**Assignment 3: Identify issues with dataset**

**Submitted to:** Amad Mumtaz  
**Submitted by:** Huzaifa Munir  
**Dated:** 05th November 2024

**Task: Identify issues with dataset**PFA Customers data. This has some updated values in the rows.

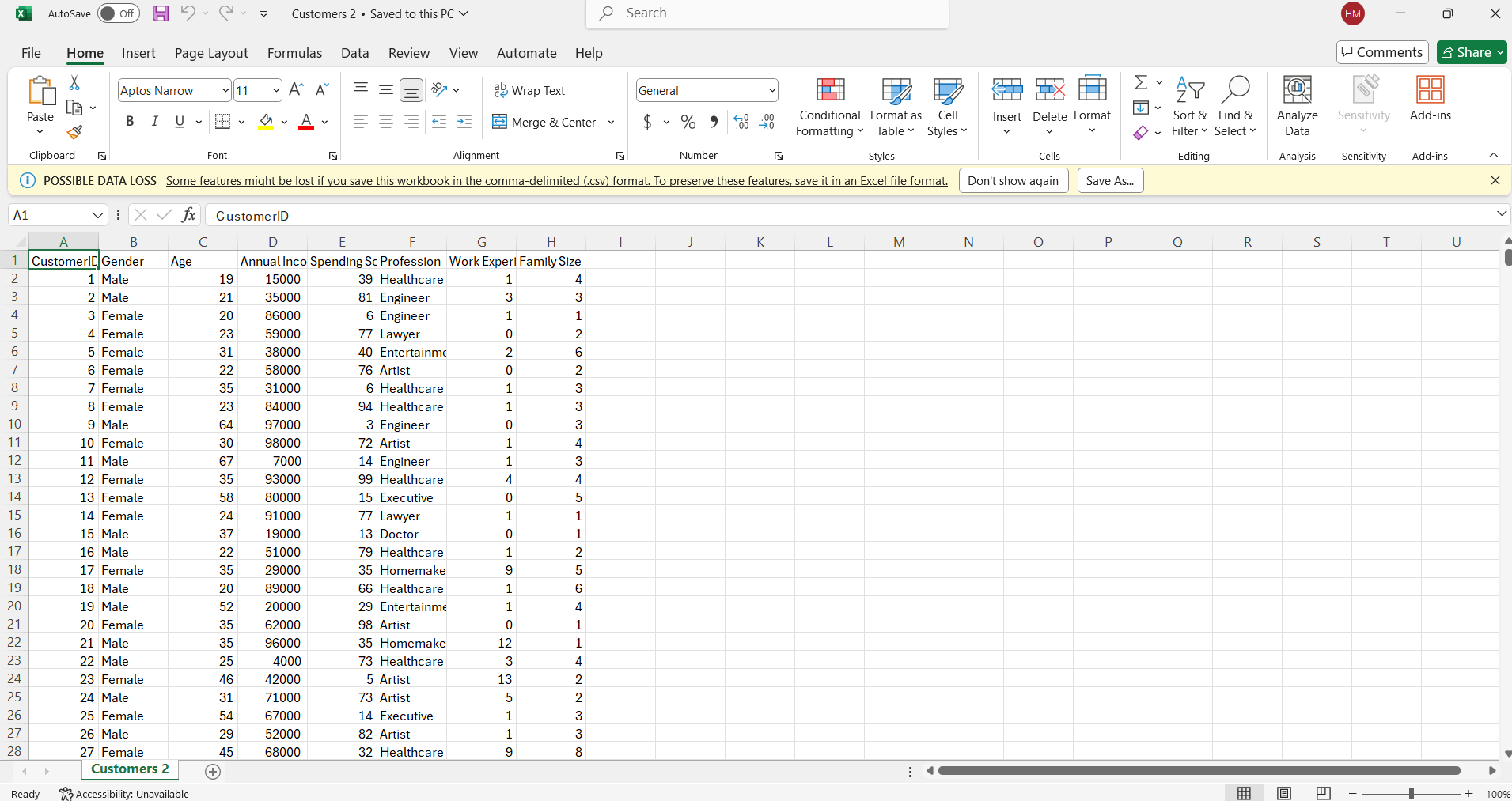
Considerations:

* + Consider this data as inserted in an application's database -> customer’s table.
  + The application has a form (UI) which loads a data table including all values fetched from this customers table (having attached/ below dataset)
  + During the page/ data table loading, the page does not provide complete values i.e. fails to load.
  + This could be a result of faulty dataset inserted in the database table.
  + Some points:
    - Customers form fetches and displays data in the web app over following fields:
    - CustomerID (int primary key)
    - Customer name (varchar) (assume coming from another referenced table)
    - Gender (varchar)
    - Age (int)
    - Annual income (int)
    - Now the form fails to load properly as the you might be missing an implementation which caters duplicate records, float displaying over an int field, or a missing value NULL (could be any error occurring in the application).
    - As a QA Engineer, you task is to explore this dataset and find what could be the values/ rows/ records which are causing these problems.
      * Example, a duplicate record in id column
      * A missing value in any other column i.e. NULL such that there is no condition in your code to cater this.
      * varchar being stored in int field

Load this dataset in your database and create new table. Run queries to know your dataset and highlight any problems which you think is impacting your application.

Prepare a word file and show what errors/ discrepancies you have observed in the dataset. Provide reasons to why this could be causing any issues. Submit your solutions in the shared folder.

**Provided Dataset:**

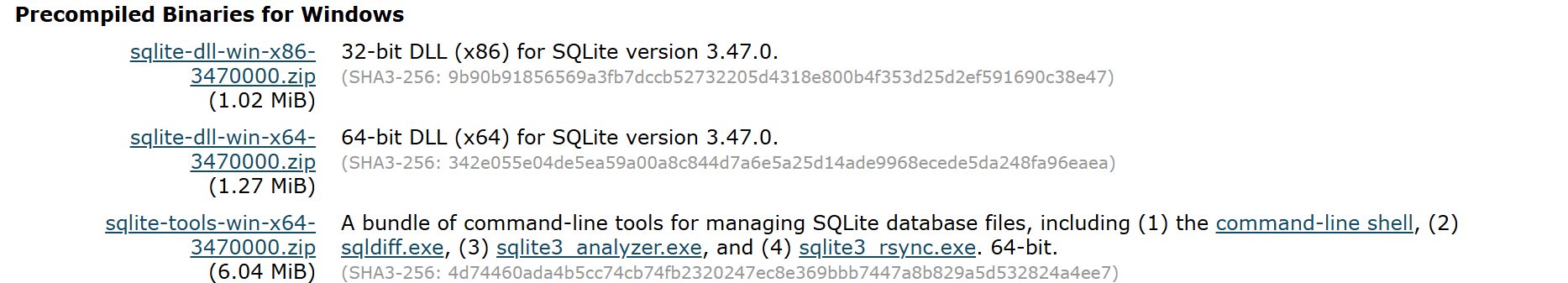


**We are provided with the customers database file In which there are issues and we have to identify those issues.**

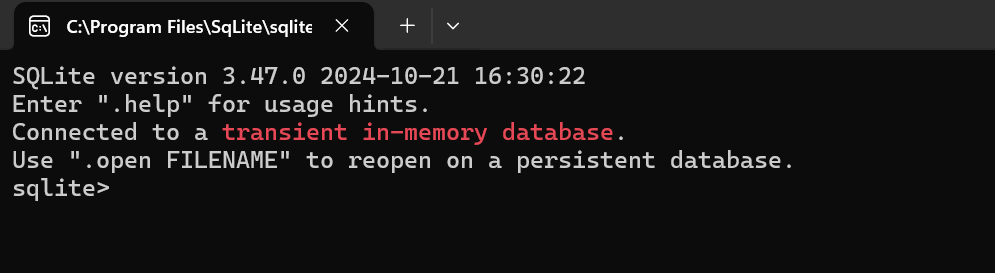
**Solution:**

1. **Setting up the SQLite Command Line Interface and its Environment:**

SQLite is a lightweight serverless database that is easy to use for small scale tasks.

* 1. **Step1: Installation**
     1. Install SQLite from the web if it is not installed on your PC.
     2. Go to the official SQLite website and according to your operating system select the respective option.
     3. Under the **Precompiled Binaries for Windows** section, download the zip file for [sqlite-tools-win-x64-3470000.zip](https://www.sqlite.org/2024/sqlite-tools-win-x64-3470000.zip) (3rd one).  
        
     4. After downloading it, now let’s set up SQLite. Create a folder by the name of SQLite in the program files (recommended, not necessary) of your C:/ drive and then extract the zip file to this folder.
     5. Open the folder where you extracted the files. Here, you will see 4 files: sqlite3\_rsync.exe, sqlite3\_analyzer.exe, sqlite3.exe, sqlite3.exe.

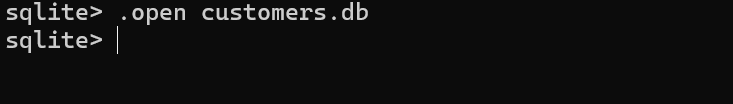
Double Click to open the sqlite3.exe file and it will open an SQLite command line interface.

****  
**Note:** The above work is for Windows. Kindly, check your OS requirements before downloading and setting.

* 1. **Step2: Database Creation**

Now create a persistent database.

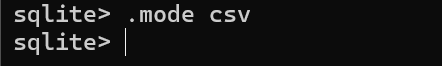
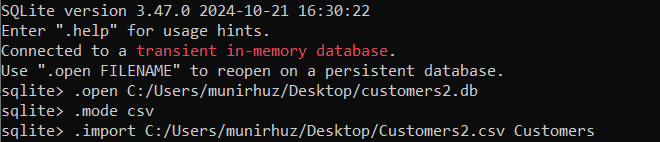
* + - To do this, first make sure you are in the right folder where you have full permissions.
    - Firstly, change the directory to **Desktop.** For this, type the following command in the SQLite command line interface.  
      A screenshot of a computer

      Description automatically generated
    - Now, create the database by typing the following command.   
      
    - Once the command runs successfully, a file called customers2.db should appear on your **Desktop** or the specified directory. You can name the file anything you like.
    - You can also perform the same task with the following command.  
      

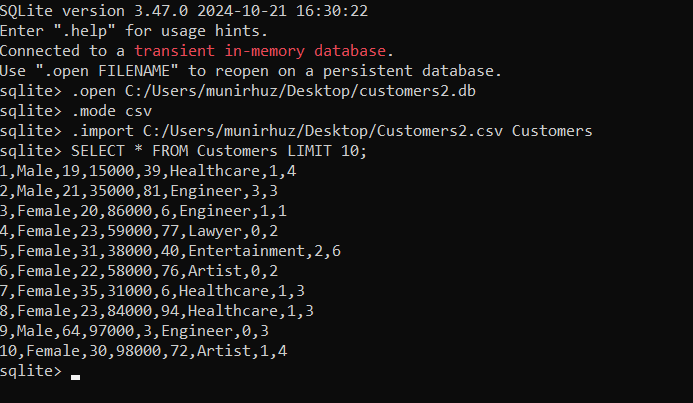
**Note:** Make sure that you are in a directory where you have permission to create files. The simplest way to avoid this issue is to navigate to a folder like your **Desktop** or **Documents** where you have full access.

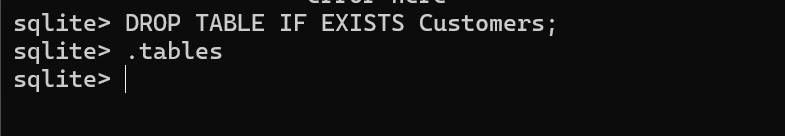
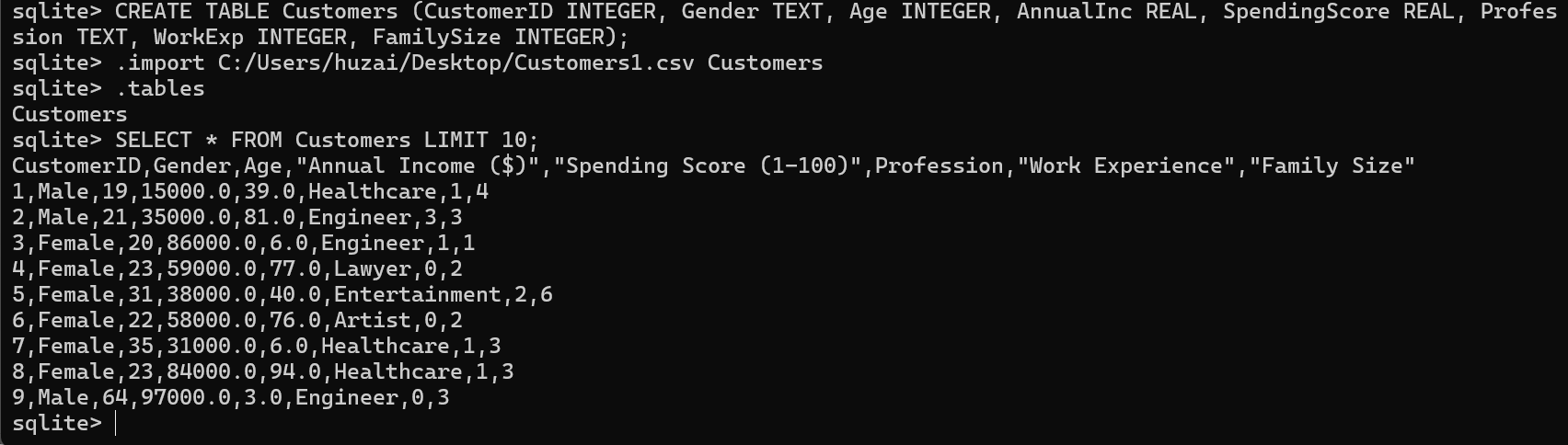
* 1. **Step3: Importing the CSV File**

Now, import the CSV file.

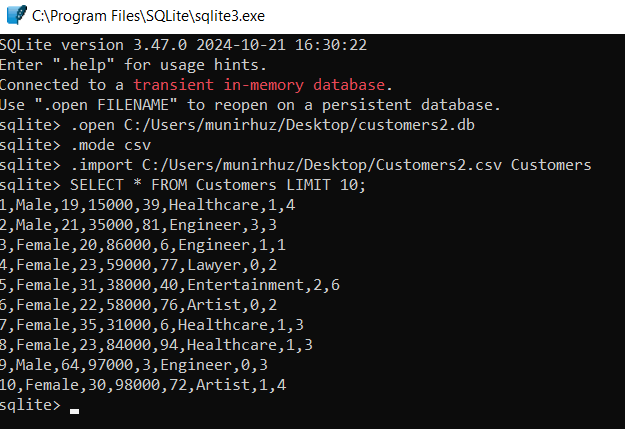
* + - To perform this task, first set the mode to CSV by following command.  
        
      
    - Now, import your CSV file. (Since, I kept it in my Desktop, you can do the same or choose your preferred directory.)  
      ****
    - The CSV is imported into a **table** (in this case, named Customers) inside your SQLite database (customers.db). The original CSV file (Customers1.csv) itself remains unchanged, but its data is copied into the SQLite table (Customers).  
      ****
  1. **Step4: Verification**

Now, verify if the above commands were successful.

****

* + - If it doesn’t work, you can try creating the table manually and then copying the data to it. I will first drop the table and then check if there are any tables.   
      ****  
      If there were, the output would have been “Customers”.
    - Now, we will be manually creating the table and importing all of the data into it.  
      Also, check with ‘.tables’ command to verify.  
      ****
  1. **Step5:**

Check the structure of the imported table.



1. **Issues in the database:**
   * **Missing Values/Blank OR Empty String**:
     + **CustomerID**: There is 1 missing value in the CustomerID column, which could disrupt data retrieval since this is the primary key and is expected to be unique and non-null.

**Example**: Row 123 in csv file.



**SQL Command**: SELECT \* FROM Customers WHERE CustomerID = ' ';



**NOTE:**

* + - * **For NULL:**If you are running the IS NULL command i.e., **SELECT \* FROM Customers WHERE CustomerID IS NULL;** and it is not running try using the above command.
      * **For Zero Values (if zero is not valid in your primary key logic):** If you are running the IS NULL command i.e., **SELECT \* FROM Customers WHERE CustomerID = 0;** and it is not running try using the above command.
    - **Profession**: There are 35 missing values in the Profession column, which may cause issues if the application expects all values to be present without a NULL-handling mechanism.

**Example**: Row 81, 120, 221, 239 etc. in csv file.

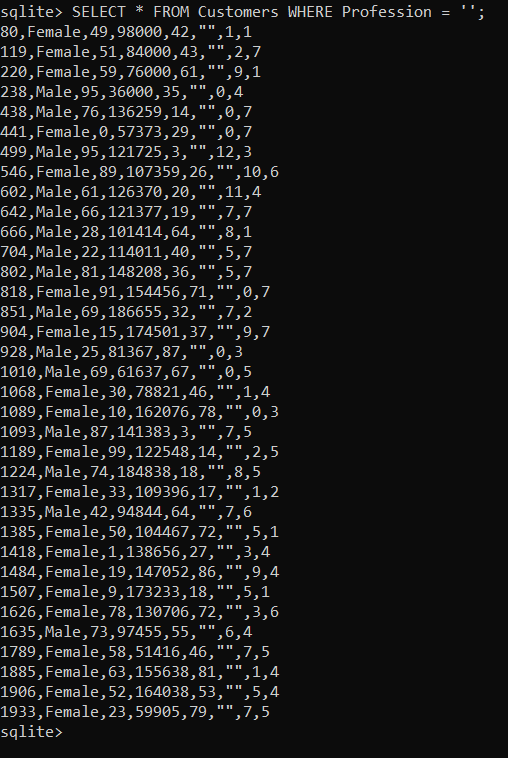








**SQL Command**: SELECT \* FROM Customers WHERE Profession = ' ';



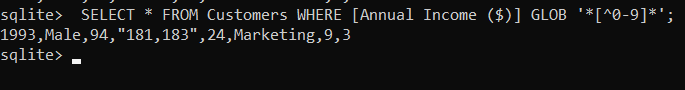
**NOTE:**

* + - * **For NULL:**If you are running the IS NULL command i.e., **SELECT \* FROM Customers WHERE Profession IS NULL;** and it is not running try using the above command.
      * **For Zero Values (if zero is not valid in your primary key logic):** If you are running the IS NULL command i.e., **SELECT \* FROM Customers WHERE Profession = 0;** and it is not running try using the above command.
  + **Data Type Discrepancies**:

**Annual Income ($)**: The Annual Income ($) column is stored as an object (string) instead of an integer, indicating potential incorrect data entries or format issues (e.g., extra spaces or symbols). This may prevent the application from processing income values correctly.

**Example**: Row 1995 in csv file.

**SQL Command**: SELECT \* FROM Customers WHERE [Annual Income ($)] GLOB '\*[^0-9]\*';



**NOTE:**

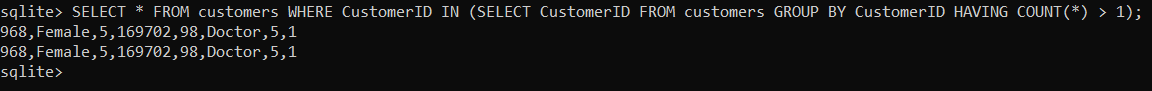
* + - * If you are using the command SELECT \* FROM Customers WHERE [Annual Income ($)] NOT LIKE '%[^0-9]%'; than this will output all of the queries in the table which is not right.
      * For this issue to be resolved we use **GLOB** which is an SQLite operator that allows **pattern matching** using **Unix shell-style wildcards**.
      * Also, \*[^0-9]\* matches any row where Annual Income ($) contains characters other than digits (0-9).
  + **Duplicate Records**:

**CustomerID**: There are duplicate entries for CustomerID 968. Since CustomerID is a primary key, duplicates here could lead to conflicts in data integrity and issues when fetching specific customer records.

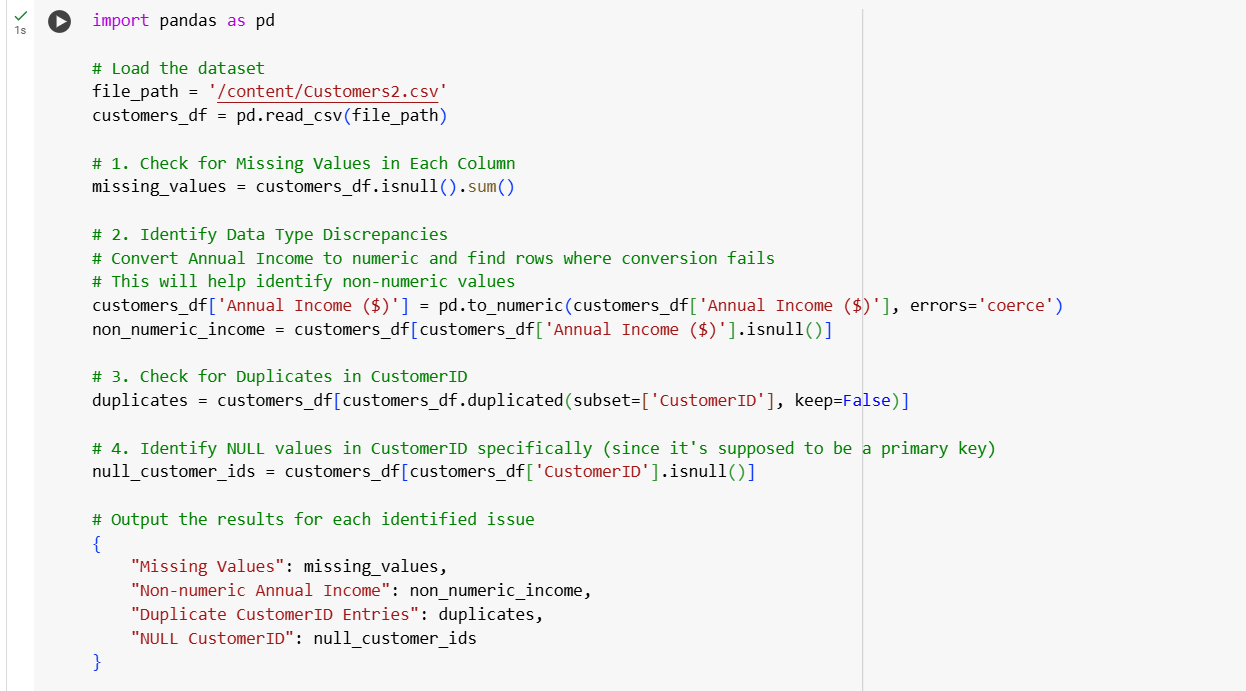
**Example**: Row 969 & 970 in csv file.



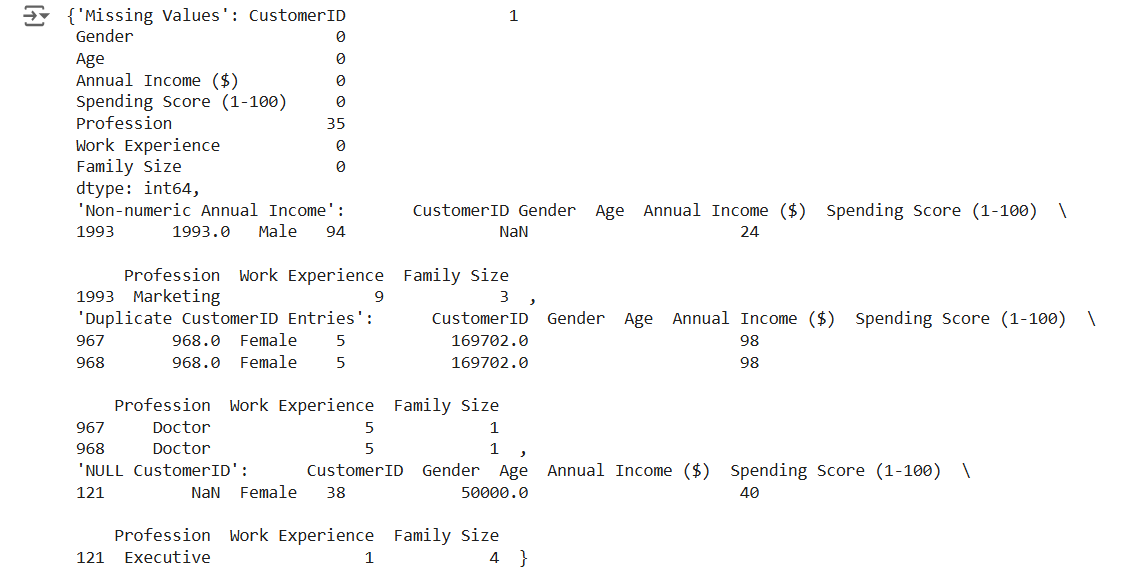
**SQL Command**: SELECT \* FROM customers WHERE CustomerID IN (SELECT CustomerID FROM customers GROUP BY CustomerID HAVING COUNT(\*) > 1);



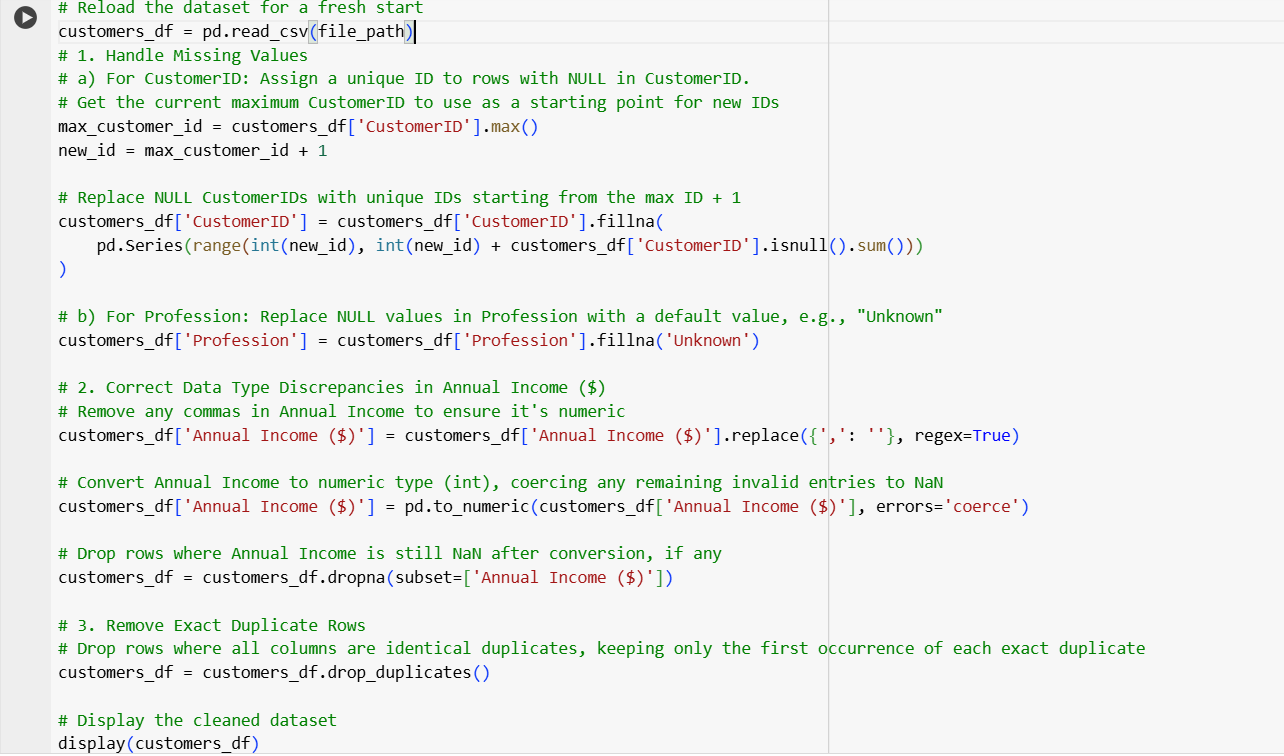
1. **Python Code for Identifying Issues:**



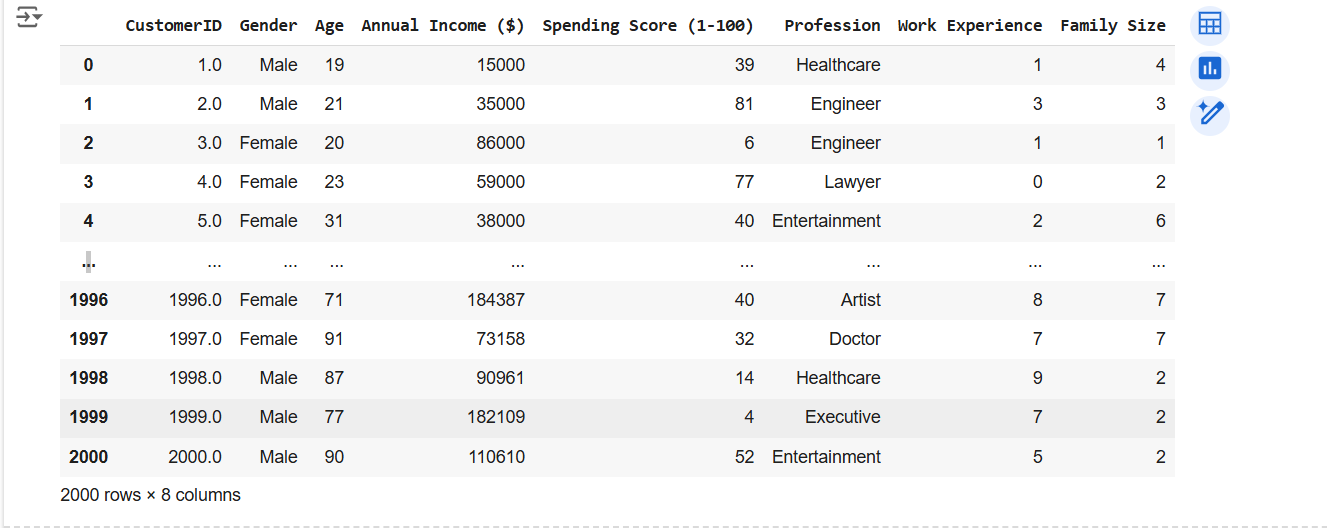
**Output:**

****

1. **Python Code to Resolve the Issues**:



**Output:**

****

1. **Conclusion:**

In conclusion, the analysis revealed issues such as missing values, data type inconsistencies, and duplicate records in the Customers dataset. These issues, including NULLs in critical fields, non-numeric entries in Annual Income, and duplicate CustomerIDs, can disrupt data retrieval and display in the application. Addressing these with SQL queries to clean and standardize the data will enhance data integrity, ensuring smoother and more reliable form functionality.