DAD 220

Professor August

Darrell Walker

1. November 2023
2. **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.

A screenshot of a computer program

Description automatically generated

1. **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.

A screenshot of a computer

Description automatically generated

1. 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
     2. Department 3 = Information Systems
     3. Department 4 = Marketing
   * Capture the results of these joins and validate your work by providing a screenshot. You should have the same number of records as you do employees.

A screenshot of a computer program

Description automatically generated

1. **Populate the Employee table with**information for ten **new employees.**  
   * Give them unique names and include attributes for **all** necessary fields. (Note: Please reference attributes from the lab in Module Two. Department ID values must be between 1 and 4.)

A screenshot of a computer program

Description automatically generated

1. **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.  
   * Take a screenshot to capture the updated results that the Employee and Department joins show to validate that they have run correctly. You should have the same number of records as you do employees.

A screenshot of a computer screen

Description automatically generated

1. **Identify the resultant outputs** of the commands that you’ve written:  
   * How many records are returned for employees in each department?

A computer screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

1. **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';
   * Print the file output to the screen.  
     1. You’ll need to type the word **quit** after your MySQL prompt and then press **Enter** to exit to the Linux shell. Do not exit the virtual lab environment itself.
     2. Next, print the output of your file to the screen by following these steps:
        1. Type pwd and press **Enter**, then type ls and press **Enter** again. This will list your files.
        2. Now, type cat HRandIS-Employees.csv and press **Enter**.
        3. Capture these outputs in a screenshot to validate that you’ve successfully completed this step.

A screenshot of a computer screen

Description automatically generated

**8. 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.
  + 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.

In this assignment, I used a method in databases called "joins" to bring together information from two tables, like combining puzzle pieces. The join I used makes sure that only the information that matches in both tables is included. For example, when I wanted to find details about employees in a specific department, the join connected the employee information to the department information using a shared element called Department\_ID. This way, I got a combined view of employee names and their departments.

The commands I used to get the Department table when selecting the Department name worked because of the simple way the SQL SELECT statement works. When I wrote a command like SELECT Department.Department\_Name FROM Department;, it was like asking the database to show me all the department names in the Department table. The SELECT statement, without any extra conditions, fetches all records from the specified table. So, the command worked because it matched the basic function of the SELECT statement to get all records from a table, in this case, the Department table.

When we write the records of our query to a CSV file, we can figure out how many pieces of information are in the file by using a special command. It's a bit like counting how many lines are in a letter or a paragraph. In this case, we’d use a command called **wc -l** along with the name of your CSV file. The computer then tells us the total number of lines in that file, and each line usually represents a record or a piece of information. So, when we see the number of lines, we know how many records are in the file.

Now, let's talk about how we put data into a flat file like a CSV. Imagine you have a table of information in a computer, like a list of employees and their details. When you want to keep a copy of that information in a simple text file, we can use a command like INTO OUTFILE followed by the path and name of the file. This command is like asking the computer to take the information from the table and put it neatly into a file. we also get to decide how the information looks in the file, like what separates one piece of information from the next. It's a bit like taking a picture of your table and saving it on your computer so you can look at it later. Once the data is in this flat file, it's easy to share or analyze.