	gender	varchar(10)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
6	race	varchar(20)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY

SQL Code:

```
-- Question 1(c)

-- Change the data type of each variable.

ALTER TABLE shopper_profile
```

```
-- Reduce the length of the characters as
-- malls generally do not have long names.
CHANGE `mall` `mall` VARCHAR(50) NULL COMMENT '',
-- Reduce the length of the characters as
-- stores generally do not have long names.
CHANGE `store` `store` VARCHAR(50) NULL COMMENT '',
-- Reduce the length of the characters as
-- citizenship generally does not have long names.
CHANGE `citizenship` `citizenship` VARCHAR(30)
     NULL COMMENT '',
-- Change data type of age from smallint to tinyint as
-- it is impossible to live until 255.
CHANGE `age` `age` tinyint NOT NULL COMMENT '',
-- Reduce the length of the characters as
-- gender generally does not have long names.
CHANGE `gender` `gender` VARCHAR(10) NULL COMMENT '',
-- Reduce the length of the characters as
-- races generally do not have long names.
CHANGE `race` `race` VARCHAR(20) NULL COMMENT '';
```

Question 1(d)

To understand shopper profiles across different malls, we require a dashboard that allows users to select specific malls and age groups, displaying charts of shopper ratios in each mall (**Figure 3**), visited stores and their gender (**Figure 4**), and shopper citizenship (**Figure 5**).

The dashboard, as shown in **Figure 6**, visualizes these shopper behaviours by segmenting age groups into Silent Generation (Ages 77+), Baby Boomers (Ages 58-76), Generation X (Ages 43-57), Millennials (Ages 27-42), and Generation Z (Ages 11-26). Millennials are set as the default group, as they represent the largest shopper base in the dataset.

Key findings indicate that Baby Boomers and the Silent Generation favour shopping at Hougang Mall, likely due to its residential location. On the other hand, younger age groups, including Generation Z and Millennials, prefer Jurong Point, which is located near universities, offices, and factories, and Vivo City, known for its wider retail offerings.

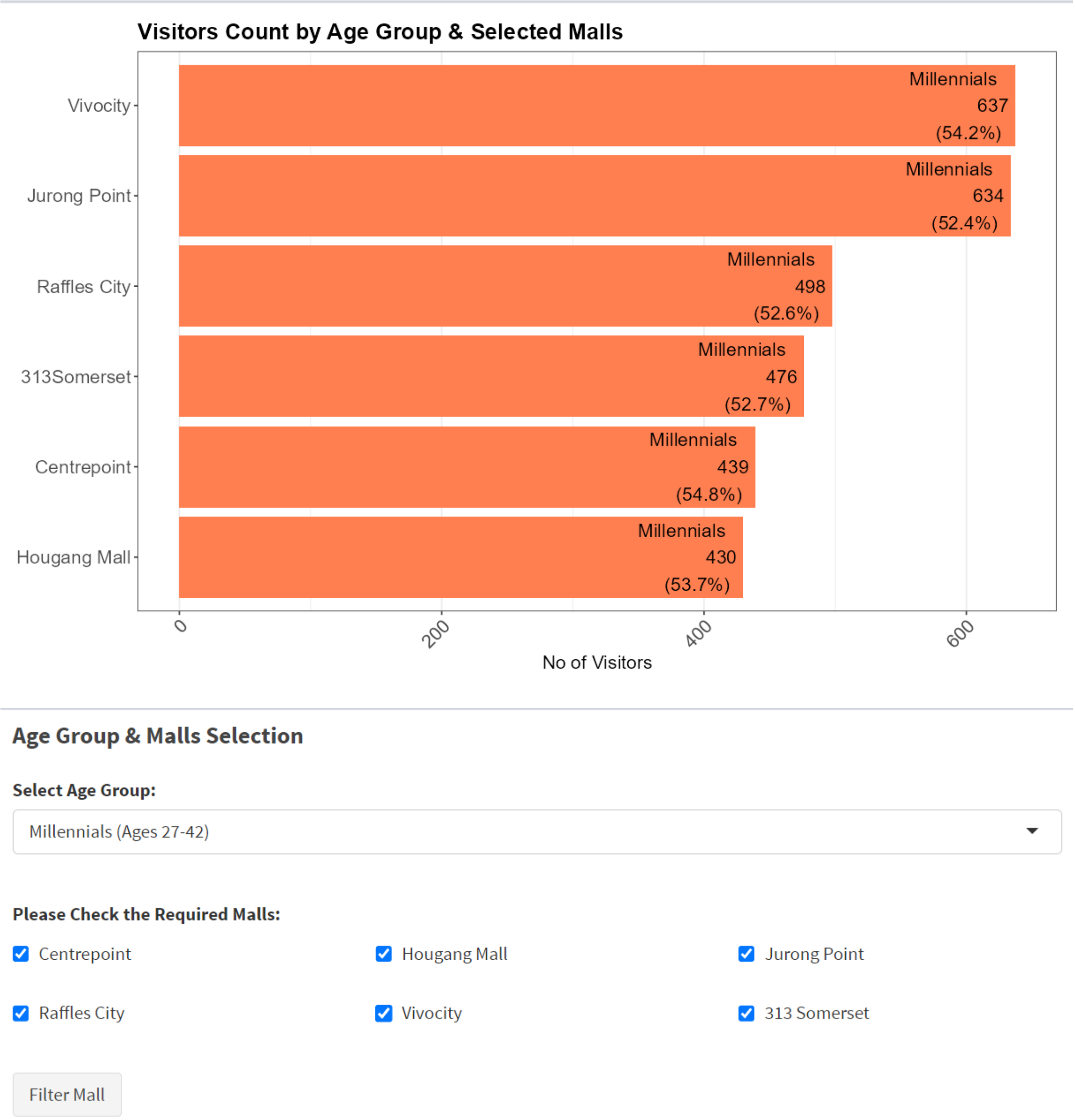
Shopping preferences also differ by age group. Baby Boomers and the Silent Generation primarily purchase health supplements, while Generation X and Millennials prefer health products and jewellery. Generation Z, alongside health supplements, has a stronger interest in fashion-forward items.

Despite these differences, commonalities exist across all age groups, particularly regarding demographics. Most shoppers, regardless of age, are Chinese and either Singapore citizens or permanent residents.

The dashboard's usability can be improved by adding sections for tourists and food stores, which would help malls attract a wider range of shoppers and potentially boost sales.

Figure 3

Millennial’s Visitor Counts in Selected Malls



Age Group & Malls Selection

Select Age Group:

Millennials (Ages 27-42)

Please Check the Required Malls:

☑ Centrepoint ☑ Hougang Mall ☑ Jurong Point

☑ Raffles City ☑ Vivocity ☑ 313 Somerset

Figure 4

Millennial's Visitor Counts in Selected Malls' Stores

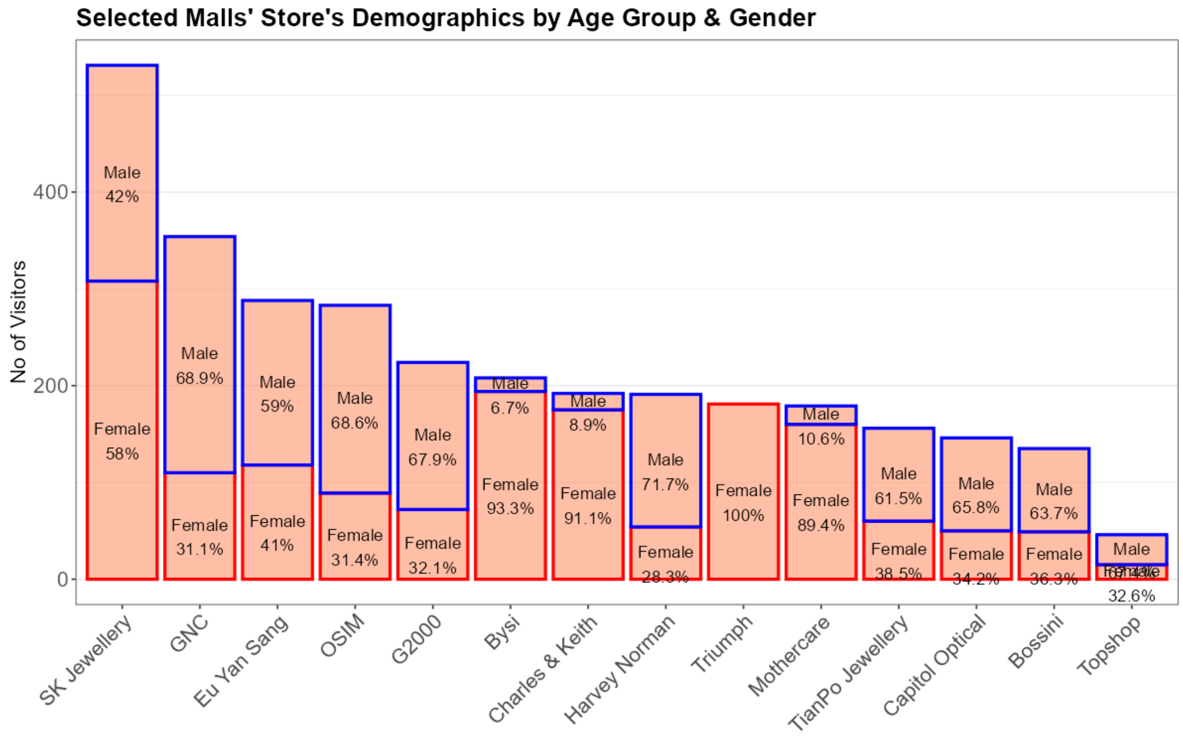
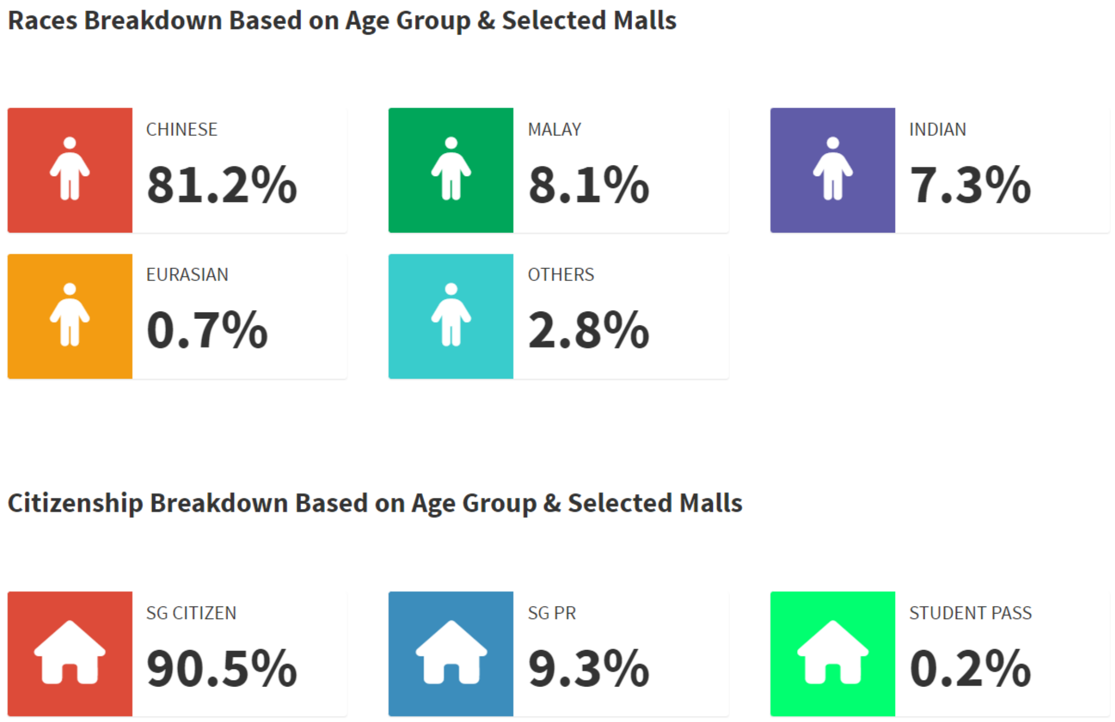


Figure 5

Interactive Shoppers' Races and Citizenship Breakdown



Races Breakdown Based on Age Group & Selected Malls

CHINESE 81.2%

MALAY 8.1%

INDIAN 7.3%

EURASIAN 0.7%

OTHERS 2.8%

Citizenship Breakdown Based on Age Group & Selected Malls

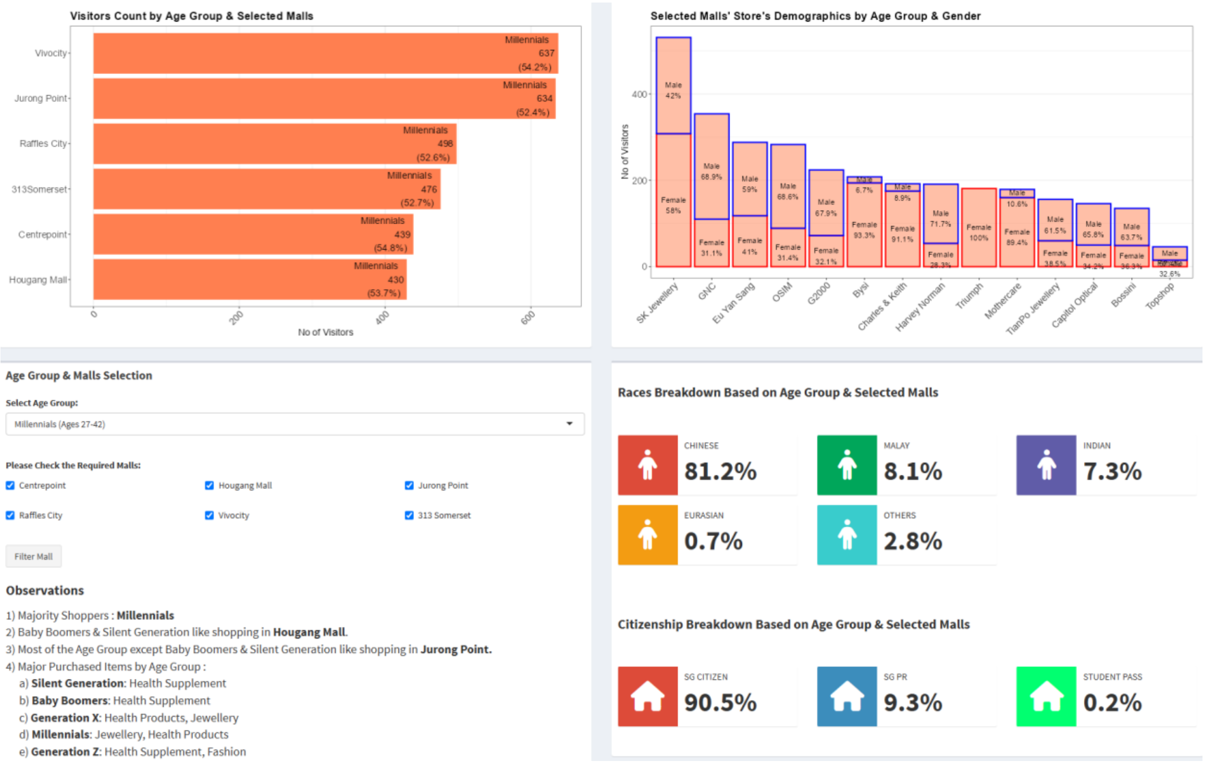
SG CITIZEN 90.5%

SG PR 9.3%

STUDENT PASS 0.2%

Figure 6

Shopper Profile Dashboard



Age Group & Malls Selection

Select Age Group:

Millennials (Ages 27-42)

Please Check the Required Malls:

☑ Centrepoint ☑ Hougang Mall ☑ Jurong Point

☑ Raffles City ☑ Vivocity ☑ 313 Somerset

Observations

1) Majority Shoppers : **Millennials**
2) Baby Boomers & Silent Generation like shopping in **Hougang Mall**.
3) Most of the Age Group except Baby Boomers & Silent Generation like shopping in **Jurong Point.**
4) Major Purchased Items by Age Group :
   a) **Silent Generation**: Health Supplement
   b) **Baby Boomers**: Health Supplement
   c) **Generation X**: Health Products, Jewellery
   d) **Millennials**: Jewellery, Health Products
   e) **Generation Z**: Health Supplement, Fashion

Races Breakdown Based on Age Group & Selected Malls

CHINESE 81.2%	MALAY 8.1%	INDIAN 7.3%
EURASIAN 0.7%	OTHERS 2.8%	

Citizenship Breakdown Based on Age Group & Selected Malls

SG CITIZEN 90.5%	SG PR 9.3%	STUDENT PASS 0.2%

R Code:

```
# Load required library

library(RMySQL)

library(tidyverse)

library(ggrepel)

library(shiny) # To quick view dashboard in R Console

library(shinydashboard) # To create a dashboard

library(shinyWidgets) # To use HTML codes in Dashboard.

theme_set(theme_bw()) # To Set the general theme.


#Set Working Directory

setwd("C:/Users/seiky/OneDrive/Documents/Course/ANL503/ECA/")


# Connect SQL server

con = dbConnect(MySQL(),

                    dbname = "anl503jul24",
                user = "root",
                password = "anl503")

# Extract shopper_profile table into R
data.frame = dbGetQuery(con, "SELECT *
                           FROM shopper_profile;")

# Close connection as connection is limited.
dbDisconnect(con)

# To review the class types for each variables.
str(data.frame)

# To review the records for verification.
head(data.frame)
dim(data.frame)

# To check for any NULL values
colSums(!is.na(data.frame))

# To create age group based on the following
#  Silent Generation (SilentGen) : Ages above 77
#  Baby Boomers: Ages 58-76 (born 1946-1964)
#  Generation X (GenX): Ages 43-57 (born 1965-1980)
#  Millennials: Ages 27-42 (born 1981-1996)
#  Generation Z (GenZ): Ages 11-26 (born 1997-2012)
data.frame.bin <-
  data.frame %>%
  mutate(age_bin =
         cut(
             age,
             breaks = c(10, 26, 42, 57, 76, Inf),
             labels = c("GenZ", "Millennials", "GenX",
                        "BabyBoomers", "SilentGen"),
             right = FALSE
            )
        )

# Summarize Shopper age group and percentage in malls.
data.frame.age <- data.frame.bin %>%
  group_by(mall, age_bin) %>%
  summarize(N = n()) %>%
  mutate(total_agebin = sum(N),
         percent_agebin = round((N / total_agebin) * 100, 1))

# Assign colour palettes to each age grouping.
colours <- c("GenZ" = "#00BFFF",
             "Millennials" = "#FF7F50",
             "GenX" = "#808000",
             "BabyBoomers" = "#FFD700",
             "SilentGen" = "#D3D3D3"
            )

# Assign colour palettes to each gender grouping.
gender_colours <- c("Male" = "blue", "Female" = "red")

## Dashboard Creation with Shiny packages
ui <- dashboardPage(
  dashboardHeader(disable = TRUE),
  dashboardSidebar(disable = TRUE),
  # Plot charts by rows using fluidRow.
  dashboardBody(fluidRow(
    box(plotOutput("plot1", height = 500)),
    box(plotOutput("plot2", height = 500)),
    box(
      title = strong("Age Group \n& Malls Selection"),
      # Dropdown Box to select the age group variable to plot
      # Millennials is selected as default as it is the main
      # shopper in malls.
      selectInput(
        "AgeGroup",
        "Select Age Group:",
        choices =
          c(
            "Silent Generation (Ages above 77)" = "SilentGen",
            "Baby Boomers (Ages 58-76)" = "BabyBoomers",
            "Generation X (Ages 43-57)" = "GenX",
            "Millennials (Ages 27-42)" = "Millennials",
            "Generation Z (Ages 11-26)" = "GenZ"
           ),
        selected = "Millennials"
      ),
      br(),
      # 1st CheckBox row of malls for selection. The selected malls
      # will be plotting together in the charts.
      strong("Please Check the Required Malls:"),
      fluidRow(
        column(
               4, checkboxInput("mall_centrepoint", "Centrepoint",
                                value = TRUE)
              ),
        column(4, checkboxInput("mall_hougang", "Hougang Mall",
                                value = TRUE)
              ),
        column(4, checkboxInput("mall_jurong", "Jurong Point",
                                value = TRUE))
              ),
      # 2nd checkBox row of malls for selection.
      fluidRow(
        column(4, checkboxInput("mall_raffles", "Raffles City",
                                value = TRUE)
               ),
        column(4, checkboxInput("mall_vivo", "Vivocity",
                                value = TRUE)
               ),
        column(4, checkboxInput("mall_313", "313 Somerset",
                                value = TRUE))
               ),
      br(),
      actionButton("filter_button", "Filter Mall"),
      br(),
      br(),
      tags$div(style = "font-size:20px;", strong(p("Observations")),
               ),
      # Observation section that allows HTML code to be used.
      htmlOutput("myObservation")
      ),
      # Create easy to view races breakdown in this section.
      # It will change upon the above selection.
      box(
        br(),
        tags$div(style = "font-size:20px;", strong(
        p("Races Breakdown Based on Age Group & Selected Malls")
        ), ),
      br(),
      br(),
      fluidRow(
        infoBoxOutput("Chinese01"),
        infoBoxOutput("Malay01"),
        infoBoxOutput("Indian01"),
        infoBoxOutput("Eurasian01"),
        infoBoxOutput("Others01")
      ),
      br(),
      br(),
      br(),
      # Create easy to view races breakdown in this section.
      # It will change upon the above selection.
      tags$div(style = "font-size:20px;", strong(
        p("Citizenship Breakdown Based on Age Group & Selected
Malls")
      ), ),
      br(),
      br(),
      fluidRow(
        infoBoxOutput("SGC"),
        infoBoxOutput("SGPR"),
        infoBoxOutput("StudentPass")
      ),
      br(),
      )
    ))
)

# Define server/dashboard logic
```

```
server <- function(input, output) {
  # To filter the malls on dashboard on CheckBox response.
  selected_malls <- reactiveVal(
    c(
      "Centrepoint",
      "Hougang Mall",
      "Jurong Point",
      "Raffles City",
      "Vivocity",
      "313Somerset"
    )
  )

  # Update selected malls based on checkbox input
  observeEvent(input$filter_button, {
    selected <- c()

    if (input$mall_centrepoint)
      selected <- c(selected, "Centrepoint")
    if (input$mall_hougang)
      selected <- c(selected, "Hougang Mall")
    if (input$mall_jurong)
      selected <- c(selected, "Jurong Point")
    if (input$mall_raffles)
      selected <- c(selected, "Raffles City")
    if (input$mall_vivo)
      selected <- c(selected, "Vivocity")
    if (input$mall_313)
      selected <- c(selected, "313Somerset")

    # Update the reactive value with selected malls
    selected_malls(selected)
  })

  # Generate 1st plot
  # To understand the demographics (age) by visiting each malls.
  output$plot1 <- renderPlot({
    data.frame.age %>%
      filter(mall %in% selected_malls(),
             age_bin == input$AgeGroup) %>%
      ggplot(aes(
        x = reorder(mall, N),
        y = N,
        group = mall,
        fill = age_bin
      )) +
      geom_col() +
      coord_flip() +
      geom_text(
        data = data.frame.age %>%
          filter(mall %in% selected_malls(),
                 age_bin == input$AgeGroup),
        aes(label = paste0(
          age_bin, "\n", N, "\n(", percent_agebin, "%)"
        )),
        position = "stack",
        hjust = 1.2,
        color = "black",
        size = 5
      ) +
      scale_fill_manual(values = colours) +
      labs(
        title = "Visitors Count by Age Group & Selected Malls",
        x = "Malls",
        y = "No of Visitors",
        fill = "Age Group"
      ) +
      theme(
        legend.position = "none",
        title = element_text(size = 14, face = "bold"),
        axis.title.x = element_text(size = 14, face = "plain"),
        axis.text.x = element_text(
          angle = 45,
          size = 14,
          hjust = 1
        ),
        axis.title.y = element_blank(),
        axis.text.y = element_text(size = 14),
        panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank()
      )
  })

  # Generate 2nd plot
  # To understand the demographics (age/gender) visiting
  # Each stores.
  output$plot2 <- renderPlot({
    # Summarize Shopper age and gender group by store level
    # based on selection.
    data.frame.binrev <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, store, gender) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin, store) %>%
      mutate(total_agebin = sum(N),
        percent_gender = round((N / total_agebin) * 100, 1)) %>%
      mutate(gender = factor(gender,
                             levels = c("Male", "Female"))) %>%
      filter(age_bin == input$AgeGroup)

    # To plot the chart with malls filtered.
    ggplot(data = data.frame.binrev, aes(
      x = reorder(store, -N, sum),
      y = N,
      fill = age_bin,
      colour = gender
    )) +
      geom_col(alpha = 0.5,
               position = "stack",
               size = 1) +
      geom_text(
        aes(
          label = paste0(gender, "\n", percent_gender, "%"),
          colour = gender
        ),
        position = position_stack(vjust = 0.5),
        hjust = 0.5,
        vjust = 0.8,
        color = "black",
        size = 4
      ) +
      scale_fill_manual(values = colours) +
      scale_colour_manual(values = gender_colours) +
      labs(
        title ="Selected Malls' Store's Demographics by Age Group &
Gender",
        x = "Stores",
        y = "No of Visitors",
        fill = "Age Group"
      ) +
      theme(
        legend.position = "none",
        title = element_text(size = 14, face = "bold"),
        axis.title.x = element_blank(),
        axis.text.x = element_text(
          angle = 45,
          size = 14,
          hjust = 1
        ),
        axis.title.y = element_text(size = 14, face = "plain"),
        axis.text.y = element_text(size = 14),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank()
      )
  })

  # Generate InfoBox
  # To easily understand the demographics (races)
  # visiting each stores by using InfoBox.
  output$Chinese01 <- renderInfoBox({
    # Summarize Shopper race (Chinese)
    summarized_data <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, race) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin) %>%
      mutate(total_agebin = sum(N),
        percent_race = round((N / total_agebin) * 100, 1)) %>%
      filter(age_bin == input$AgeGroup, race == "Chinese")

    # Provide the display of the InfoBox
    if (nrow(summarized_data) > 0) {
      chinese_percent <- summarized_data$percent_race[1]

      infoBox(
        title = "Chinese",
        value = tags$div(paste0(chinese_percent, "%"),
                         style = "font-size: 40px;"),
        icon = icon("person"),
        color = "red",
        fill = FALSE
      )
    } else {
      infoBox(
        title = "Chinese",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("person"),
        color = "red",
        fill = FALSE
      )
    }
  })

  output$Malay01 <- renderInfoBox({
    # Summarize Shopper race (Malay)
    summarized_data <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, race) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin) %>%
      mutate(total_agebin = sum(N),
        percent_race = round((N / total_agebin) * 100, 1)) %>%
      filter(age_bin == input$AgeGroup, race == "Malay")

    # Provide the display of the InfoBox
    if (nrow(summarized_data) > 0) {
      malay_percent <- summarized_data$percent_race[1]

      infoBox(
        title = "Malay",
        value = tags$div(paste0(malay_percent, "%"),
                         style = "font-size: 40px;"),
        icon = icon("person"),
        color = "green",
        fill = FALSE
      )
    } else {
      infoBox(
        title = "Malay",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("person"),
        color = "green",
        fill = FALSE
      )
    }
  })
```

```
output$Indian01 <- renderInfoBox({
  # Summarize Shopper race (Indian)
  summarized_data <- data.frame.bin %>%
    filter(mall %in% selected_malls()) %>%
    group_by(age_bin, race) %>%
    summarize(N = n(), .groups = "drop") %>%
    group_by(age_bin) %>%
    mutate(total_agebin = sum(N),
      percent_race = round((N / total_agebin) * 100, 1)) %>%
    filter(age_bin == input$AgeGroup, race == "Indian")

  # Provide the display of the InfoBox
  if (nrow(summarized_data) > 0) {
    indian_percent <- summarized_data$percent_race[1]

    infoBox(
      title = "Indian",
      value = tags$div(paste0(indian_percent, "%"),
                       style = "font-size: 40px;"),
      icon = icon("person"),
      color = "purple",
      fill = FALSE
    )
  } else {
    infoBox(
      title = "Indian",
      value = tags$div("0%", style = "font-size: 40px;"),
      icon = icon("person"),
      color = "purple",
      fill = FALSE
    )
  }
})

output$Eurasian01 <- renderInfoBox({
  # Summarize Shopper race (Eurasian)
  summarized_data <- data.frame.bin %>%
    filter(mall %in% selected_malls()) %>%
    group_by(age_bin, race) %>%
    summarize(N = n(), .groups = "drop") %>%
    group_by(age_bin) %>%
    mutate(total_agebin = sum(N),
      percent_race = round((N / total_agebin) * 100, 1)) %>%
    filter(age_bin == input$AgeGroup, race == "Eurasian")

  # Provide the display of the InfoBox
  if (nrow(summarized_data) > 0) {
    eurasian_percent <- summarized_data$percent_race[1]

    infoBox(
      title = "Eurasian",
      value = tags$div(paste0(eurasian_percent, "%"),
                       style = "font-size: 40px;"),
      icon = icon("person"),
      color = "yellow",
      fill = FALSE
    )
  } else {
    infoBox(
      title = "Eurasian",
      value = tags$div("0%", style = "font-size: 40px;"),
      icon = icon("person"),
      color = "yellow",
      fill = FALSE
    )
  }
})
output$Others01 <- renderInfoBox({
  # Summarize Shopper race (Others)
  summarized_data <- data.frame.bin %>%
    filter(mall %in% selected_malls()) %>%
    group_by(age_bin, race) %>%
    summarize(N = n(), .groups = "drop") %>%
    group_by(age_bin) %>%
    mutate(total_agebin = sum(N),
      percent_race = round((N / total_agebin) * 100, 1)) %>%
    filter(age_bin == input$AgeGroup, race == "Others")

  # Provide the display of the InfoBox
  if (nrow(summarized_data) > 0) {
    others_percent <- summarized_data$percent_race[1]

    infoBox(
      title = "Others",
      value = tags$div(paste0(others_percent, "%"),
                       style = "font-size: 40px;"),
      icon = icon("person"),
      color = "teal",
      fill = FALSE
    )
  } else {
    infoBox(
      title = "Others",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("person"),
        color = "teal",
        fill = FALSE
      )
    }

  })

  output$SGC <- renderInfoBox({
    # Summarize Shopper citizenship (Singapore Citizen)
    summarized_data <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, citizenship) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin) %>%
      mutate(total_agebin = sum(N),
             percent_citizen = round((N / total_agebin) * 100,
1)) %>%
      filter(age_bin == input$AgeGroup, citizenship == "SG Citizen")

    # Provide the display of the InfoBox
    if (nrow(summarized_data) > 0) {
      sgc_percent <- summarized_data$percent_citizen[1]

      infoBox(
        title = "SG Citizen",
        value = tags$div(paste0(sgc_percent, "%"),
                         style = "font-size: 40px;"),
        icon = icon("house"),
        color = "red",
        fill = FALSE
      )
    } else {
      infoBox(
        title = "SG Citizen",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("house"),
        color = "red",
        fill = FALSE
      )
    }
  })

  output$SGPR <- renderInfoBox({
    # Summarize Shopper itizenship (Singapore PR)
    summarized_data <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, citizenship) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin) %>%
      mutate(total_agebin = sum(N),
             percent_citizen = round((N / total_agebin) * 100,
1)) %>%
      filter(age_bin == input$AgeGroup, citizenship == "SG PR")

    # Provide the display of the InfoBox
    if (nrow(summarized_data) > 0) {
      sgpr_percent <- summarized_data$percent_citizen[1]

      infoBox(
        title = "SG PR",
        value = tags$div(paste0(sgpr_percent, "%"),
                         style = "font-size: 40px;"),
        icon = icon("house"),
        color = "light-blue",
        fill = FALSE
      )
    } else {
      infoBox(
        title = "SG PR",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("house"),
        color = "light-blue",
        fill = FALSE
      )
    }
  })

  output$StudentPass <- renderInfoBox({
    # Summarize Shopper citizenship (Student Pass)
    summarized_data <- data.frame.bin %>%
      filter(mall %in% selected_malls()) %>%
      group_by(age_bin, citizenship) %>%
      summarize(N = n(), .groups = "drop") %>%
      group_by(age_bin) %>%
      mutate(total_agebin = sum(N),
             percent_citizen = round((N / total_agebin) * 100,
1)) %>%
      filter(age_bin == input$AgeGroup, citizenship == "Student
Pass")

    # Provide the display of the InfoBox
    if (nrow(summarized_data) > 0) {
      student_percent <- summarized_data$percent_citizen[1]

      infoBox(
        title = "Student Pass",
        value = tags$div(paste0(student_percent, "%"),
                         style = "font-size: 40px;"),
        icon = icon("house"),
        color = "lime",
        fill = FALSE
      )
    } else {
      infoBox(
        title = "Student Pass",
        value = tags$div("0%", style = "font-size: 40px;"),
        icon = icon("house"),
        color = "lime",
        fill = FALSE
      )
    }
  })

  # Display the observation summary using HTML coding.
  output$myObservation <- renderText({
    HTML(
      "
    <div style='font-size: 18px;'>
      1) Majority Shoppers : <B>Millennials</B><BR>
      2) Baby Boomers & Silent Generation like shopping in
<B>Hougang Mall</B>.<BR>
      3) Most of the Age Group except Baby Boomers & Silent
Generation like shopping in <B>Jurong Point.</B><BR>
      4) Major Purchased Items by Age Group :<BR>
      <div style='margin-left: 20px;'>
         a) <B>Silent Generation</B>: Health Supplement<br>
         b) <B>Baby Boomers</B>: Health Supplement<br>
         c) <B>Generation X</B>: Health Products, Jewellery<br>
         d) <B>Millennials</B>: Jewellery, Health Products<br>
         e) <B>Generation Z</B>: Health Supplement, Fashion<br>
      </div>
      </div>
         "
    )
  })
}

# Run the dashboard result using shinyApp
shinyApp(ui, server)

sessionInfo()
# End
```

References

Stablepoint (n.d.). *Choosing Between VARCHAR and TEXT in MySQL.* Retrieved from https://kb.stablepoint.com/docs/choosing-between-varchar-and-text-in-mysql