Tashyra Adams

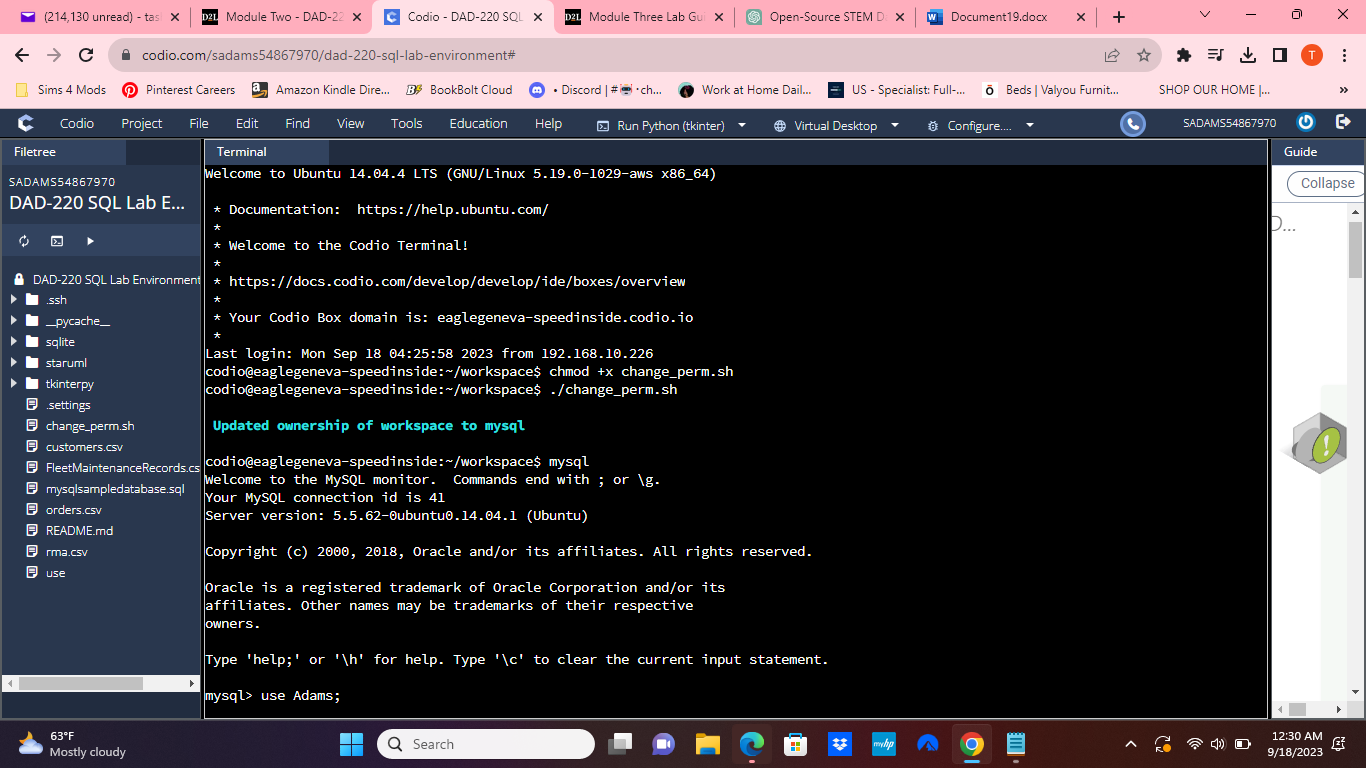
Linda Wilson

DAD-220

18 September 2023

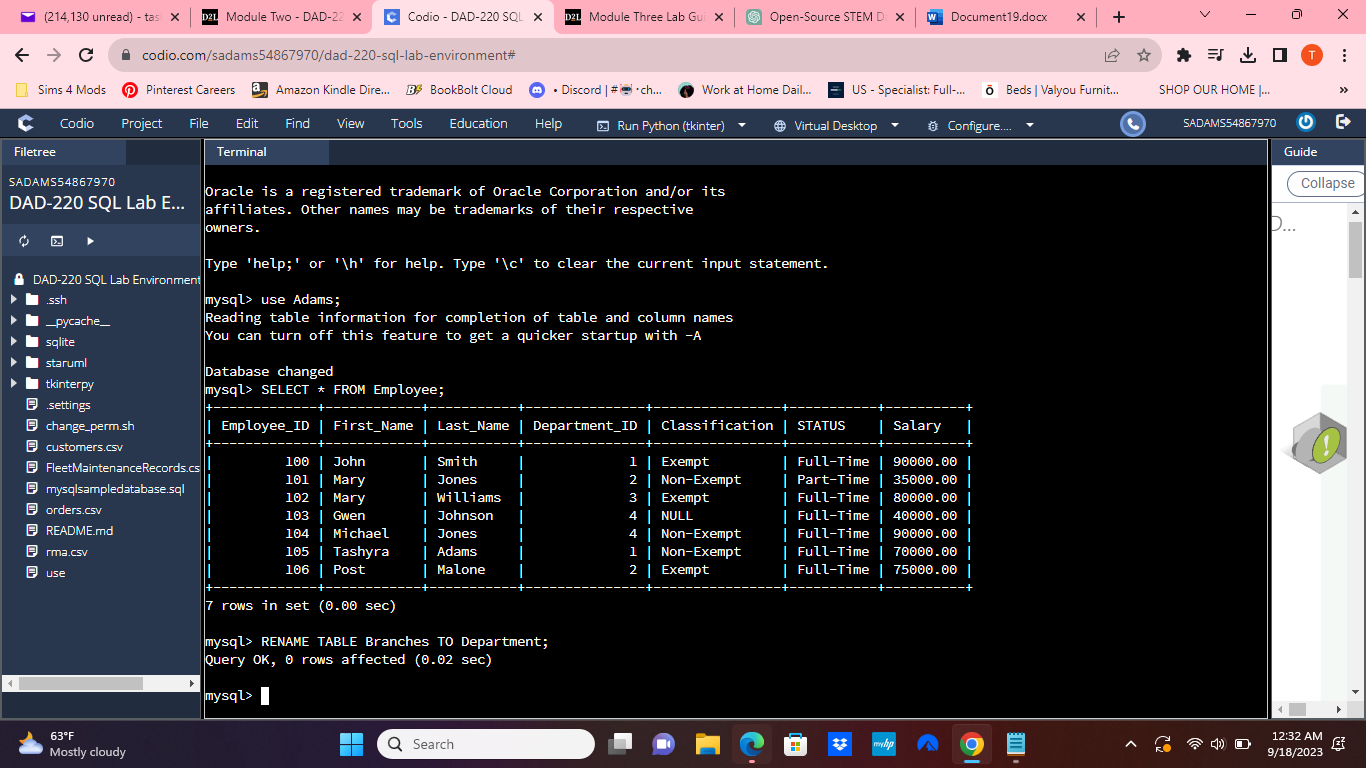
Before you begin, type the following commands prior to typing MySQL to set file permissions. This will allow you to perform the file output creation:

* chmod +x change\_perm.sh
* Press **Enter**.
* ./change\_perm.sh
* Then, enter MySQL and reconnect to the employee information you entered in the previous lab.
* Write a SELECT statement for the Employee table to check that you’ve reconnected to the right information.



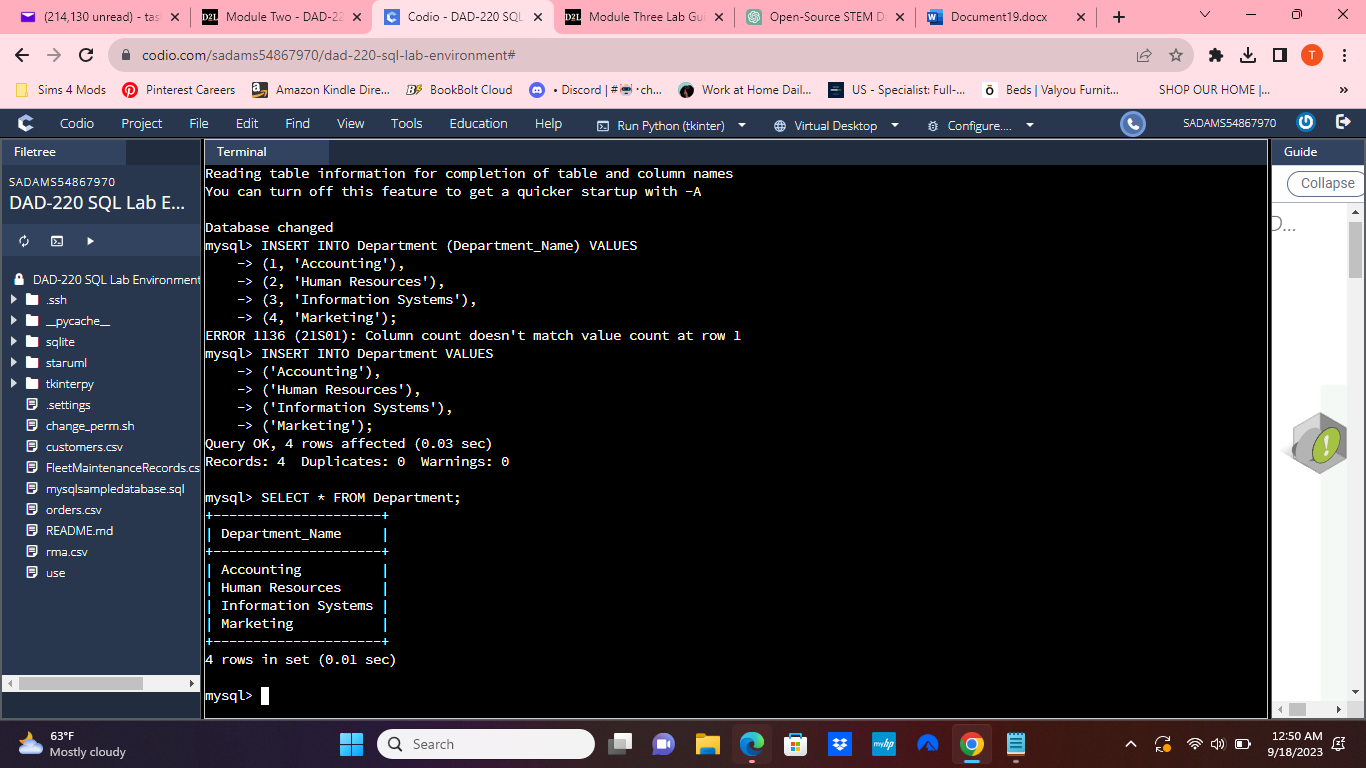
**Update the name of the Branches table** that you created in the previous lab to say "Department".

* Use an ALTER statement to successfully RENAME the "Branches" table to "Department".
* Capture these outputs in a screenshot to validate that you’ve successfully completed this step.



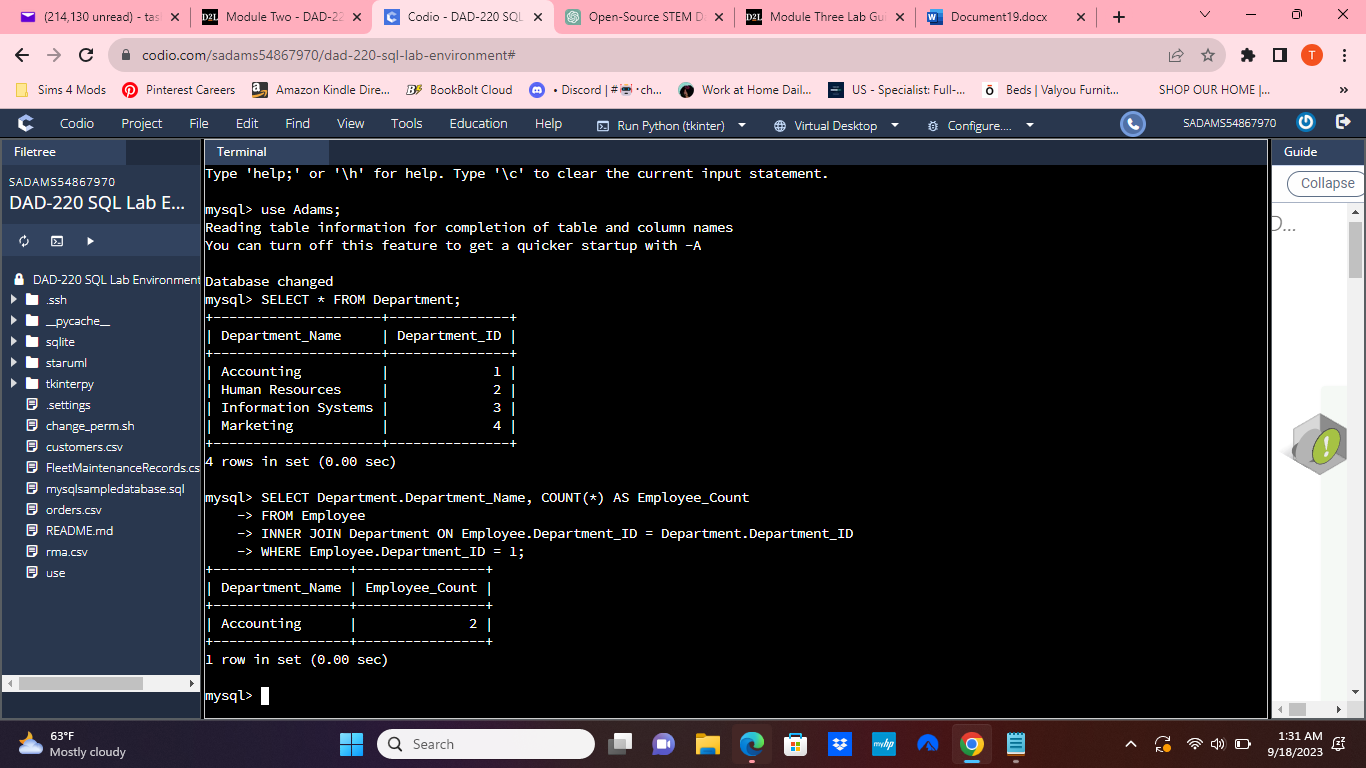
**Insert fields to the Department table** so that you’ll be able to perform joins on them.

* INSERT INTO Department VALUES  
  (1, 'Accounting'),  
  (2, 'Human Resources'),  
  (3, 'Information Systems'),  
  (4, 'Marketing');
* Write a SELECT statement for this table to prove this step and validate that it ran correctly with a screenshot.



Now, **perform joins between the Department and Employee tables and show results** for how many employees work in each one of the four departments. This will only provide information on the records that are already there.

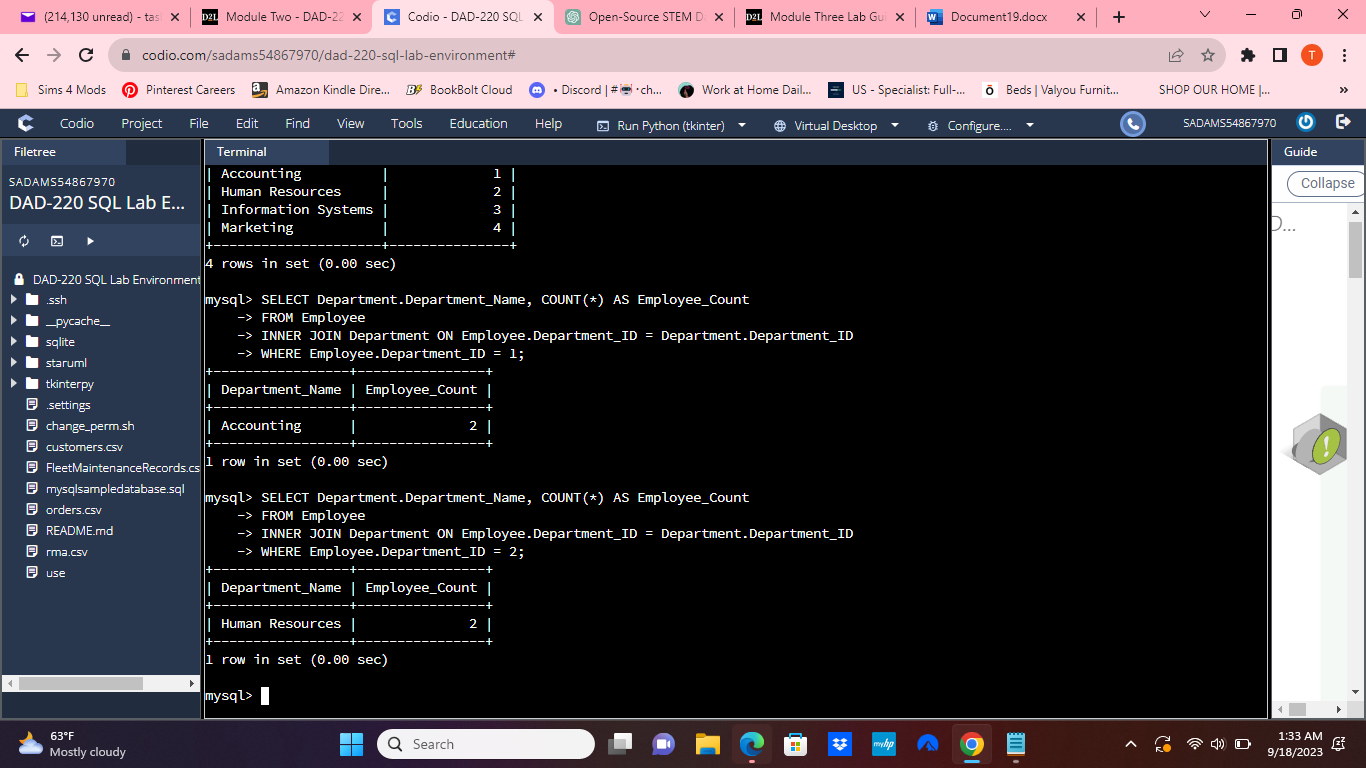
* Department 1 = Accounting



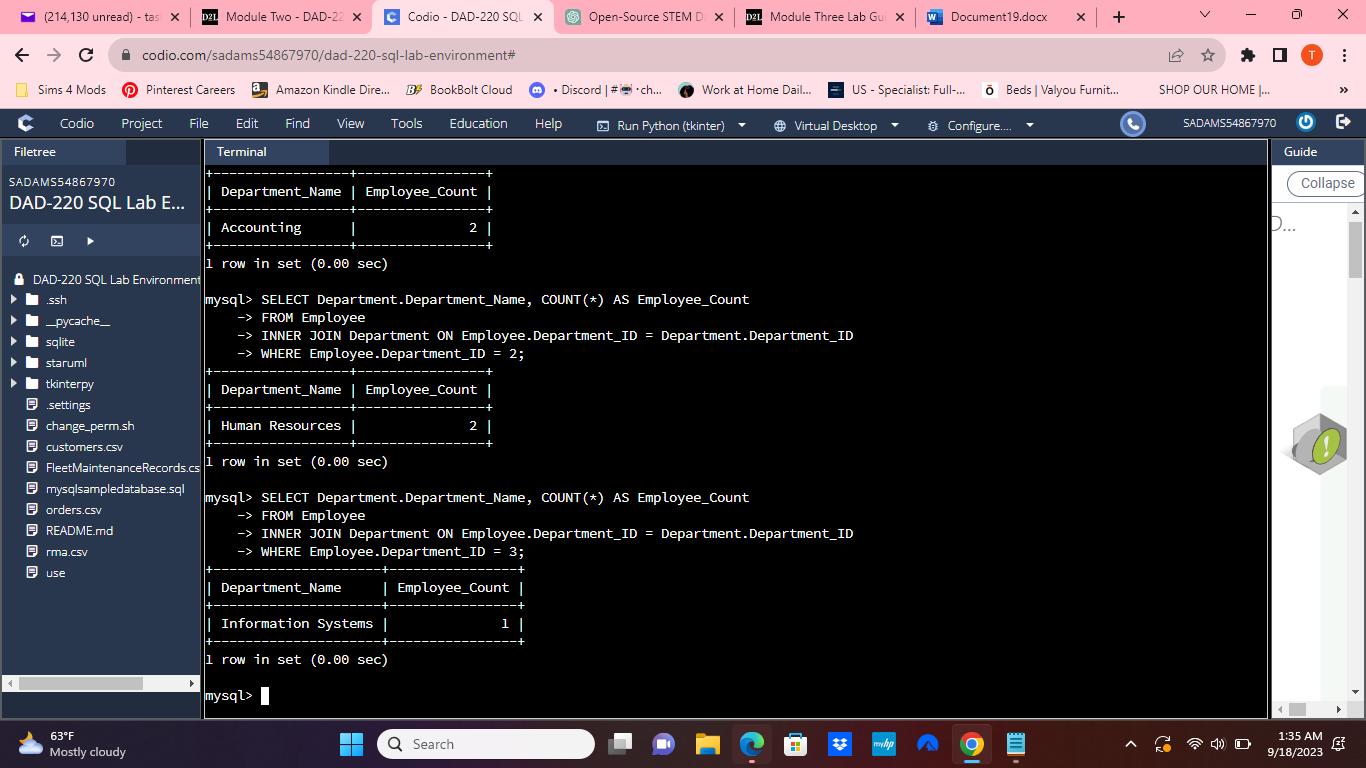
1. Command: SELECT First\_Name, Last\_Name, Department.Department\_Name FROM Employee INNER JOIN Department ON  
   Employee.Department\_ID = Department.Department\_ID WHERE Employee.Department\_ID = 1;

* Using SELECT statements similar to the one above, **perform joins to produce results** for the following tables:

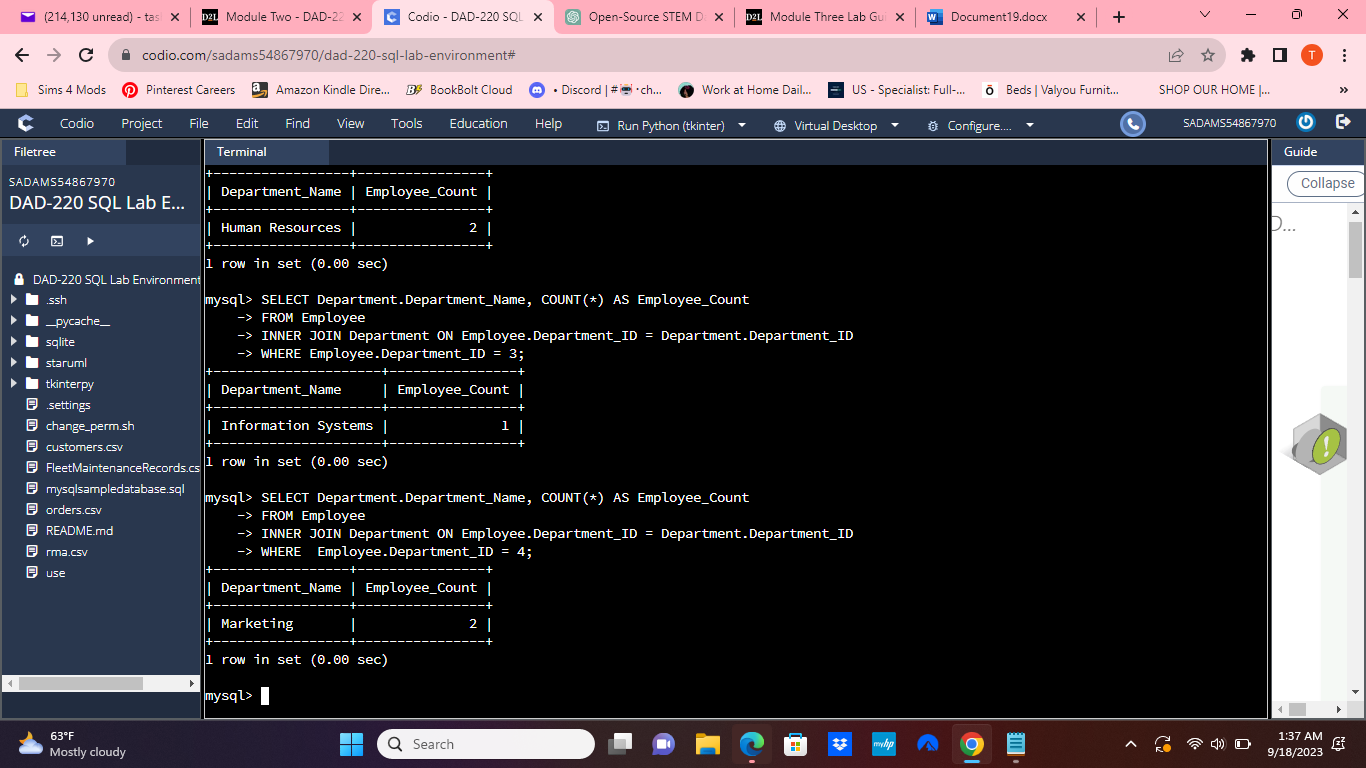
1. Department 2 = Human Resources



1. Department 3 = Information Systems

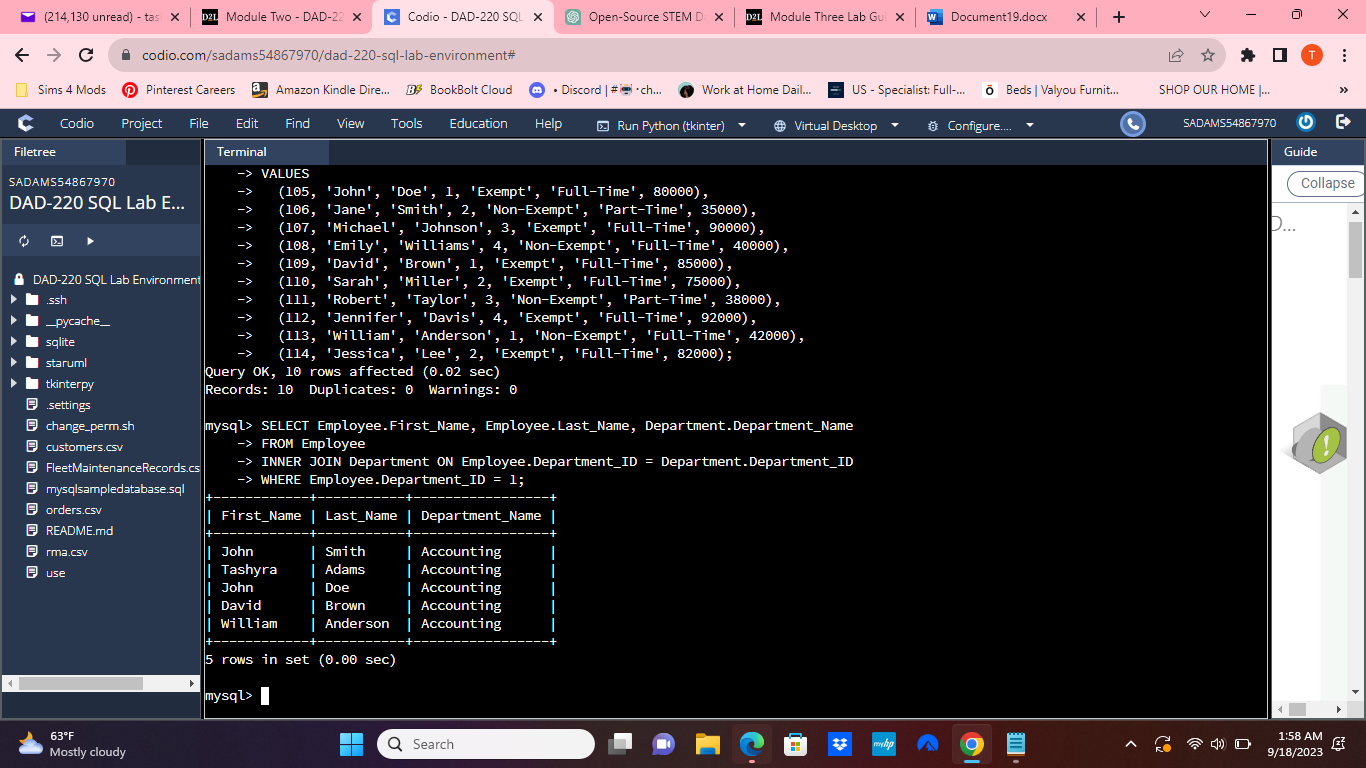


1. Department 4 = Marketing

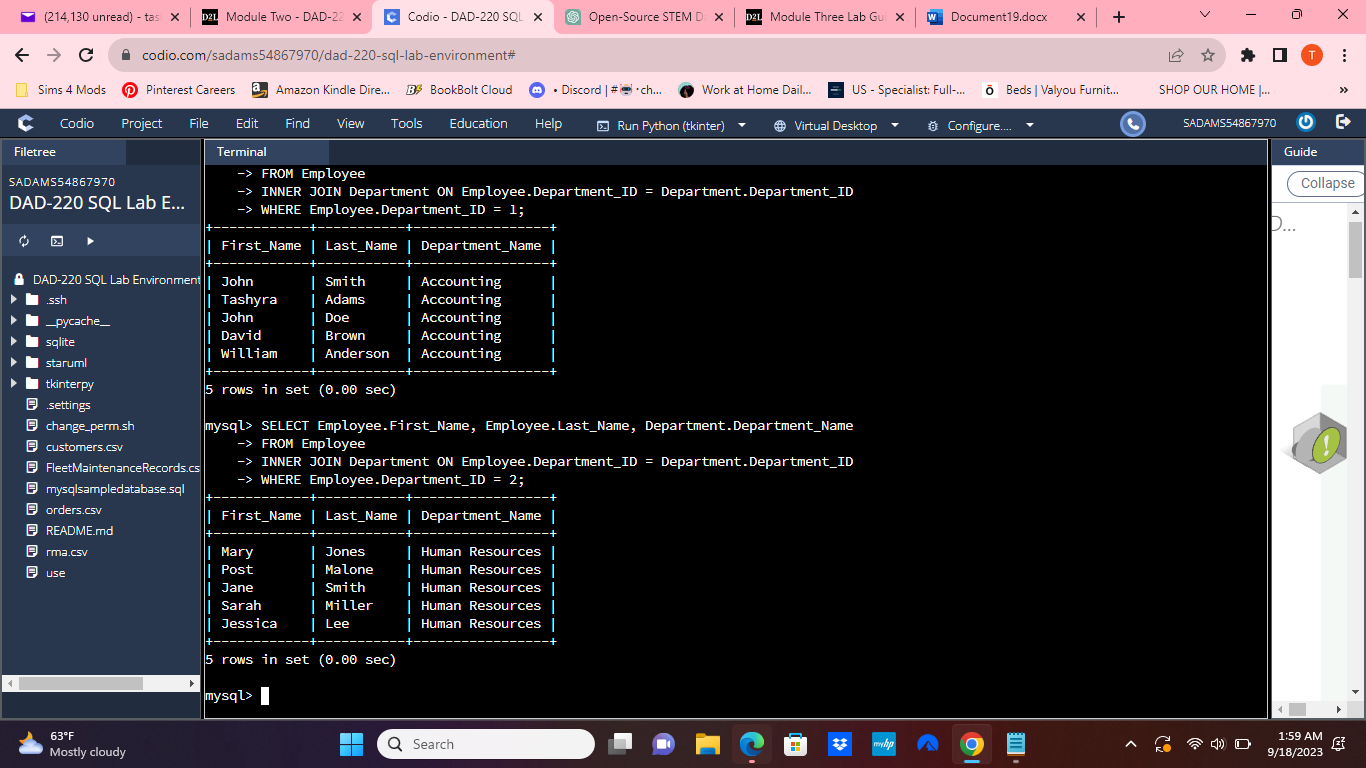


**Perform a join across the Employee and Department Tables** for each of the four departments. New and existing records should be displayed in the results.

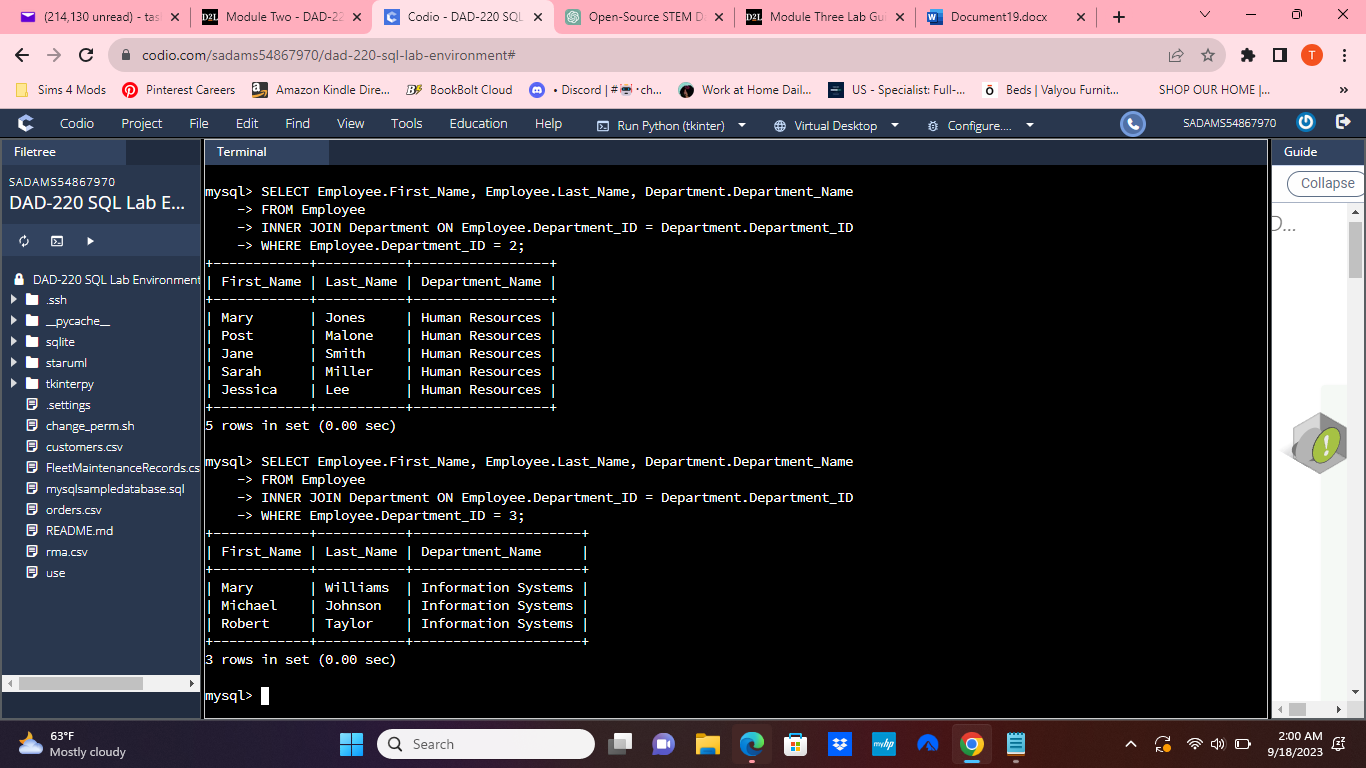
* Accounting



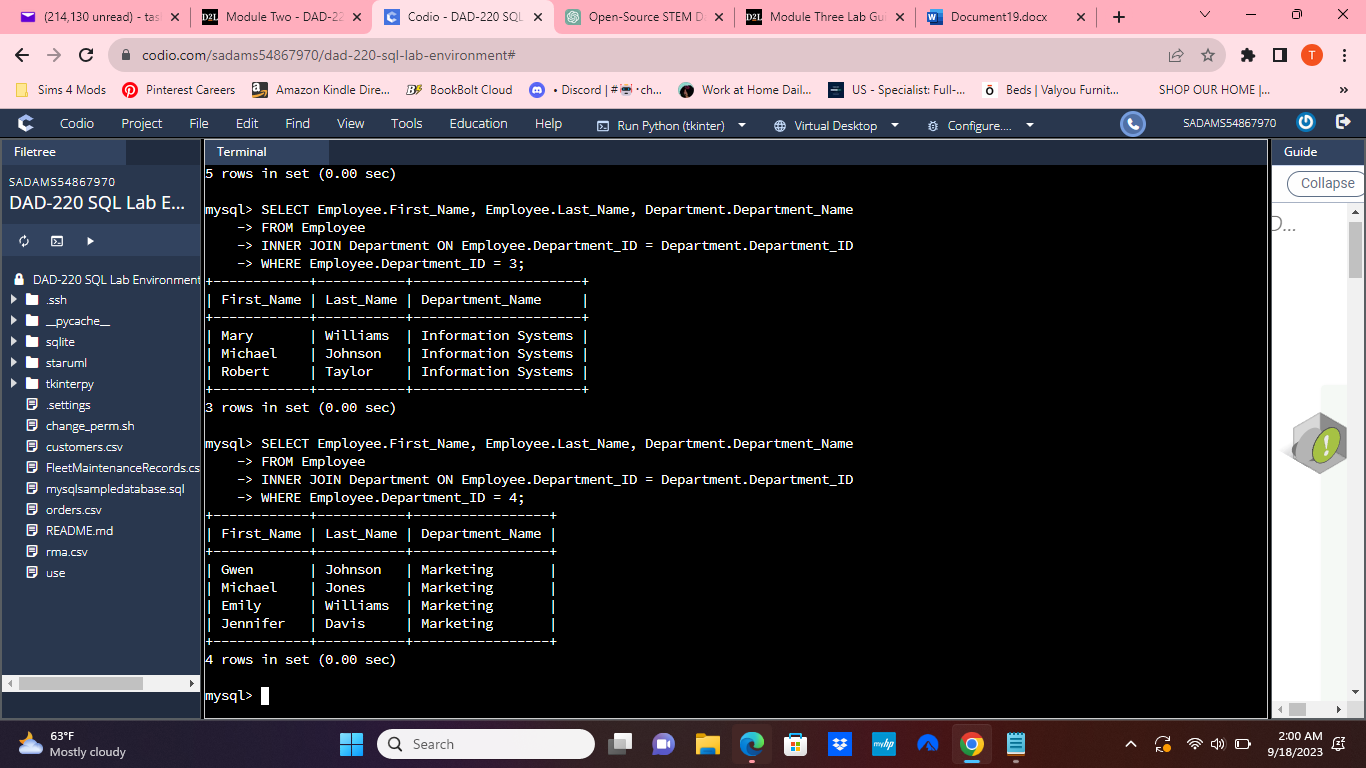
* Human Resources



* Information Systems



* Marketing



**Identify the resultant outputs** of the commands that you’ve written:

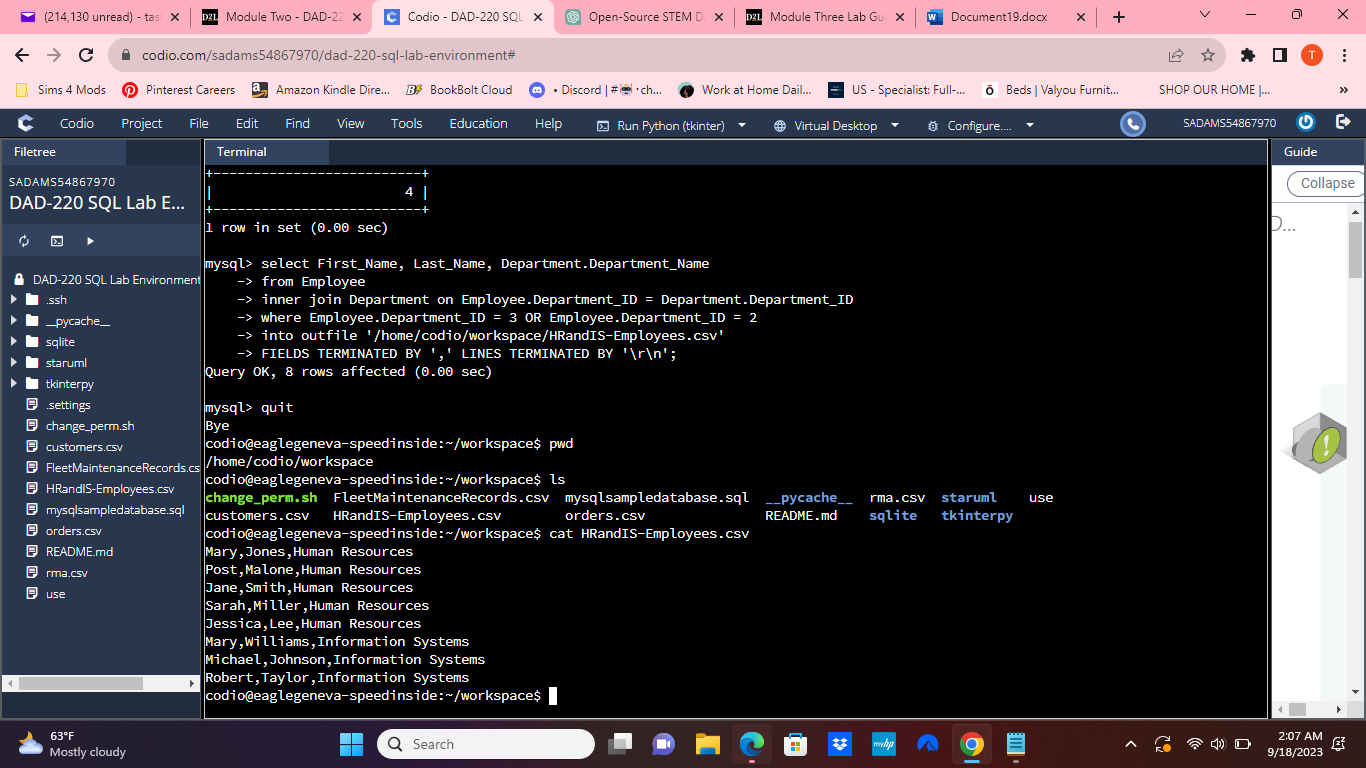
How many records are returned for employees in each department?

* Accounting = 5
* Human Resources = 5
* Information Systems = 3
* Marketing = 4

**Create a CSV file** that contains only the records of employees in Human Resources and Information Systems. If you run this query multiple times, be sure to use a different file name each time. MySQL will not overwrite an existing file.

* Enter the command listed below.

1. Command: select First\_Name, Last\_Name, Department.Department\_Name from Employee inner join Department on Employee.Department\_ID = Department.Department\_ID where Employee.Department\_ID = 3 OR Employee.Department\_ID = 2 into outfile'/home/codio/workspace/HRandIS-Employees.csv' FIELDS TERMINATED BY',' LINES TERMINATED BY '\r\n';



**Reflection**: Provide detailed insight on the prompts below by explaining your process along with how and why it ultimately worked.

* **Process**

1. **Explain** how **the joins** you used in this assignment worked.
2. **Describe** whythe **commands** you used were able to retrieve the Department table when you selected the Department name.

In this assignment, I had to work with a MySQL database and perform several tasks like creating tables, inserting data, performing joins, and exporting data to a CSV file. Let me reflect on how I went through the process and why it all worked out.

**Process:**

1. Connecting to the Database:

- First, I needed to connect to the MySQL database that I had set up in a previous lab. This was essential to access and manipulate the data.

2. Checking Employee Table:

- To ensure that I was working with the right database, I ran a simple `SELECT` statement on the Employee table. It was a basic check to make sure I had the correct dataset.

3. Renaming the "Branches" Table:

- I had to rename the "Branches" table to "Department." I used the `ALTER` statement for this, which allows changes to existing tables, including renaming them. It worked smoothly, and I had the table renamed as required.

4. Inserting Data into the "Department" Table:

- I tried to insert data into the "Department" table to populate it with department information. However, there was a small hiccup initially. My SQL statement was missing the column names in the `INSERT INTO` statement. Once I provided the column names, the data insertion went through as expected.

5. Checking "Department" Table:

- After inserting data, I wanted to make sure that the "Department" table now contained the information I added. So, I ran a `SELECT` statement to confirm that the data was correctly inserted.

6. Performing Joins:

- The heart of this task was performing SQL joins between the "Employee" and "Department" tables. These joins worked by matching the "Department\_ID" in the "Employee" table with the "Department\_ID" in the "Department" table. This created a connection between the two tables. I used `INNER JOIN` to retrieve records where there were matches in both tables.

7. Populating the "Employee" Table:

- Next, I had to insert data for ten new employees into the "Employee" table. It was important to ensure that the "Department\_ID" values fell between 1 and 4. The good news is that all these new records were successfully added to the table.

8. Performing Joins for All Departments:

- Finally, I performed joins between the "Employee" and "Department" tables for each of the four departments. To do this, I ran separate `SELECT` statements for each department, filtering by the corresponding "Department\_ID."

**Why It Worked:**

- SQL Joins: SQL joins function by combining rows from different tables based on related columns. In my case, the `INNER JOIN` statement allowed me to merge rows from the "Employee" and "Department" tables where "Department\_ID" values matched. This was crucial for retrieving information about employees and their respective departments.

- Table Renaming: Renaming the "Branches" table to "Department" worked perfectly because the `ALTER TABLE` statement is designed to modify a table's structure. Once I provided the new name, the database system updated the table name accordingly.

- Data Insertion: Data insertion into both the "Department" and "Employee" tables was successful when I correctly specified the column names and values. SQL requires this to ensure that the data goes into the right places.

- SELECT Statements: The `SELECT` statements worked as intended to retrieve specific data from the tables. By specifying columns and conditions in the `WHERE` clause, I could filter and retrieve the desired information.

In the end, the successful execution of these SQL operations demonstrated my ability to interact with a relational database, manipulate data effectively, and carry out various essential database tasks.

File creation and extraction

1. **Identify** how many **records** are in the file when you write the records of your query to a CSV file.
2. **Explain**, in detail, the process of **extracting data** to a flat file.

Identifying the Number of Records:

To find out how many records are in the CSV file, I used a Linux command. First, I opened my terminal and navigated to the directory where the CSV file was located. Then, I used the `wc -l` command followed by the name of the CSV file, like this: `wc -l your-file.csv`. This command counted the lines in the file, and the number of lines corresponds to the number of records in the CSV file.

Process of Extracting Data to a Flat File (CSV):

The process of extracting data from MySQL and saving it to a CSV file involves several steps:

1. Write the SQL Query: I started by crafting an SQL query that selects the specific data I wanted to export. This query can include conditions, joins, and other SQL clauses to customize the data retrieval.

2. Use the `INTO OUTFILE` Clause: In my SQL query, I used the `INTO OUTFILE` clause to specify the file path where I wanted to save the CSV file. This clause is followed by `FIELDS TERMINATED BY` and `LINES TERMINATED BY` clauses, which determine the format of the CSV file.

3. Execute the Query: I executed the SQL query in my MySQL client, which initiated the export process. The data selected in the query was then written to the specified CSV file.

4. Verify the CSV File: After executing the query, I checked whether the CSV file was created in the location I specified. This can be done using file management commands or tools.

5. Counting Records: As mentioned earlier, I used the `wc -l` command to count the number of lines in the CSV file, which corresponds to the number of records.

6. Open and Inspect the CSV File: Finally, I opened the CSV file using various tools such as text editors, spreadsheet software, or data analysis tools. This allowed me to review the exported data, ensure its accuracy, and perform further analysis if necessary.

In summary, the process involves formulating an SQL query, specifying file details, executing the query, verifying the CSV file's existence, counting records, and then inspecting and analyzing the data in the CSV file.