**Assignment 4- Databases, Exceptions and Software testing**

**Jasmine Bajracharya[u3312590]**

**Question 1  
Solution:**

The code for solution 1 is given as :

def main():

    while True:

        file\_name = input('Please, enter the name of the file to be read: ')

        try:

            with open(file\_name, 'r') as file:

                content = file.read().strip()

                age = int(content)

                print(f'Age: {age}')

            break

        except FileNotFoundError:

            print(f'File {file\_name} not found.')

        except ValueError:

            print(f'File {file\_name} contains an invalid age.')

        except Exception as e:

            print(f'An unexpected error occurred: {e}')

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

    main()

The solution returns and invalid age message when the age is not in number and returns the age only when the age is a number:

The output from solution 1 is obtained as follows:

A screenshot of a computer

AI-generated content may be incorrect.

**Question 2  
Solution:**

The code for question 2 is as:  
def main():

    while True:

        file\_name = input('Please, enter the name of the file to be read: ')

        try:

            with open(file\_name, 'r') as file:

                content = file.read().strip()

                age = int(content)

                print(f'Salary: ${age}')

                print('Thank you for using our program!!')

            break

        except FileNotFoundError:

            print(f'File {file\_name} not found.')

            print('Try Again')

        except ValueError:

            print(f'File {file\_name} contains an invalid salary.')

            print('Try Again')

        except Exception as e:

            print(f'An unexpected error occurred: {e}')

            print('Try Again')

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

    main()

The output from the program is obtained as:  
  
A computer screen shot of a computer program

AI-generated content may be incorrect.

**Question 3  
Solution:**

Assert is used in solution 3 to prevent the use of if..else conditions. Assert checks if the value is True or not and AssertionError raises an exception when the value is False.

def main():

    while True:

        try:

            value = int(input("Enter an integer between 1 to 10: "))

            # Used assert to prevent the use of if..else statement as using ValueError requires using if..else statement.

            assert value != 0, "Opps, you entered zero."

            assert 1 <= value <= 10, "You did not enter a number between 1 and 10!!!"

            print(f"The Reciprocal of your number is {1 / value}")

            break

        except AssertionError as e:

            print(e)

            print('Please, try again.\n')

        except ValueError:

            print('You did not enter an integer!!!')

            print('Please, try again.\n')

        except Exception as e:

            print(f'An unexpected error occurred: {e}')

            print('Please, try again.\n')

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

    main()

The output obtained is:  
A computer screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Question 4  
Solution:**The following are the test cases for the conditions mentioned in the file Holiday.html

A white sheet with black text

AI-generated content may be incorrect.  
  
 **Question 5  
Solution:**The following are the test cases for the conditions mentioned in the file interview.html

A list of text on a white background

AI-generated content may be incorrect.

**Question 6  
Solution:**The following are the test cases for the conditions mentioned in the file referendum.html

A screenshot of a ballot

AI-generated content may be incorrect.

**Question 7**

**Solution 7a  
a) Find the total number of Managers and the sum of their salaries.**

SELECT COUNT(\*) AS manager\_count, SUM(salary) AS total\_salary

FROM Staff

WHERE oPosition = 'Manager';

-----------------------------------------------------------------

Primary Keys: none

Foreign Keys: none

Degree of Resulting View: 2

Cardinality of the Resulting View: 1

Content Overview based on the attributes required:

A screenshot of a computer

AI-generated content may be incorrect.

**Solution 7b**

b) Find the minimum, maximum, and average staff salary.

SELECT MIN(salary) as minimum\_salary, MAX(salary) as maximum\_salary, AVG(salary) as average\_salary

FROM Staff;

------------------------------------------------------------------------------------------

Primary Keys: none

Foreign Keys: none

Degree of Resulting View: 3

Cardinality of the Resulting View: 1

Content Overview based on the attributes required:

**A screenshot of a computer

AI-generated content may be incorrect.**

**Solution 7c**

c) For each branch office with more than one member of staff, find the number of staff working in each branch and the sum of their salaries.

SELECT Branch.branchNo, COUNT(\*) AS staff\_count, SUM(Staff.salary) AS total\_salary

FROM Staff

JOIN Branch ON Branch.branchNo = Staff.branchNo

GROUP BY Branch.branchNo

HAVING COUNT(\*) > 1;

--------------------------------------------------------------------------------

Primary Keys: Branch table- branchNo (branch.branchNo)

Foreign Keys: staff.branchNo references branch.branchNo

Degree of Resulting View: 3

Cardinality of the Resulting View: 2

Content Overview based on the attributes required:

A screenshot of a computer

AI-generated content may be incorrect.

**Solution 7d**

d) Construct a list of all cities where there is either a branch office or a property.

SELECT city

FROM Branch

UNION

SELECT city

FROM PropertyForRent;

------------------------------------------------------------------------------------

Primary Keys: none

Foreign Keys: none

Degree of Resulting View: 1

Cardinality of the Resulting View: 4

Content Overview based on the attributes required:

A screenshot of a computer screen

AI-generated content may be incorrect.

**Solution 7e**

e) Construct a list of all cities where there is both a branch office or a property.

SELECT city

FROM Branch

INTERSECT

SELECT city

FROM PropertyForRent;

------------------------------------------------------------------------------------

Primary Keys: none

Foreign Keys: none

Degree of Resulting View: 1

Cardinality of the Resulting View: 3

Content Overview based on the attributes required:

A screenshot of a computer

AI-generated content may be incorrect.

**Solution 7f**

f) Find the total number of Assistants, and the sum and average of their salaries.

SELECT COUNT(\*) AS number\_of\_assistants, SUM(salary) AS total\_salary, AVG(salary) AS average\_salry

FROM Staff

WHERE oPosition = 'Assistant';

------------------------------------------------------------------------------------

Primary Keys: none

Foreign Keys: none

Degree of Resulting View: 3

Cardinality of the Resulting View: 1

Content Overview based on the attributes required:

A close up of numbers

AI-generated content may be incorrect.

**Solution 7g**

g) For each branch office, list the staff numbers and names of staff who manage properties alongside the properties they manage.

SELECT

Branch.branchNo,

COALESCE(COUNT(DISTINCT Staff.staffNo), 0) AS staff\_count,

MAX(CASE WHEN Staff.oPosition = 'Manager' THEN Staff.fName || ' ' || Staff.lName END) AS manager\_name,

GROUP\_CONCAT(DISTINCT PropertyForRent.propertyNo) AS properties\_managed

FROM Branch

LEFT JOIN Staff ON Branch.branchNo = Staff.branchNo

LEFT JOIN PropertyForRent

ON Staff.staffNo = PropertyForRent.staffNo

AND PropertyForRent.branchNo = Branch.branchNo

GROUP BY Branch.branchNo;

------------------------------------------------------------------------------------

Primary Keys: branch.branchNo, staff.staffNo

Foreign Keys: staff.branchNo references (branch.branchNo) , propertyForRent.staffNo (references staff.staffNo), propertyForRent.branchNo (references branch.branchNo)

Degree of Resulting View: 4

Cardinality of the Resulting View: 5

A screenshot of a computer

AI-generated content may be incorrect.

**Question 8**

**Solution**

import tkinter as tk

from tkinter import messagebox

import os

def read\_file():

file\_name = e1.get().strip() # Get the filename from the entry box

base\_dir = os.getcwd() # Current working directory

full\_path = os.path.join(base\_dir, file\_name)

try:

with open(full\_path, 'r') as file:

content = file.read().strip()

age = int(content)

messagebox.showinfo("Success", f"Age: {age}")

except FileNotFoundError:

messagebox.showerror("Error", f"File '{file\_name}' not found in:\n{base\_dir}")

except ValueError:

messagebox.showerror("Error", f"File '{file\_name}' contains an invalid age.")

except Exception as e:

messagebox.showerror("Error", f"An unexpected error occurred:\n{e}")

def main():

global e1 # to make e1 accessible inside read\_file()

root = tk.Tk()

root.title("Solution of Question 8")

root.geometry("500x250")

root.configure(bg="#f0f8ff") #setting the background color to light blue

label = tk.Label(root, text="Enter the name of the file to be read:", bg="#f0f8ff" , fg= "Blue")

label.grid(row=1, column=0, padx=10, pady=5, sticky="w")

e1 = tk.Entry(root, width=50, justify="center") #using the entry box to take input form the user for the file to be read

e1.grid(row=2, column=0, padx=10, pady=5)

read\_btn = tk.Button(root, text="Read File", command=read\_file, bg="white", fg="black")

read\_btn.grid(row=3, column=0, pady=15)

root.mainloop()

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

main()

The output is obtained as:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.