

**ANL503**

**End-of-Course Assessment - January Semester 2025**

**Data Wrangling**

**PI no. Y2510707**

**Name: Ng Chee Wee**

**Submission Date:12.04.2025**

Contents

[1a. Create Python Program to access CSV file and store it in table named ECA\_data\_raw 3](#_Toc194655418)

[1b. Create a new table named ECA\_data 5](#_Toc194655419)

[1c. ALTER TABLE 6](#_Toc194655420)

[1d. Create new table ECA\_summary 7](#_Toc194655421)

[1e. R program to read in eca\_summary MySQL table and store it as an R data.frame 8](#_Toc194655422)

[R code to create Visualization 8](#_Toc194655423)

[Visualization 9](#_Toc194655424)

[References 11](#_Toc194655425)

[Data Appendix 13](#_Toc194655426)

# 1a. Create Python Program to access CSV file and store it in table named ECA\_data\_raw

import pandas as pd

from sqlalchemy import create\_engine, URL

# Set up the sqlalchemy engine

url\_object = URL.create(

'mysql+pymysql',

username='root',

password='your\_pw',

host='localhost',

database='anl503\_eca')

engine = create\_engine(url\_object)

# Read in csv data into Pandas DataFrame

df = pd.read\_csv('ECA\_data\_raw.csv')

# Check variable type via 'type()' function

print(type(df))

# Check num rows/cols via 'shape' attribute

print(df.shape)

# Check col names via 'columns' attribute

print(df.columns)

# Check col datatype via 'dtypes' attribute

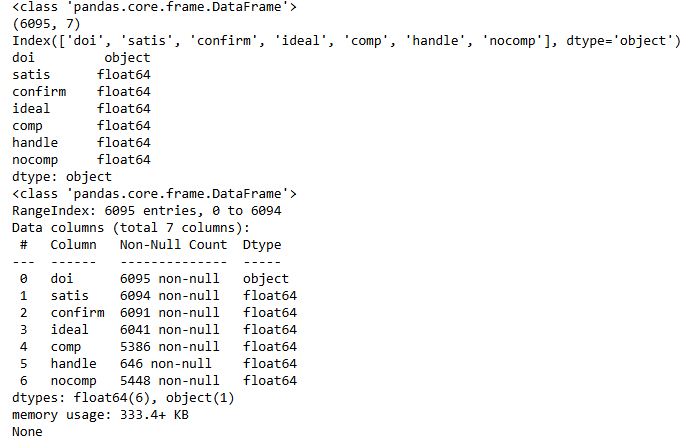
print(df.dtypes)

# Use 'info()' method to view summary info

print(df.info())

# Use 'to\_sql()' method to upload to MySQL

df.to\_sql('ECA\_data\_raw', engine, if\_exists='replace', index=False)



# 1b. Create a new table named ECA\_data

MySQL code:

-- Create new table ECA\_data based on eca\_data\_raw

DROP TABLE IF EXISTS ECA\_data;

CREATE TABLE ECA\_data AS

SELECT

CASE

-- When doi is already in YYYY-MM-DD format, leave it as is

WHEN doi REGEXP '^[0-9]{4}-[0-9]{2}-[0-9]{2}$' THEN doi

-- When doi is a number (days since Jan 1, 2024), convert to date

WHEN doi REGEXP '^[0-9]+$' THEN

DATE\_FORMAT(DATE\_ADD('2024-01-01', INTERVAL doi DAY), '%Y-%m-%d')

ELSE doi

END AS doi,

satis,

confirm,

ideal,

comp,

handle,

nocomp

FROM eca\_data\_raw;

# 1c. ALTER TABLE

MySQL code:

ALTER TABLE Eca\_data

MODIFY doi DATE NULL,

MODIFY satis TINYINT UNSIGNED NULL,

MODIFY confirm TINYINT UNSIGNED NULL,

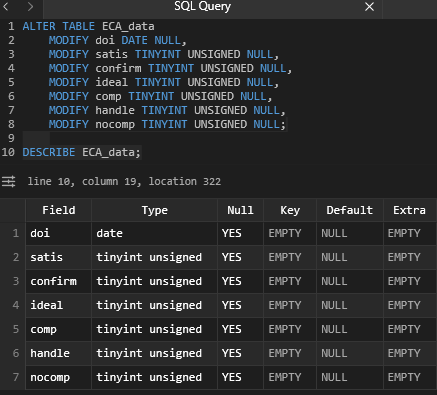
MODIFY ideal TINYINT UNSIGNED NULL,

MODIFY comp TINYINT UNSIGNED NULL,

MODIFY handle TINYINT UNSIGNED NULL,

MODIFY nocomp TINYINT UNSIGNED NULL;

DESCRIBE eca\_data;



For satis, confirm, ideal and handle, these four columns correspond to respondent’s rating on a scale of 1-10. Therefore, using tinyint is space efficient (1 byte) and UNSIGNED restrict it to a non-negative value with a range of 0 to 255.

For comp, it has an input value of either 0 or 1, hence, using tinyint is also justified for saving storage space.

For nocomp, it has an input value from 1 to 4, hence it is optimal to use tinyint as data type to achieve space efficiency.

# 1d. Create new table ECA\_summary

MySQL code:

DROP TABLE IF EXISTS ECA\_summary;

CREATE TABLE ECA\_summary AS

SELECT

MONTH(doi) AS mth,

ROUND(AVG(satis),2) AS satis,

ROUND(AVG(confirm),2) AS confirm,

ROUND(AVG(ideal),2) AS ideal,

ROUND(AVG(comp),2) AS comp,

ROUND(AVG(handle),2) AS handle,

CONCAT(ROUND((SUM(CASE WHEN nocomp = 1 THEN 1 ELSE 0 END) / COUNT(\*) \* 100), 2),'%') AS TNCR

FROM ECA\_data

GROUP BY mth

ORDER BY mth;



# 1e. R program to read in eca\_summary MySQL table and store it as an R data.frame

R code to read in ECA\_summary table and store it as an R data.frame and the Visualization.

rm(list = ls())

#install.packages("RMySQL")

library(RMySQL)

library(tidyverse)

# Establish a connect to MySQL

con = dbConnect (MySQL(), dbname = "anl503\_eca", username="root", password="mango4673")

query <- "SELECT \* FROM eca\_summary"

df <- dbGetQuery(con, query)

head(df)

# Close the database connection

dbDisconnect(con)

# R code to create Visualization

# Clean TNCR column before pivoting

df <- df %>%

mutate(

TNCR = as.numeric(str\_replace(TNCR, "%", "")), # replaces all occurrences of **%** in the **TNCR** column with an empty string (**""**) and the result converted to numeric value.

mth = as.integer(mth) # ensure month is numeric

)

# Reshape data from wide to long format

df\_long <- df %>%

pivot\_longer(cols = -mth, names\_to = "variable", values\_to = "value") #all columns except mth are pivoted

#Plot the visualization

p <- ggplot(df\_long, aes(x = mth, y = value)) +

geom\_line(linewidth = 1.0, color = "blue") + # Single color for simplicity

facet\_wrap(~ variable, scales = "free\_y", ncol = 2) + # Separate panels, free y-scales for better readability, arrange facet in 2 columns layout

scale\_x\_continuous(breaks = 1:12, labels = 1:12) +

labs(

title = "Monthly ECA Summary Metrics (2024)", # Chart title

x = "Month", # x-axis label

y = "Value" # y-axis label

) +

theme\_minimal() +

theme(

plot.title = element\_text(hjust = 0.5, size = 16), # Centered title

strip.text = element\_text(size = 12, face = "bold"), # Facet labels

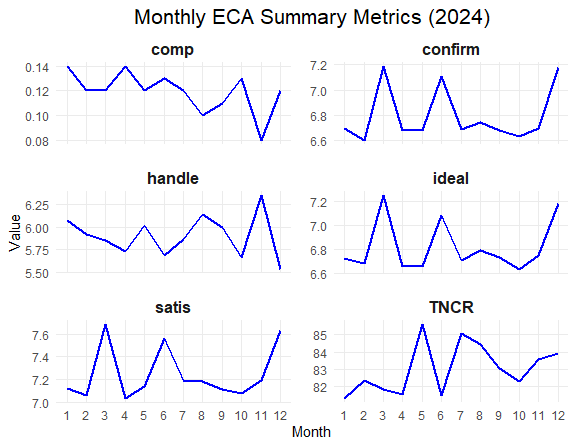
panel.grid.minor = element\_blank(), # Remove minor grid lines

panel.spacing = unit(1, "lines") # Space between facets

)

print(p)

# Visualization



To visualize multiple variables of different scales over time (months) from the ECA\_summary dataset, I evaluated options like a dual-axis line chart and a facet grid of line charts, as suggested by ChatGPT.

The advantages of a dual-axis line chart are it could plot the six variables in ECA\_summary in **one single chart** as separate lines in different colors and the x-axis representing the months.

The dual y-axes will take care of the different scale of the variables, TNCR in percentage and the other variables in scale of 0-10.

The disadvantages of a dual-axis line chart are that it could become cluttered to have six different variables in a single chart, especially if the lines overlapped. Furthermore, the dual y-axes will add to the clutter and confuse viewer due to the different scale.

On the other hand, the advantages of a facet grid of line chart are **clutter-free individual chart** for each variable and **reduce confusion** as it is easier to read and interpret as there is only one y-axis.

The disadvantages of a dual-axis line chart are that it is harder to compare the different variables since they are in sperate panels and less likely for user to spot correlation between the variables.

In conclusion, I have chosen a facet grid of line charts as it offers clutter-free and easy to understand visualization, especially without the dual axes to add to the chart clutter. In addition, the customer experience manager of the mall is likely to want to view the individual variable on its own and very rarely will require to compare those variables against each other.

# References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s/No. | AI tools | Prompt used | Output used in assignment | |
| 1 | ChatGPT | is there a function in MySQL that returns the integer representing the month of the year? | MONTH(doi) AS mth | |
| Full output | |  | | |
| 2. ChatGPT | | suggest visualizations which plot multiple metrics of different scale across months 1-12 | Facet Grid of Line Charts | |
|  | |  | | |
| 3. ChatGPT | | In MySQL, how to add a '%' to a calculated value in a column | | CONCAT(MySQL statement), ‘%’) AS TNCR |
|  | |  | | |

# Data Appendix

Table 1 shows the data dictionary describing the values of each variable.

