R Notebook

```
# This R code includes libraries necessary for database connectivity and data visualization.
# - The 'RMySQL' library allows us to interact with MySQL databases.
# - The 'DBI' library provides a common interface to various database systems.
# - The 'ggplot2' library is used for creating advanced and customized data visualizations using the Gr
library(RMySQL)
```

```
## Loading required package: DBI
```

```
library(DBI)
library(ggplot2)
```

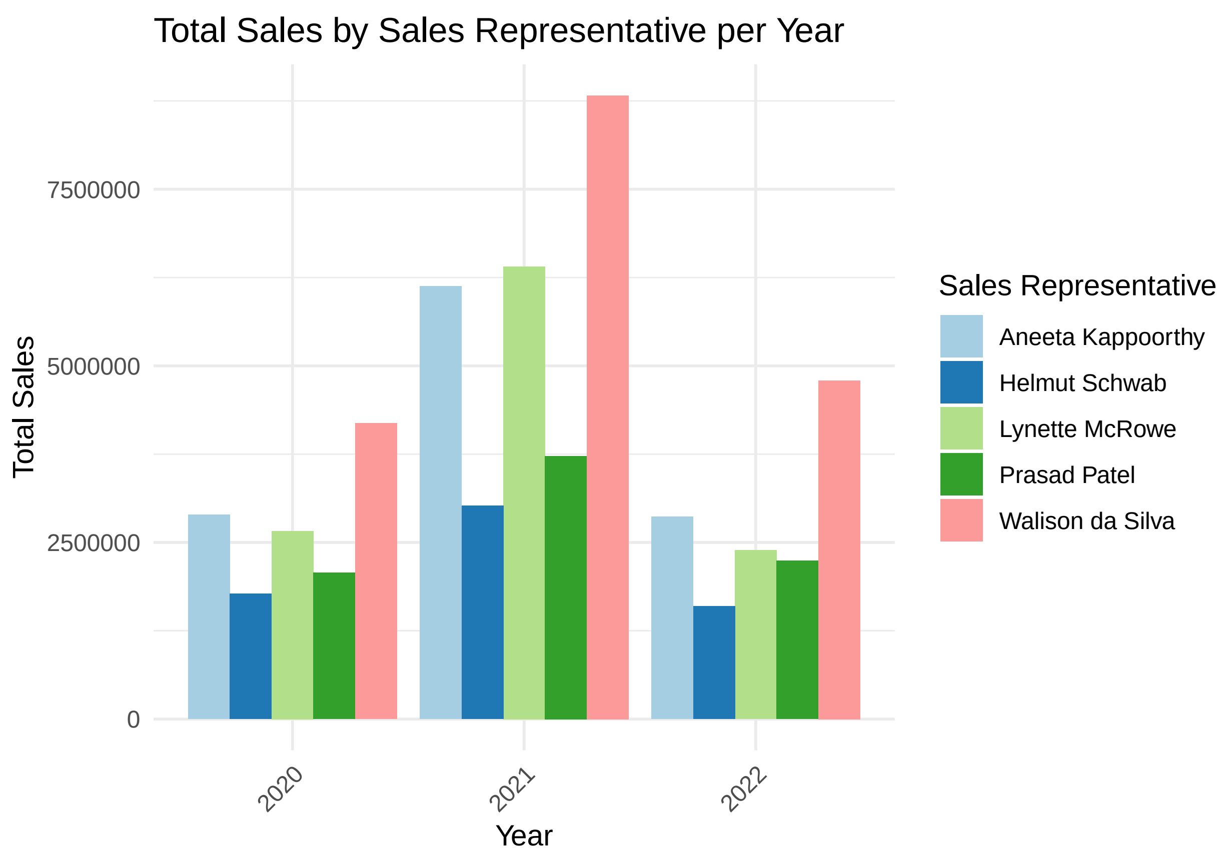
```
# Establish a MySQL database connection using RMySQL package
# Set database user, password, name, host, and port for connection parameters
db.user <- 'admin'
db.password <- 'practicum25200'
db.name <- 'practicum2'
db.host <- 'practicum2.cwtq29inp2km.us-east-2.rds.amazonaws.com'
db.port <- 3306
dbConnMySQL <-
  dbConnect(
    RMySQL::MySQL(),
    user = db.user,
    password = db.password,
    dbname = db.name,
    host = db.host,
    port = db.port
  )
```

```
# Analytical Query: Retrieve sales data for the top 5 performing representatives each year
# This query selects sales information for the top 5 representatives in each year based on their total
# and ranks them within each year. It retrieves the year, representative name, total sales, and sales r
# The query first aggregates sales data by year and representative, calculates the sales rank within ea
# and then filters the results to include only the top 5 representatives in each year.
# Finally, the results are sorted by year and sales rank.
analytical.Query.A <- "SELECT
    year,
    rep_name,
    total_sales,
    sales_rank
FROM (
    SELECT
        rf.year,
        CONCAT(rd.first_name, ' ', rd.last_name) AS rep_name,
        SUM(rf.total_sold) AS total_sales,
        RANK() OVER (PARTITION BY rf.year ORDER BY SUM(rf.total_sold) DESC) as sales_rank
    FROM rep_facts rf
    JOIN reps_dimension rd ON rf.rep_id = rd.rep_id
    GROUP BY rf.year, rd.rep_id, rd.first_name, rd.last_name
) as ranked_sales
WHERE ranked_sales.sales_rank <= 5
ORDER BY year, sales_rank;
"
```

```
# This line of code retrieves analytical sales data for a specific sales representative
# from a MySQL database using a predefined query (analytical.Query.A) and stores it
# in the 'analytical.sales.rep' variable. 'dbConnMySQL' represents the established database connection.
analytical.sales.rep <- dbGetQuery(dbConnMySQL, analytical.Query.A)
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```
# Create a ggplot object using the 'analytical.sales.rep' data frame.
# 'year' is plotted on the x-axis, 'total_sales' on the y-axis, and 'rep_name' determines the fill colo
ggplot(analytical.sales.rep, aes(x=factor(year), y=total_sales, fill=rep_name)) +

  # Add bars to the plot, using 'stat="identity"' to use the data as is, and 'position="dodge"' to dodg
  geom_bar(stat="identity", position="dodge") +

  # Customize the fill colors using the 'scale_fill_brewer' function with the 'Paired' palette
  scale_fill_brewer(palette="Paired") +

  # Add labels and title to the plot
  labs(title="Total Sales by Sales Representative per Year",
       x="Year",
       y="Total Sales",
       fill="Sales Representative") +

  # Use a minimal theme for the plot
  theme_minimal() +

  # Customize the appearance of x-axis labels with text rotation, vertical justification, and horizonta
  theme(axis.text.x = element_text(angle=45, vjust=1, hjust=1))
```

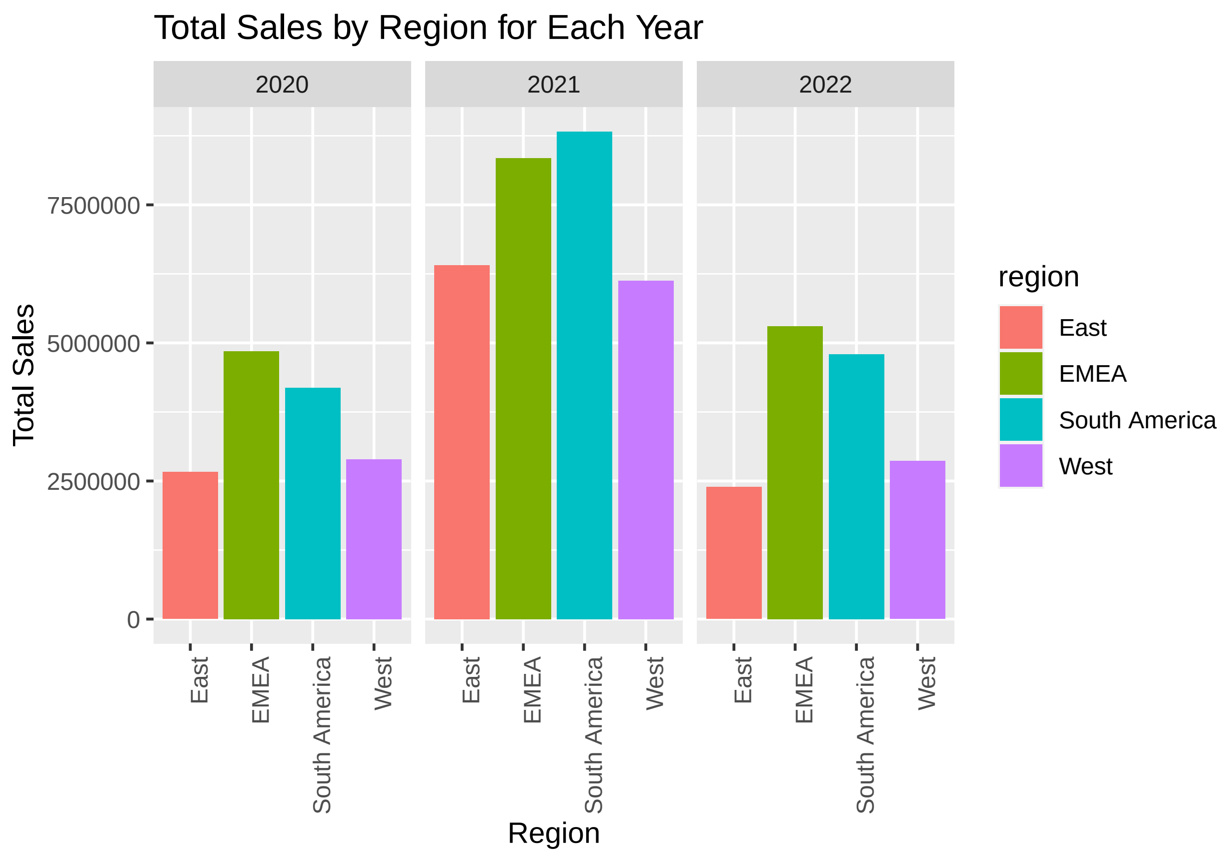


```
# Analytical Query to Calculate Total Sales by Year and Region
# This SQL query retrieves data from the sales_facts table, date_dimension table, and reps_dimension ta
# It calculates the total sales amount for each year and region, and sorts the results by year and tota
analytical.Query.B <- "SELECT
    dd.year,
    rd.territory AS region,
    SUM(sf.total_sales_amount) AS total_sales
FROM sales_facts sf
JOIN date_dimension dd ON sf.date_id = dd.date_id
JOIN reps_dimension rd ON sf.reps_key = rd.reps_key
GROUP BY dd.year, rd.territory
ORDER BY dd.year, total_sales DESC;
"
```

```
# Retrieve the total sold items per year per region
# using the dbGetQuery function to execute the SQL query
# specified in the analytical.Query.B and store the result
# in the analytical.TotalSold.PerYear.PerRegion variable.
analytical.TotalSold.PerYear.PerRegion <- dbGetQuery(dbConnMySQL, analytical.Query.B)
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```
# Create a ggplot object with data from analytical.TotalSold.PerYear.PerRegion
# Mapping 'region' to x-axis, 'total_sales' to y-axis, and filling by 'region'
ggplot(analytical.TotalSold.PerYear.PerRegion, aes(x = region, y = total_sales, fill = region)) +
  # Add a bar geometry layer, representing data as bars with 'stat = "identity"'
  geom_bar(stat = "identity") +
    # Create multiple facets (subplots) based on the 'year' variable
    facet_wrap(~ year) +

    # Customize the appearance of the x-axis text by rotating the labels 90 degrees
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +

    # Set the title and labels for the plot
    labs(title = "Total Sales by Region for Each Year", x = "Region", y = "Total Sales")
```



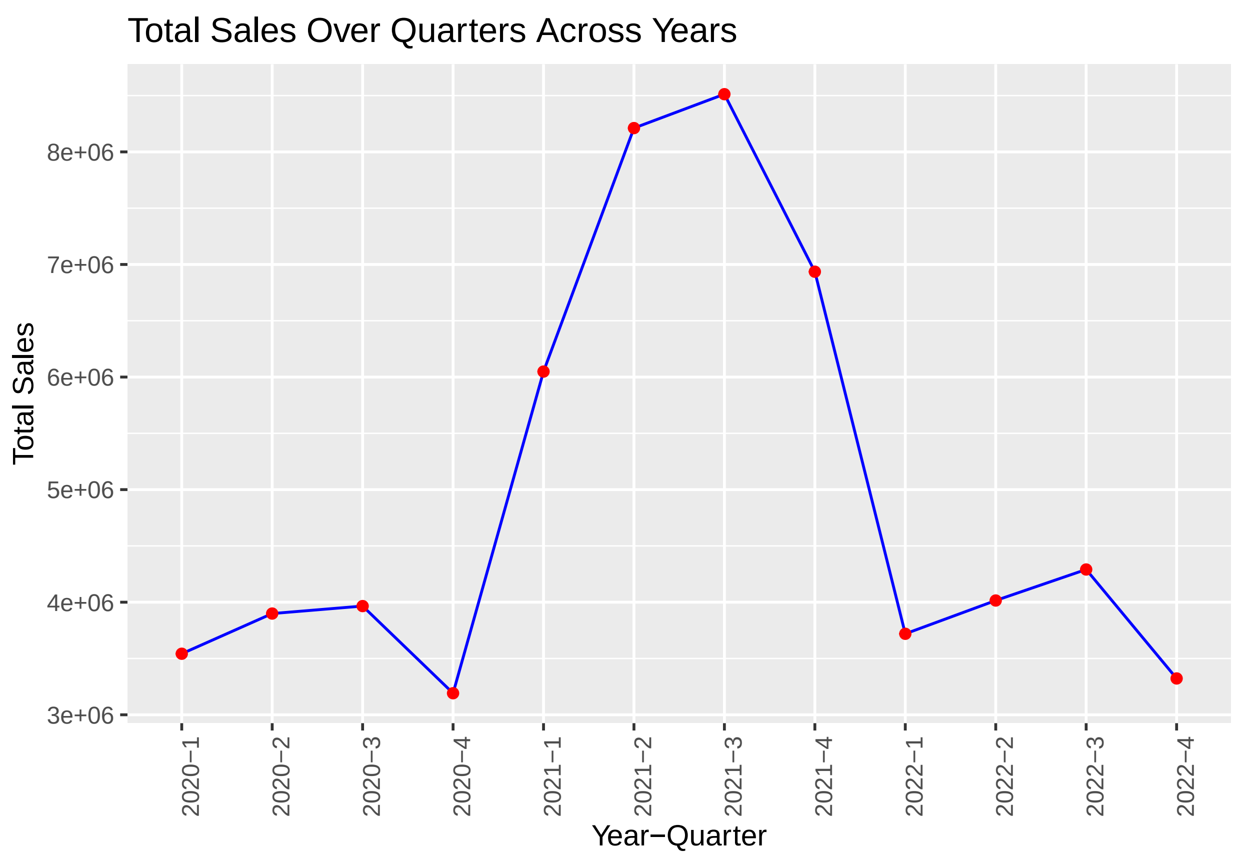
```
# The following SQL query is intended to retrieve total sales amounts
# aggregated by year and quarter from a sales facts table.
analytical.Query.C <- "SELECT
    dd.year,
    dd.quarter,
    SUM(sf.total_sales_amount) AS total_sales
FROM sales_facts sf
JOIN date_dimension dd ON sf.date_id = dd.date_id
GROUP BY dd.year, dd.quarter
ORDER BY dd.year, dd.quarter;
"
# We start by selecting the 'year' and 'quarter' columns from the 'date_dimension' table
# and calculating the sum of 'total_sales_amount' from the 'sales_facts' table.
# The result will be aliased as 'total_sales'.
```

```r
# Next, we perform a JOIN operation between 'sales_facts' (as 'sf') and 'date_dimension' (as 'dd')
# using the 'date_id' column as the joining condition.
# This helps us link sales data to corresponding dates.

# We proceed to GROUP the results by 'year' and 'quarter', which means
# we'll get the total sales amount for each combination of year and quarter.

# Finally, we ORDER the results by 'year' and 'quarter' to present the data in a structured manner.
# This ensures that the output will be sorted by year and then quarter.
```

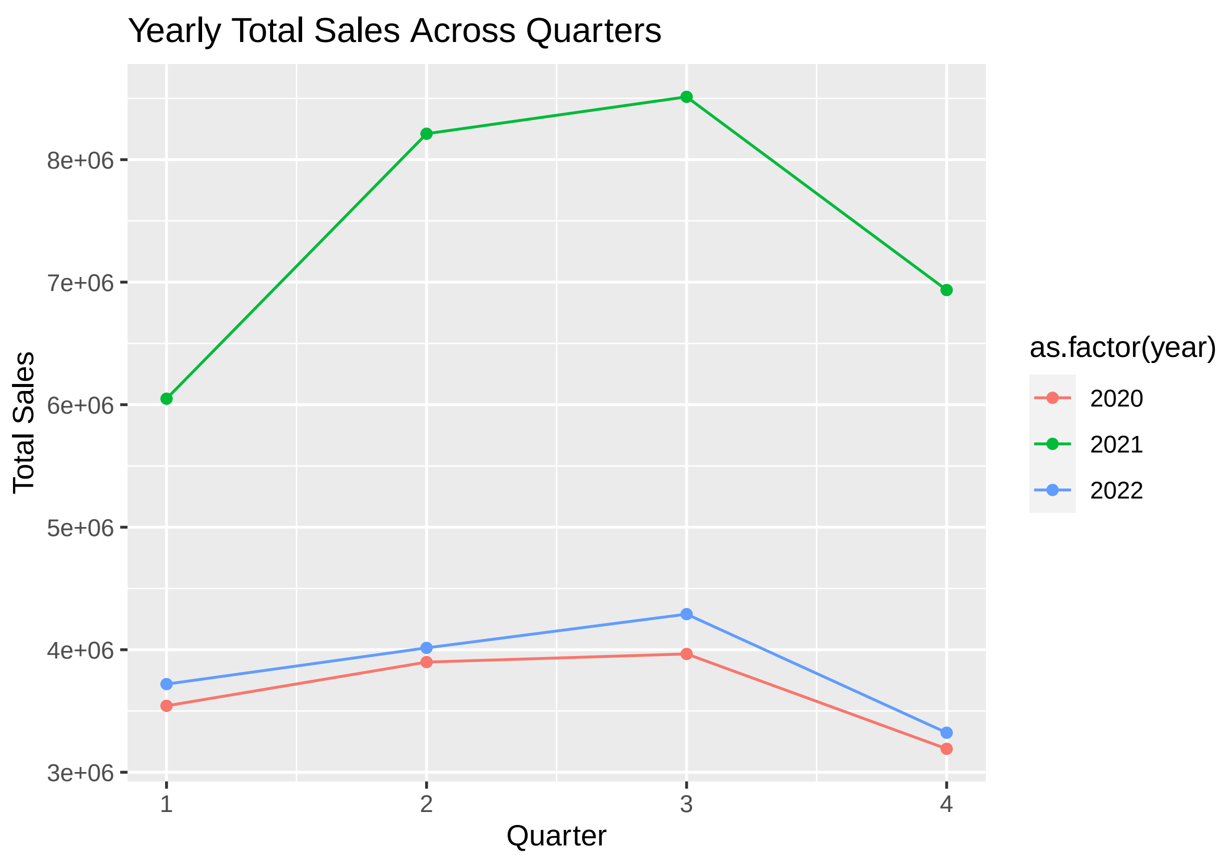
```r
# Define a variable 'analytical.Totalsold.Quarter' and assign the result of a database query to it.
analytical.Totalsold.Quarter <- dbGetQuery(dbConnMySQL, analytical.Query.C)
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```r
# This line creates a new column 'year_quarter' in the dataframe 'analytical.Totalsold.Quarter'
# by pasting together the 'year' and 'quarter' columns with a hyphen separator.
analytical.Totalsold.Quarter$year_quarter <- paste(analytical.Totalsold.Quarter$year, analytical.Totals

# This block of code generates a line plot using ggplot2 library.
# It specifies the data source as 'analytical.Totalsold.Quarter' and sets the aesthetics.
# The x-axis represents 'year_quarter', the y-axis represents 'total_sales', and a single group is defi
ggplot(analytical.Totalsold.Quarter, aes(x = year_quarter, y = total_sales, group = 1)) +

  # This line adds a blue line to the plot.
  geom_line(color = "blue") +

  # This line adds red points to the plot.
  geom_point(color = "red") +

  # This line customizes the appearance of the x-axis labels by rotating them 90 degrees and adjusting
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +

  # This line sets the plot title, x-axis label, and y-axis label.
  labs(title = "Total Sales Over Quarters Across Years", x = "Year-Quarter", y = "Total Sales")
```



```r
# Create a ggplot object using the 'analytical3' dataset
# Mapping 'quarter' to the x-axis and 'total_sales' to the y-axis
# Group the data by 'year' and color each line by 'year'
ggplot(analytical.Totalsold.Quarter, aes(x = quarter, y = total_sales, group = year, color = as.factor(y

    # Add a line plot layer to the ggplot
    geom_line() +

    # Add a point plot layer on top of the line plot
    geom_point() +

    # Set the title of the plot
    labs(title = "Yearly Total Sales Across Quarters", x = "Quarter", y = "Total Sales")
```


Yearly Total Sales Across Quarters
Total Sales
8e+06
7e+06
6e+06
5e+06
4e+06
3e+06
1
2
3
4
Quarter
as.factor(year)
2020
2021
2022