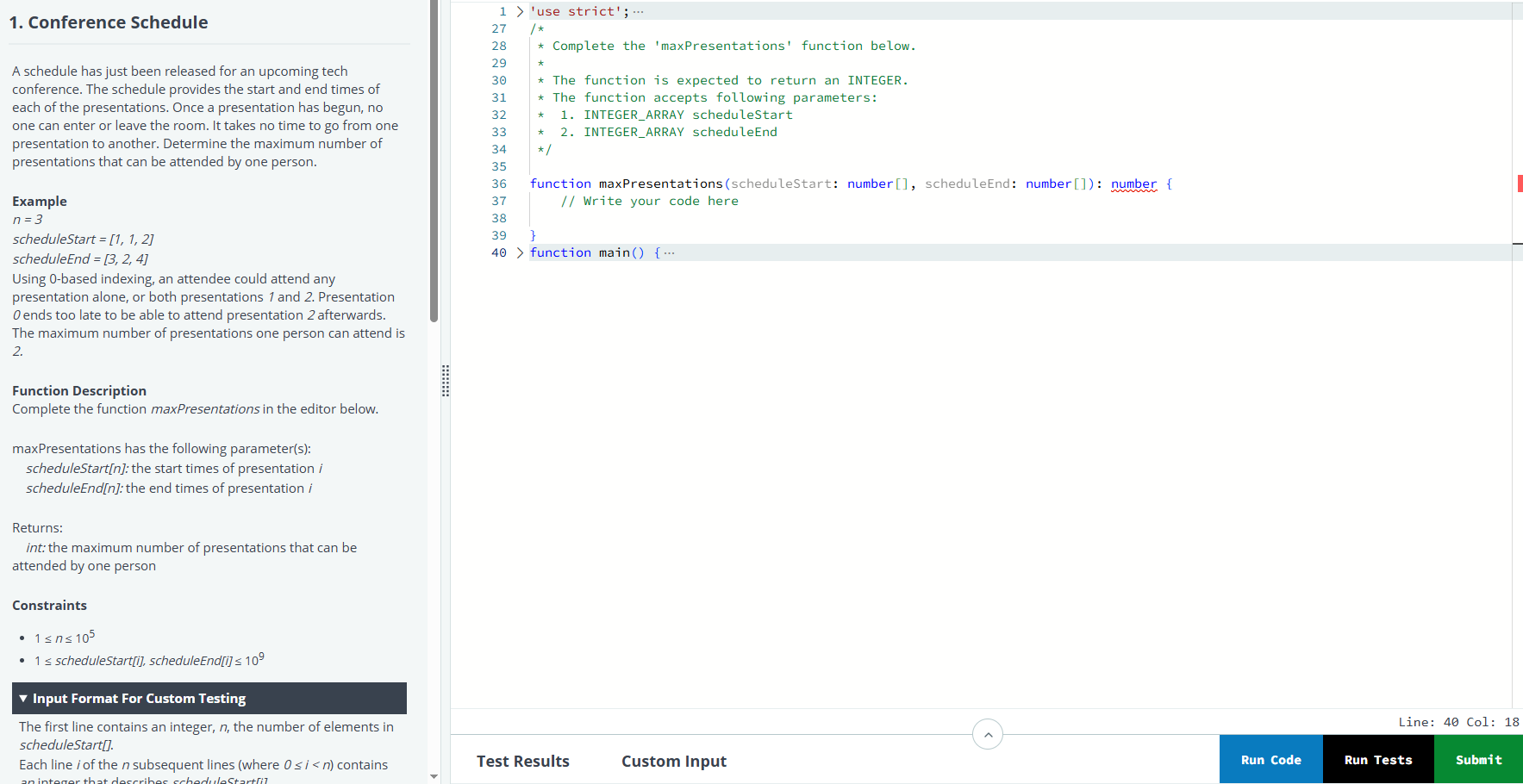
# Conference Schedule



function maxPresentations(scheduleStart: number[], scheduleEnd: number[]): number {

const n = scheduleStart.length;

// Pair up start and end times

const presentations = scheduleStart.map((start, i) => ({

start,

end: scheduleEnd[i]

}));

// Sort by end time

presentations.sort((a, b) => a.end - b.end);

let count = 0;

let lastEndTime = 0;

for (const presentation of presentations) {

if (presentation.start >= lastEndTime) {

count++;

lastEndTime = presentation.end;

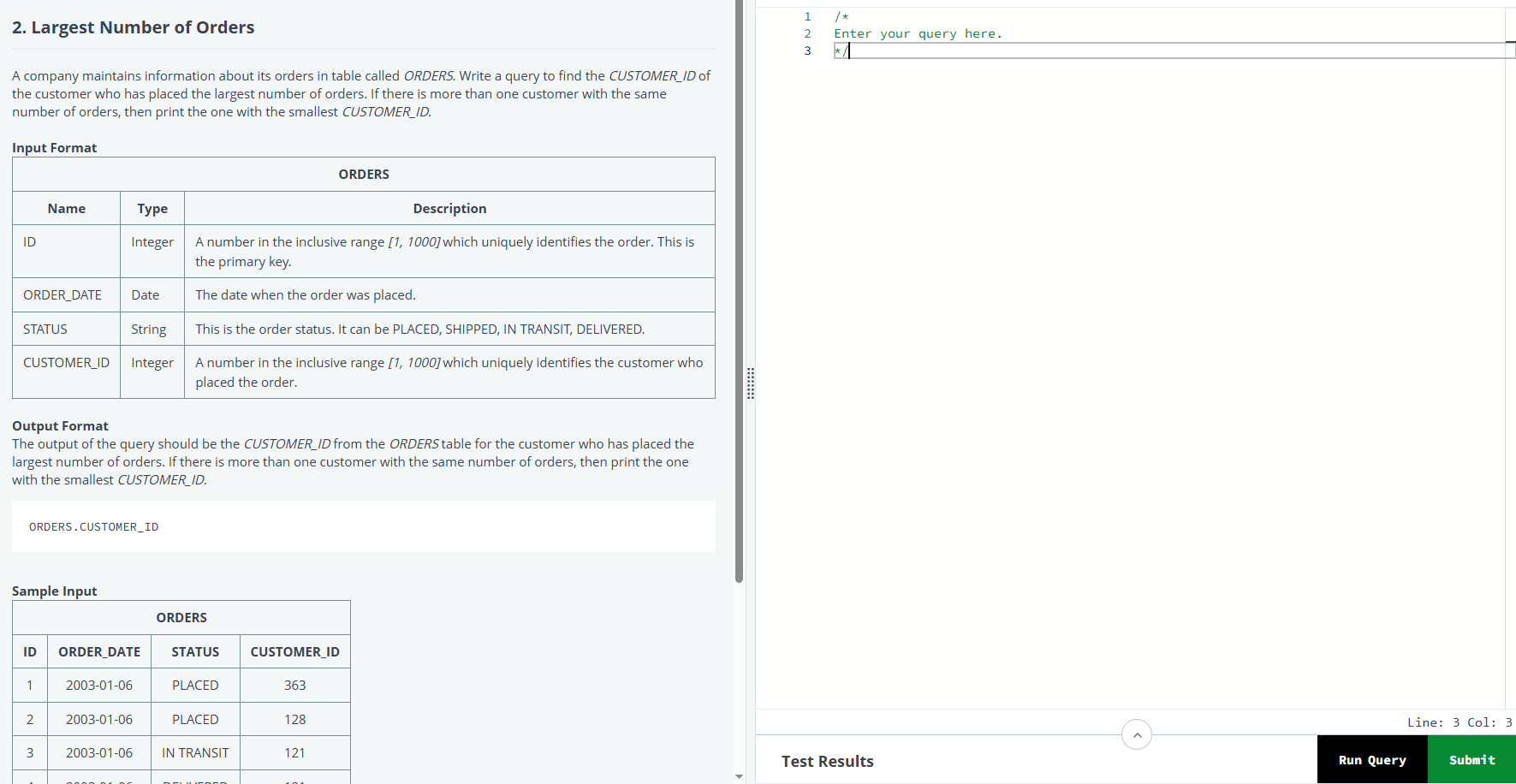
}

}

return count;

}

# Largest Number of Orders



SELECT CUSTOMER\_ID

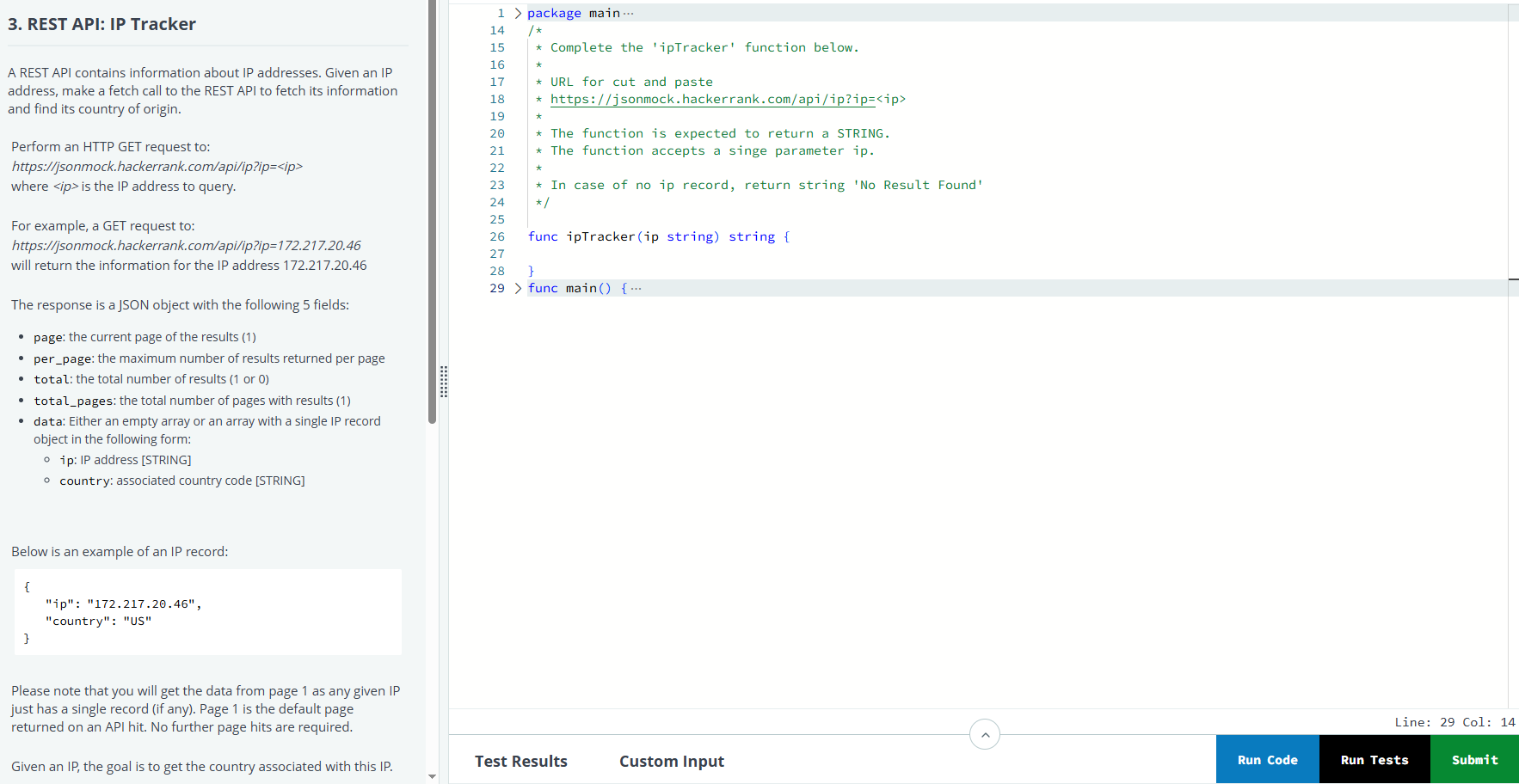
FROM ORDERS

GROUP BY CUSTOMER\_ID

ORDER BY COUNT(\*) DESC, CUSTOMER\_ID ASC

LIMIT 1;

# REST API: IP Tracker



package main

import (

"bufio"

"encoding/json"

"fmt"

"io"

"net/http"

"os"

"strings"

)

type IPResponse struct {

Page int `json:"page"`

PerPage int `json:"per\_page"`

Total int `json:"total"`

TotalPages int `json:"total\_pages"`

Data []struct {

IP string `json:"ip"`

Country string `json:"country"`

} `json:"data"`

}

func ipTracker(ip string) string {

url := fmt.Sprintf("https://jsonmock.hackerrank.com/api/ip?ip=%s", ip)

resp, err := http.Get(url)

if err != nil {

return "No Result Found"

}

defer resp.Body.Close()

var ipResponse IPResponse

if err := json.NewDecoder(resp.Body).Decode(&ipResponse); err != nil {

return "No Result Found"

}

if ipResponse.Total == 0 || len(ipResponse.Data) == 0 {

return "No Result Found"

}

return ipResponse.Data[0].Country

}

func main() {

reader := bufio.NewReaderSize(os.Stdin, 16\*1024\*1024)

stdout, err := os.Create(os.Getenv("OUTPUT\_PATH"))

checkError(err)

defer stdout.Close()

writer := bufio.NewWriterSize(stdout, 16\*1024\*1024)

ip := readLine(reader)

result := ipTracker(ip)

fmt.Fprintf(writer, "%s\n", result)

writer.Flush()

}

func readLine(reader \*bufio.Reader) string {

str, \_, err := reader.ReadLine()

if err == io.EOF {

return ""

}

return strings.TrimRight(string(str), "\r\n")

}

func checkError(err error) {

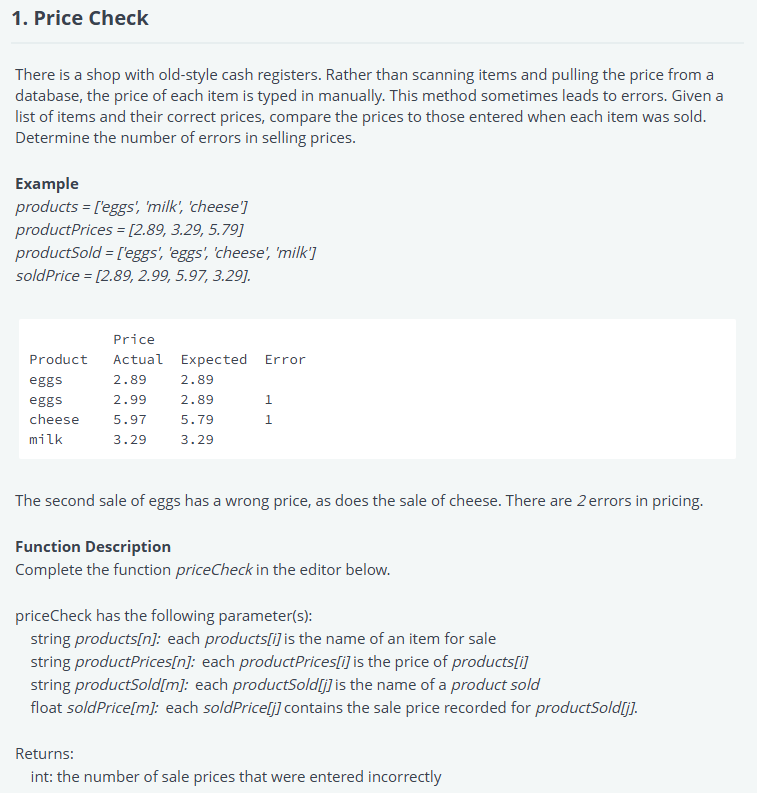
if err != nil {

panic(err)

}

}

# Price Check



#!/bin/python3

import math

import os

import random

import re

import sys

#

# Complete the 'priceCheck' function below.

#

# The function is expected to return an INTEGER.

# The function accepts following parameters:

#  1. STRING\_ARRAY products

#  2. FLOAT\_ARRAY productPrices

#  3. STRING\_ARRAY productSold

#  4. FLOAT\_ARRAY soldPrice

#

def priceCheck(products, productPrices, productSold, soldPrice):

    # Write your code here

    price\_map = dict(zip(products, productPrices))

    errors = 0

    # Compare each sold product's recorded price to the correct price

    for i in range(len(productSold)):

        if price\_map[productSold[i]] != soldPrice[i]:

            errors += 1

    return errors

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    products\_count = int(input().strip())

    products = []

    for \_ in range(products\_count):

        products\_item = input()

        products.append(products\_item)

    productPrices\_count = int(input().strip())

    productPrices = []

    for \_ in range(productPrices\_count):

        productPrices\_item = float(input().strip())

        productPrices.append(productPrices\_item)

    productSold\_count = int(input().strip())

    productSold = []

    for \_ in range(productSold\_count):

        productSold\_item = input()

        productSold.append(productSold\_item)

    soldPrice\_count = int(input().strip())

    soldPrice = []

    for \_ in range(soldPrice\_count):

        soldPrice\_item = float(input().strip())

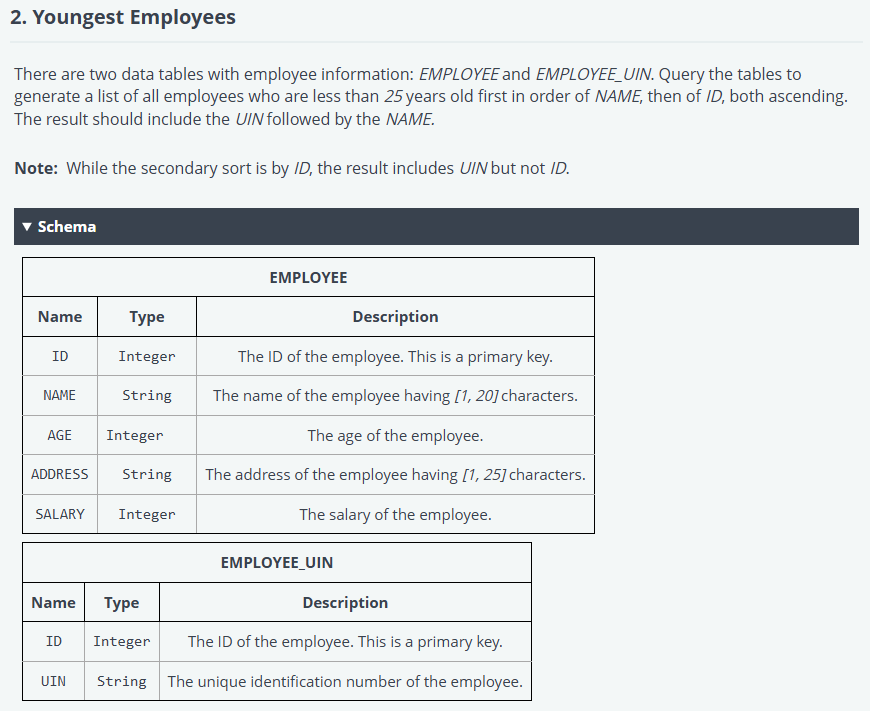
        soldPrice.append(soldPrice\_item)

    result = priceCheck(products, productPrices, productSold, soldPrice)

    fptr.write(str(result) + '\n')

    fptr.close()

# Youngest Employees



/\*

Enter your query here.

\*/

SELECT EU.UIN, E.NAME

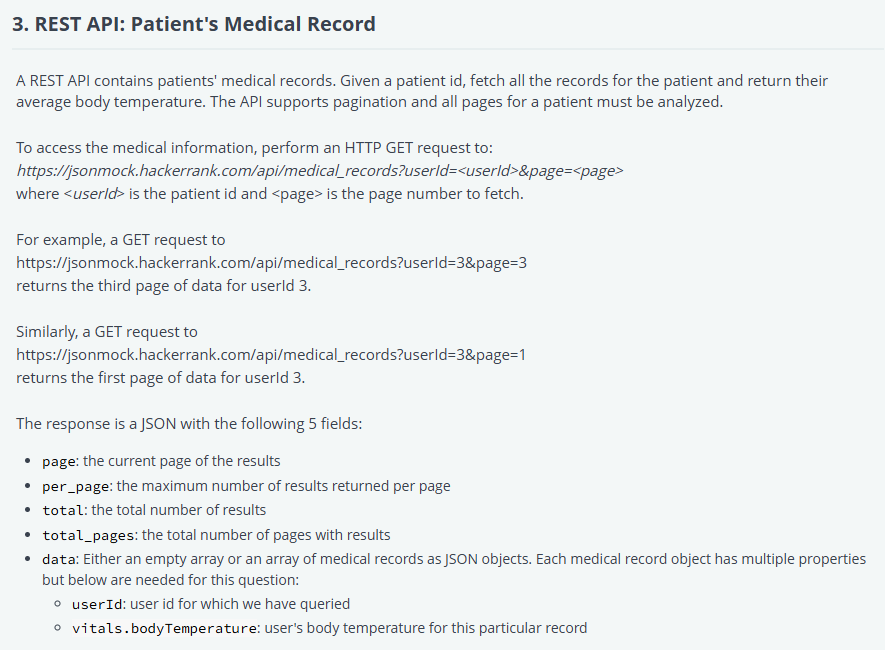
FROM EMPLOYEE E

JOIN EMPLOYEE\_UIN EU ON E.ID = EU.ID

WHERE E.AGE < 25

ORDER BY E.NAME ASC, E.ID ASC;

# REST API: Patient’s Medical Record



#!/bin/python3

import math

import os

import random

import re

import sys

import requests

#

# Complete the 'getAverageTemperatureForUser' function below.

#

# URL for cut and paste

# https://jsonmock.hackerrank.com/api/medical\_records?userId=<userId>&page=<page>

#

# The function is expected to return a String value.

# The function accepts a userId argumnent (Integer).

# In the case of an empty array result, return value '0'

#

def getAverageTemperatureForUser(userId):

    # Write your code here

    base\_url = "https://jsonmock.hackerrank.com/api/medical\_records"

    page = 1

    temperatures = []

    while True:

        response = requests.get(f"{base\_url}?userId={userId}&page={page}")

        data = response.json()

        # Exit early if no data

        if not data["data"]:

            break

        for record in data["data"]:

            temperatures.append(record["vitals"]["bodyTemperature"])

        if page >= data["total\_pages"]:

            break

        page += 1

    if not temperatures:

        return "0"

    else:

        avg\_temp = sum(temperatures) / len(temperatures)

        return "{:.1f}".format(avg\_temp)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    userId = int(input().strip())

    result = getAverageTemperatureForUser(userId)

    fptr.write(result + '\n')

    fptr.close()