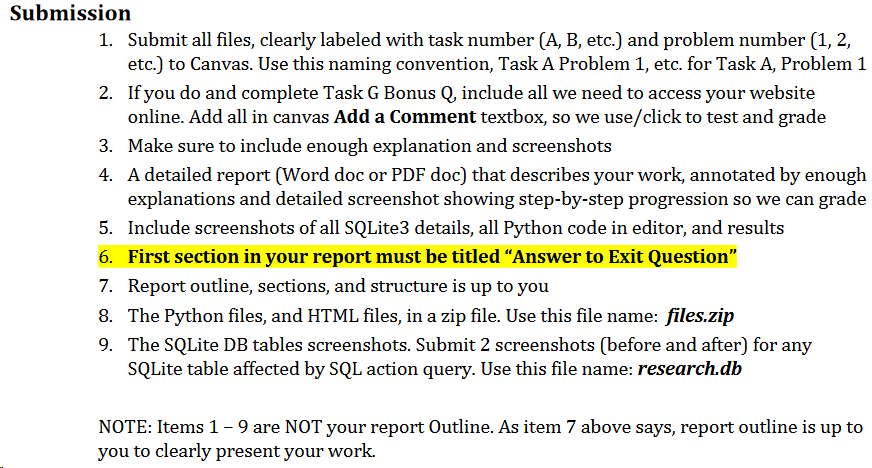
NORTHEASTERN UNIVERSIY  
MIE Department  
IE 3425 DB

# Research Assignment

# Submitted Contents

1. Main report, titled ‘research\_assignment.docx’
2. Supplementary description of website deployment (this document), titled ‘website\_deploy.docx’
3. A comment in the canvas submission noting that I have completed the Bonus and this link to access the live site: <https://tedyee114.github.io/websites/>
4. All files, including python, HTML, images, and reports in ‘files.zip’ -note that the .css file and .js file are that of the root website and located in the root folder
5. A copy of the retail\_app.db as it exists upon this project’s completion, titled ‘research.db’



Please note that the above screenshotted list of requested submission documents is unclear and best guess is made at which files are required to be submitted directly. However, all other files are directly available upon request or via GitHub codebase accessible through linked website. Lots of tedious HTML, JavaScript, and CSS code screenshots are omitted from this report on the grounds that their result is all visible in the live website and that all code is provided as an appendix and on the site itself.

# Answer to exit question:

I like that the project ended in webhosting of sites. I did that for my final project a few weeks ago and found that I really liked the web dev part of the course, and it turned out to be handy for this since I had already learned how to efficiently setup sites to add to my GitHub site, which I’m going to use for a portfolio. I didn’t learn anything new from the research, just took from the lecture slides and my previous assignments and in general in just felt like a much more frantic homework in place of the last 3 homework that caused unnecessary stress in anticipation as well as execution. I definitely do not like the 38hr “marathon” setup. Though the idea of teaching us to be able to do it all efficiently is nice, it clearly does not respect students’ time, especially at the end of the semester, assuming that we have a 42-hr period to devote solely to this project. On a different note, teaching subqueries earlier would be beneficial, as I found it made SQL much easier for me in early assignments.

My suggestions for this course in the future are to include one lecture on git and GitHub, a skill that I learned by myself recently and am finding extremely useful. This way, we could delve deeper into web dev (like JavaScript non-static sites), and we could hand in code much more easily. The TAs seem overly stressed and bothered by homework being handed in via word doc screenshots. Fillable homework templates, a submission site or a codebase submission like GitHub might make the process much smoother.

# Research Tasks

Included below are coding requirements, the code to complete them, screenshots, and decision-process notes where applicable. Note the different fonts and colors used for each part of each section.

Contents

[Research Assignment 1](#_Toc152618467)

[Answer to exit question: 1](#_Toc152618468)

[Research Tasks 1](#_Toc152618469)

[Notes 9](#_Toc152618470)

1. Using the course retail\_app database and SQL in VSCode
   1. Identify Employees with the Highest Salary in Each Department

select max(employee\_salary), employee.employee\_id, employee\_first\_name, employee\_last\_name, seller.seller\_department from employee left join seller on employee.employee\_id=seller.employee\_id group by seller\_department;

A screenshot of a computer

Description automatically generated

The code here starts with selecting data from the employee and seller tables via inner join, then groups the records by their department, and ends by selecting the record from each group with the maximum salary entry.

* 1. List all employees who have been with the company for more than 10 years, along with their employee ID, full name, hire date, and current age.

select employee\_id, employee\_first\_name, employee\_last\_name, employee\_hire\_date, strftime('%Y', employee\_hire\_date) as 'hire\_year', 2023-strftime('%Y', employee\_hire\_date) as years\_worked, employee\_dob, strftime('%Y', employee\_dob) as 'birth\_year', 2023-strftime('%Y', employee\_dob) as 'age' from employee where years\_worked>=10;

A screen shot of a computer

Description automatically generated

This query works by first selecting the basic person and date data from the employee table. Then, it breaks the year out of the YYYY-MM-DD formatted dates using the ‘STRFTIME’ SQLite function. The final step is to subtract the extracted year from 2023 to calculate, then display the elapsed time since birth and hire, respectively.

* 1. Identify the product with the highest inventory turnover rate, calculated as the total number of units sold divided by the inventory level defined in product table.

select max(100\*units\_sold/product\_inventory) from (select \*, 100\*units\_sold/product\_inventory from (select product.product\_id, product\_name, sum(invoice\_units) as 'units\_sold', product\_inventory from invoice inner join product on invoice.product\_id=product.product\_id group by product.product\_id));

select product\_name, max(100\*units\_sold/product\_inventory) from (select \*, 100\*units\_sold/product\_inventory from (select product.product\_id, product\_name, sum(invoice\_units) as 'units\_sold', product\_inventory from invoice inner join product on invoice.product\_id=product.product\_id group by product.product\_id));

A screenshot of a computer

Description automatically generated

The highest inventory turnover rate belongs to Uniball Pen. This query again starts with an inner join of product and invoice this time. The basic attributes are selected, then the total number of units sold is summed up per product sold. That number is multiplied by 100 (for percentage and to avoid integer division resulting in extreme rounding) and divided by the number in the inventory as instructed. The next command clearly lists the item by repeating the first command as a subquery and selecting only the record with the highest rate.

* 1. Find the information of the employee(s) who have the highest degree and have a salary >10000.

select employee.employee\_id, employee\_first\_name, employee\_last\_name, employee\_salary, degree\_name from employee left join employee\_degree on employee.employee\_id=employee\_degree.employee\_id left join degree on employee\_degree.degree\_id=degree.degree\_id;

select employee\_id, employee\_first\_name, employee\_last\_name, employee\_salary, degree\_name from (select employee.employee\_id, employee\_first\_name, employee\_last\_name, employee\_salary, degree\_name from employee left join employee\_degree on employee.employee\_id=employee\_degree.employee\_id left join degree on employee\_degree.degree\_id=degree.degree\_id) where employee\_salary>10000 and degree\_name='Doctorate';

A black screen with text

Description automatically generated

These two commands work very similarly in tandem like the last pair. The first selects from the employee, employee\_degree, and degree tables using two left joins to list all employees and their degrees as a subquery. From that subquery, it removes all employees with salaries less than $10,000, which happens to be none in this file. Then, the second query assumes that ‘Doctorate’ is the highest degree. It uses the first query as a subquery and selects all from it that have the degree ‘Doctorate’, which happens to only be one person. If there were none, a subsequent query would be executed searching for ‘Masters of Science’ and ‘Masters of Business Administration’ assuming they are both the next highest degree. Admittedly, this process is not scalable, but given the small number of possible degrees and the lack of defined order, it suffices.

1. Using the course retail\_app DB, create the following multi-level nav website shown below: home page (level 1) has 2 links: customers, products > click one link to go to level 2 (say customers) > click customers ink to go to page with 1 link (level 3): Add Customer. When you click Add Customer, it goes to a page with form to add customer to retail\_app DB. Similarly, when you click Get Product link, it goes to page with form to get product with product\_id as input. Website specs are:
   1. You must use at least 2 images in Get Product webpage.

3 images are used: the logo, the storefront, and the cartoon.

* 1. 2. All pages use one CSS file for formatting

All 3 pages reference the <https://tedyee114.github.io/websites/theme.css> file which keeps this site consistent with the main site’s theme.

* 1. 3. Each page has a link, named Home > when user clicks it, they go back to Home page. This way, the user does not have to hit browser back button to navigate website.

All sites have the same duplicated top navigation bar (also duplicated are the footer and root site homepage option). The repeated code for the navigation bar, and footer, however, are unnecessary. They are instead replaced by a snippet of JavaScript code that imports from separate files containing the elements. This eliminates the need to update each individually for thematic changes. Unfortunately, with the external <script> referencing, the homepage box and the importing script cannot be removed and must be manually duplicated (until another solution has the funding to be researched).

* 1. Use Flask and Python to create the website.

The live site can be accessed here: <https://tedyee114.github.io/websites/>. Screenshots of the two functioning webpages are included below. However, other than the specifically main python code, snippets and screenshots of all HTML renderings are left to readers discretion for viewing at the above hyperlink in the ‘websites’ repository and ‘research\_assignment’ subfolder. HTML code quality should be evaluable via the live site’s renderings and via the attached .zip file. Note that the project entails:

* 3 webpages with 1 HTML file per page, each containing a snippet of <script>JavaScript</script> to import the navbar and footer.
* 2 HTML files for imported navigation bars and footers
* 1 Python file running the site’s connection to the database on local machines (provided below)
* 1 JavaScript file in the root directory of the repository that controls the rotating element in both the research\_assignment project and the root website
* 1 CSS file in the root directory of the repository that controls both the research\_assignment project and the root website
* Static content

#region 0: import libraries and prepare environment

from flask import Flask, render\_template, request

import sqlite3

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import io

import base64

import pandas as pd

import pathlib

app = Flask(\_\_name\_\_)

#endregion

#region 1: homepage##############################################################################################

@app.route('/')

@app.route('/homepage')

def homepage():

    return render\_template('homepage.html')

#endregion

#region 2: customers##############################################################################################

@app.route('/add\_customer', methods=['GET', 'POST'])

def add\_customer():

    if request.method == 'POST':

        # read the form data

        customer\_first\_name = request.form.get('customer\_first\_name')

        customer\_last\_name = request.form.get('customer\_last\_name')

        customer\_gender = request.form.get('customer\_gender')

        customer\_dob = request.form.get('customer\_dob')

        db\_file = 'C:/Users/tedye/Desktop/db\_course/retail\_app.db'  #database connection

        conn = sqlite3.connect(db\_file)

        cursor = conn.cursor()

        # retrieve the maximum customer\_id from the customer table

        cursor.execute("SELECT MAX(customer\_id) FROM customer")

        max\_customer\_id = cursor.fetchone()[0]

        # if the table is empty, start with customer\_id = 1

        if max\_customer\_id is None:

            customer\_id = 1

        else:

            customer\_id = max\_customer\_id + 1

        # insert a new record into the customer table

        cursor.execute("INSERT INTO customer (customer\_id, customer\_first\_name,\

        customer\_last\_name, customer\_gender, customer\_dob)\

        VALUES (?, ?, ?, ?, ?)",\

        (customer\_id, customer\_first\_name, customer\_last\_name,\

        customer\_gender, customer\_dob))

        # commit the transaction

        conn.commit()

        # close the database connection

        conn.close()

        # return a response to the user

        return ('<h1>Success</h1>The customer record has been \

        successfully added to the database.')

    else:

        # Render the form

        return render\_template('add\_customer.html')

#endregion

#region 3: products##############################################################################################

@app.route('/get\_product', methods=['GET', 'POST'])

def get\_product():

    if request.method == 'POST':

        product\_id = request.form['product\_id']     #read form data

        db\_file = 'C:/Users/tedye/Desktop/db\_course/retail\_app.db'  #database connection

        conn = sqlite3.connect(db\_file)

        cursor = conn.cursor()

        cursor.execute("SELECT \* FROM product WHERE product\_id=?", (product\_id,))

A screenshot of a computer

Description automatically generated        output = cursor.fetchall()

        cursor.close()

        conn.close()

        if output is None:

            message = 'product with id {} does not exist.'.format(product\_id)

            return render\_template('get\_product.html', message=message)

        else:

            return render\_template('get\_product.html', output=output)

    else:

        return render\_template('get\_product.html')

#endregion

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

    app.run(debug=True)

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

* 1. A diagram of a product

     Description automatically generatedYou must create the website manually. Using any web development tools result in ZERO grade

Following the above sitemap, the site was created manually, using only snippets from previous assignments and researched methods. The unusual design of the code should provide sufficient evidence as such.

1. Do Problems F and G in Lab 19 in Week 11 on canvas. For G, use Python to delete the table. Prove/verify that the table has been deleted.

**F.** Import the ‘consumption.csv’ dataset file into retail\_app database. What food category is consumed the most across all countries per person per year? What food category is consumed the least across all countries? Verify SQL result for the top country using Excel.

create table consumption(

country text not null,

food\_category text,

consumption int,

c02\_emmission int);

.import --csv --skip 1 consumption.csv consumption

select food\_category, sum(consumption) as total from consumption group by food\_category;

select food\_category, max(total)from (select food\_category, sum(consumption) as total from consumption group by food\_category);

select food\_category, min(total)from (select food\_category, sum(consumption) as total from consumption group by food\_category);

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The greatest category is milk products, and the least is soybeans. This query involves the schema creation and importing of a .csv file, followed by a query to create a new column. The new column simply sums the ‘consumption’ column within each type of food. The second shown query uses the first query as a subquery and selects the one with the greatest sum, and the third does the same with the smallest sum. As can be seen in the excel sheet graph, dairy products and soybeans are the largest and smallest, respectively, sums.

**G.** Delete the consumption table from retail\_app DB.

import csv

import sqlite3

conn = sqlite3.connect('C:\\Users\\tedye\\Desktop\\db\_course\\retail\_app.db')              # Establish a connection to the SQLite database

cursor = conn.cursor()                                  # Create a cursor object to execute SQL queries

cursor.execute('drop TABLE consumption;')  # Create a table in the database to store the CSV data

conn.commit()                                                  # Commit the changes to the database

cursor.close()

conn.close()

A screenshot of a computer program

Description automatically generated

In this screenshot, note that the python script is fairly simple, operates by connecting to the database via command line and SQLite3 and then carries out the same ‘DROP’ command as would be directly used in SQLite3. There may be more complex ways to achieve the same output, but the shown method was chosen for its simplicity. In the marked command line view of the tables in the .db file, note that the consumption table was dropped without any direct command inputs and that the change had to come from the Python script. It is useful to note that after deleting, the snippet attached in the last task is useful to save as it can recreate the table without issues.

1. The Soccer World Cup has two groups. Each group has 4 teams: Team 1, Team 2, Team 3, Team 4. The total goals each team scored during the World Cup is:  
   Group 1: 20, 35, 25, 30

Group 2: 15, 28, 22, 18

Use Python to display the goals of each team in each group in one bar chart. Use Teams as the label for chart horizontal axis, Total Goals as the label for chart vertical axis. Also, add chart title, call it Comparing Goals. Also, add legend to chart to be able to read it.

A screenshot of a graph

Description automatically generated

The task instructions are unclear about the group numbers, so it is assumed that teams in group 1 are numbered 1-4 and 5-8 in group 2. The legend information is not specified, so the groups are displayed in it. Note that in this graph, the Group 1 and Group 2 teams are graphed via 2 separate commands but are overlayed so that they appear on the same graph. This process allows for different colors between the two groups.

1. Do Problems 5 and 11 in Lab 21, Week13 on canvas. Submit screenshots of employee table before and after commit and the SQL code used.
   1. The seller\_department attribute from the seller table contains the location of the seller. Write a subquery to find all employees who at a particular location. for example all employees who work at ‘scranton’

select \* from (select employee.employee\_id, employee\_first\_name, employee\_last\_name, seller\_department from employee left join seller on employee.employee\_id=seller.employee\_id) where seller\_department='Scranton';

A screen shot of a computer

Description automatically generated

This task is simple a query. As such, screenshots before and after committing are not applicable as nothing is changed. The query follows the logic of acquiring employee data including department via an inner join subquery, and then selecting from it only records with a given location, in this case ‘Scranton’ for the seller\_department.

* 1. Add an employee to the employee table. Make sure to commit the transaction.

insert into employee values ((select max(employee\_id)+1 from employee), 'first', 'last', 'H', '0000-12-25', '2000-01-01', '9999');

A screen shot of a computer program

Description automatically generated

The nuance in this task is that a subquery is required to extract the highest existing employee\_id and create the next consecutive, unique id. Otherwise, the command is quite simple and is the same as is used in task B with the website.

1. Do Problems 2 and 3 in Lab 22, Week13 on canvas. Submit screenshots of employee table before and after commit and the SQL code used. Also submit screenshot of the employee\_log table.
2. Recreate the view from example 2 in the lecture slides, but this time add new columns for 2 more currencies, Australian Dollar, and New Zealand Dollar. Then order the records by their price in US Dollar.

create view foreign\_currency\_prices (product\_description, us\_dollar\_price, canadian\_dollar\_price, euro\_price, great\_britain\_pound\_price, australian\_dollar, nz\_dollar) as select product.product\_description, product.product\_price, (product.product\_price\*1.26), (product.product\_price\*.93), (product.product\_price\*.8), (product.product\_price\*.75), (product.product\_price\*.73) from product;

select \* from foreign\_currency\_prices;

A screen shot of a computer

Description automatically generated

Again, this task is simple a query. As such, screenshots before and after committing are not applicable as nothing is changed. The query follows the logic of a query selecting two columns and more with derived attributes. The query is ‘saved’ as a view for later recall, as seen byt eh command immediately preceding the printout above. The saved ‘view’ can be seen in the tables list via the ‘.tables’ command.

1. Create a new table called employee\_log. This table should contain 2 attributes, employee\_id (same as employee table) and log\_time (current timestamp). The purpose of this table is to record the time when a new employee is added to the employee table. It is a means of record keeping. Create a trigger that inserts the employee\_id, and the  
   timestamp in the employee\_log\_time table every time a new record is inserted into the employee table.

CREATE TABLE employee\_log(

employee\_id int not null,

log\_time\_GMT,

foreign key(employee\_id) references employee(employee\_id));

A computer screen shot of white text

Description automatically generated

CREATE TRIGGER employee\_changelog

after insert on employee

begin

insert into employee\_log(employee\_id, log\_time\_GMT) values(new.employee\_id, time());

end;

A computer screen with white text

Description automatically generated

This task is straightforward, it requires only a trigger to add a record to a newly created table upon insertion into the employee table. Note that the timestamps are in Greenwich Mean Time (GMT), which may or may not be the user’s current time zone. Adjustments could be made using the time zone settings or manual, hard-coded changes, but such detail is not necessary and is omitted for brevity.

1. BONUS Q: (5% of your final points you score): Deploy the website you created in Task B on the web, so anyone can access it, like amazon.com. submit a separate detailed document in Word to canvas that describes the steps/instructions with clear screenshots. Use this name for the doc: website deploy NOTE: Do NOT attempt this question unless you are completely finished b/c there is NO partial credit.

The live site can be accessed here: <https://tedyee114.github.io/websites/>. See attached document titled ‘website\_deploy.docx’ for more details.

# Notes

Please note that all code can be found at the link above, by clicking on the redirect to the GitHub repository upon site home page landing. It is also recommended that the code and the rendered HTML pages be viewed there, as it is a superior experience to screenshots. All screenshots included are using SQLite3 with ‘.headers on’ and ‘.mode box.’