# Title - ORM CSV Saver

In this assignment, the objective is to develop a CSV data management system using the concept of parent and child classes in Python. The primary purpose of this system is to provide a streamlined approach for handling CSV data through the implementation of class structures that facilitate various data operations. The assignment consists of a parent class, CSVSaver, and a child class, CSVOperation, both of which are designed to enhance the efficiency and organization of CSV data manipulation.

# Tech Stack

- Programming Language: Python
- Object-Oriented Programming (OOP)
- Integrated Development Environment (IDE)
- Version Control
- MS Excelm

## Guidelines

**Parent Class: CSVSaver**

The primary aim is to create a parent class, CSVSaver, that offers a comprehensive set of methods for managing CSV data. The objectives include:

Developing class methods within CSVSaver for reading, updating, and deleting data rows in a CSV file.
Designing a static method to create a new CSV file with specified column headings.

Establishing the CSV_Saver class as a foundational structure for building more specialized child classes.

**Child Class: CSVOperation**

The secondary objective is to design a child class, CSVOperation, which inherits from the CSVSaver parent class. This child class will provide a tailored interface for specific data manipulation tasks. The objectives are:

Constructing the child class with a constructor that initializes the CSV file name and column headings.
Implementing methods within CSVOperation to create and update data rows, utilizing functionalities from the parent class.
Demonstrating the advantages of inheritance in terms of code reusability and modularity.

**Demonstration and Practical Application**

The assignment's practical aspect involves showcasing the use of the CSVSaver parent class and its derived CSVOperation child class. Objectives include:

Illustrating scenarios of data creation, reading, updating, and deletion using the child class.
Emphasizing the benefits of adopting this approach for structured and organized code.
Exploring possibilities for extending the child class to cater to specific application needs.

**Conclusion**

By fulfilling these objectives, this assignment aims to provide a comprehensive understanding of the proposed solution for managing CSV data. The use of parent and child classes in Python facilitates efficient and effective data manipulation, while the implementation details and demonstrations underscore the advantages of this approach for real-world applications.

**Submission guidelines**

To submit your assignment, you are required to push your code to either GitHub or GitLab repositories. Both platforms offer version control and collaboration tools that facilitate the submission and review process.

***Cond… checklist***

**Checklist while completing the assignment**

*Parent Class: CSVSaver:*

- ☐ Defined the CSVSaver class using the class keyword.
- ☐ Created class methods for reading, updating, and deleting data rows in a CSV file.
- ☐ Implemented a static method to create a new CSV file with specified column headings.
- ☐ Utilized the csv module to handle CSV file operations.

*Child Class: CSVOperation:*

- ☐ Derived the CSVOperation class from the CSVSaver parent class using inheritance.
- ☐ Created the child class constructor with appropriate initialization of CSV file name and column headings.
- ☐ Implemented methods within CSVOperation to create and update data rows, using methods from the parent class.
- ☐ Demonstrated the inheritance mechanism by accessing parent class methods from the child class.

*Class Method Usage:*

- ☐ Invoked the class methods of both the parent and child classes correctly.
- ☐ Used @classmethod decorator for class methods in the parent class.
- ☐ Accessed class methods using the class name (ClassName.method_name()).

*Static Method Usage:*

- ☐ Defined static methods with the @staticmethod decorator.
- ☐ Utilized static methods to create new CSV files with defined column headings.
- ☐ Accessed static methods using the class name (ClassName.method_name())

*Inheritance and Encapsulation:*

- ☐ Demonstrated proper inheritance by utilizing parent class functionalities in the child class.
- ☐ Encapsulated data and methods within the classes to achieve modular and organized code.

*Code Readability and Style:*

- ☐ Used appropriate variable and method names that convey their purpose.
- ☐ Followed Python naming conventions (e.g., snake_case for variables and methods).
- ☐ Maintained consistent and appropriate indentation for code blocks.

*Comments and Documentation:*

- ☐ Added comments to explain complex sections of code or methods.
- ☐ Included brief documentation for class definitions, constructor, and major methods.

*Testing and Validation:*

- ☐ Tested the implemented classes and methods for correctness and expected behavior.
- ☐ Verified that CRUD operations (Create, Read, Update, Delete) work as intended.
- ☐ Conducted various test cases to cover different scenarios.