**Final Exam FE 513-B**

**Lorenzo Ausiello**

**Problem 1**

-- 1.1

-- Import given bank data into PostgreSQL database.

DROP TABLE IF EXISTS bank\_data;

SET datestyle = 'MDY';

CREATE TABLE bank\_data(

id INTEGER,

date DATE,

asset INTEGER,

liability INTEGER,

idx INTEGER

);

COPY bank\_data (id, date, asset, liability, idx)

FROM 'C:\Users\Public\bank\_data-1.csv'

DELIMITER ','

CSV HEADER;

SELECT \* FROM bank\_data;

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

-- 1.2

-- Create a primary key for the import table.

ALTER TABLE bank\_data

ADD PRIMARY KEY (idx);

-- 1.3

-- Find the highest asset observation for each bank

-- Sort the resulting table according to asset value.

-- Report the first 10 observations of output table.

SELECT bd.id, bd.date, bank\_max.max\_asset AS asset, bd.liability

FROM bank\_data bd

JOIN (

SELECT id, MAX(asset) AS max\_asset

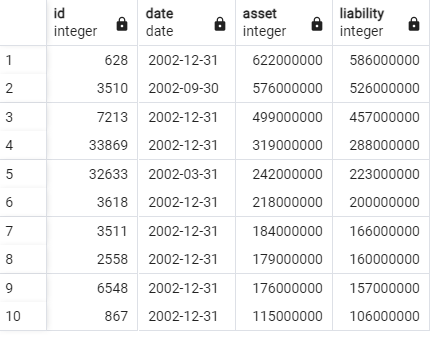
FROM bank\_data

GROUP BY id

) bank\_max ON bd.id = bank\_max.id AND bd.asset = bank\_max.max\_asset

ORDER BY bank\_max.max\_asset DESC

LIMIT 11;



-- 1.4

-- Show the query plan for question 1.3 using EXPLAIN tool

EXPLAIN

SELECT bd.id, bd.date, bank\_max.max\_asset AS asset, bd.liability

FROM bank\_data bd

JOIN (

SELECT id, MAX(asset) AS max\_asset

FROM bank\_data

GROUP BY id

) bank\_max ON bd.id = bank\_max.id AND bd.asset = bank\_max.max\_asset

ORDER BY bank\_max.max\_asset DESC

LIMIT 11;

Immagine che contiene testo, schermata, software, Icona del computer

Descrizione generata automaticamente

-- 1.5

-- Given the highest asset table from question 1.3, count how many observations are there for

-- each quarter.

SELECT

EXTRACT(QUARTER FROM bd.date) AS quarter,

COUNT(\*) AS observation\_count

FROM bank\_data bd

JOIN (

SELECT id, MAX(asset) AS max\_asset

FROM bank\_data

GROUP BY id

) bank\_max ON bd.id = bank\_max.id AND bd.asset = bank\_max.max\_asset

GROUP BY quarter

ORDER BY quarter;

Immagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente

-- 1.6

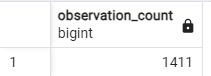
-- For the whole sample data, how many observations have asset value higher than 100,000 and

-- liability value smaller than 100,000.

SELECT COUNT(\*) AS observation\_count

FROM bank\_data

WHERE asset > 100000 AND liability < 100000;



-- 1.7

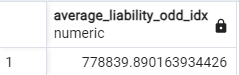
-- Each observation was given an ’idx’ number. Find the average liability of observation with

-- odd ’idx’ number.

SELECT AVG(liability) AS average\_liability\_odd\_idx

FROM bank\_data

WHERE idx % 2 <> 0;



-- 1.8

-- Find the average liability of observation with even ’idx’ number. What’s the difference between

-- these two average number.

SELECT AVG(liability) AS average\_liability\_even\_idx

FROM bank\_data

WHERE idx % 2 = 0;

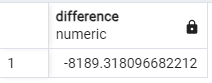
Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

SELECT

(SELECT AVG(liability) FROM bank\_data WHERE idx % 2 <> 0) -

(SELECT AVG(liability) FROM bank\_data WHERE idx % 2 = 0) AS difference;



--1.9

-- For each bank find all records with increased asset

-- Report the first 10 observation of output table.

SELECT b1.id, b1.date, b1.asset

FROM bank\_data b1

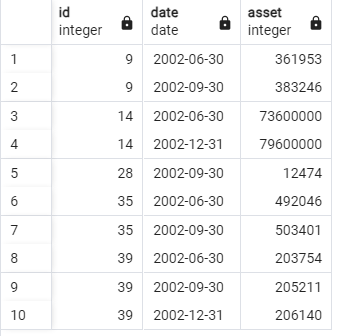
JOIN bank\_data b2 ON b1.id = b2.id

AND EXTRACT(QUARTER FROM b1.date) = EXTRACT(QUARTER FROM b2.date) + 1

AND b1.asset > b2.asset

ORDER BY b1.id, b1.date

LIMIT 10;



**Problem 2**

library(quantmod)

get\_stock\_data <- function(ticker, start\_time, end\_time, window\_size) {

# Download daily stock data using a given stock ticker for a given time period

stock\_data <- getSymbols(ticker, from = start\_time, to = end\_time, auto.assign = FALSE)

# Get the adjusted close price

adj\_close <- stock\_data[,6]

# Perform rolling window estimation for mean and standard deviation

mean\_estimates <- rollapply(adj\_close, width = window\_size, FUN = function(x) mean(x, na.rm = TRUE), by.column = FALSE)

std\_estimates <- rollapply(adj\_close, width = window\_size, FUN = function(x) sd(x, na.rm = TRUE), by.column = FALSE)

# Store the statistical results of into a dataframe

results <- data.frame(

Index = 1:length(mean\_estimates),

Mean = mean\_estimates,

Std\_Dev = std\_estimates

)

results <- na.omit(results)

# Plot this statistical dataframe using scatter plot

plot(results$Index, results$Mean, col = "blue", xlab = "Index", ylab = "Statistical Values",

ylim = c(min(c(results$Mean, results$Std\_Dev)), max(c(results$Mean, results$Std\_Dev))), main = "Rolling Window Statistics")

points(results$Index, results$Std\_Dev, col = "red")

legend("topright", legend = c("Mean", "Std Dev"), col = c("blue", "red"), pch = 1)

# Return the statistical dataframe

return(results)

}

# Test your function with suitable parameters

start\_date <- '2021-01-01'

end\_date <- '2023-01-01'

ticker <- "TSLA"

rolling\_window <- 20

stock\_stats <- get\_stock\_data(ticker, start\_date, end\_date, rolling\_window)

head(stock\_stats)

Immagine che contiene testo, schermata, numero, diagramma

Descrizione generata automaticamente

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

**Problem 3**

library("RPostgres")

# Make a connection to your local PostgreSQL database

con <- dbConnect(RPostgres::Postgres(),

dbname = "postgres",

host = "127.0.0.1",

port = 5432,

user = "postgres",

password = 'Culturismo99.')

# Query the PostgreSQL database via API to get the original bank data.

# Store the data into a dataframe.

query <- "SELECT \* FROM bank\_data;"

bank\_data <- dbGetQuery(con, query)

head(bank\_data)

Immagine che contiene testo, schermata, numero, linea

Descrizione generata automaticamente

# 3.3

# Calculate asset growth rate for each quarter and each bank.

# The result start from second quarter, since we don’t have all necessary data for first quarter calculation. Store the

# calculation result in a data frame.

library(dplyr)

bank\_data <- bank\_data %>%

arrange(id, date) %>%

group\_by(id) %>%

mutate(asset\_growth\_rate = ifelse(row\_number() == 1, NA,

(asset - lag(asset)) / lag(asset))) %>%

ungroup()

# Export the dataframe to the PostgreSQL database via API

dbWriteTable(con, "asset\_growth\_rates", bank\_data, row.names = FALSE, overwrite = TRUE)

dbDisconnect(con)

-- 3.4 (SQL)

SELECT \* FROM asset\_growth\_rates;

Immagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente