**Objective:**

Develop a linear program from scratch using the PuLP package in Python to generate a master schedule for Chico High School. The code should be structured in two separate sections (cells):

**Section 1:** Implement the linear program to create the master schedule.  
**Section 2:** Simulate and generate CSV data (students.csv, teachers.csv, and courses.csv) that will be used to test the linear program.

**Key Requirements:**

* **Thorough Review:** Ensure that each line of the program is thoroughly reviewed and satisfies all the constraints provided.
* **Detailed Comments:** Include detailed comments explaining how each constraint is implemented within the code.
* **No Duplicate Course Requests:** Ensure there are no duplicate course requests in the course\_requests field for any student.
* **Maintain Linearity:** All constraints must maintain linearity to adhere strictly to linear programming principles.
* **Data Consistency:** Ensure data consistency between the simulated data and the constraints.
* **Objective Function Accuracy:** The objective function must maximize the number of students receiving their requested courses.
* **Avoid Over-Constraining:** Adjust constraints to avoid infeasibility while reflecting accurate scheduling requirements.
* **Qualified Teachers:** Ensure teachers are assigned only to courses they are qualified to teach.
* **Realistic Data Simulation:** Simulate data that reflects real-world scenarios, ensuring the linear program can find a feasible solution.

**Section 1: Implement the Linear Program**

**Purpose:** Write a linear program that generates the master schedule for Chico High School based on the provided constraints.

**Program Requirements:**

* **Data Loading:** Load data from CSV files (students.csv, teachers.csv, and courses.csv).
* **Constraints Handling:** Implement all constraints while maintaining linearity. Include capacity and section constraints for courses and ensure data consistency.
* **Objective Function:** Maximize the total number of requested courses assigned to students, aiming to maximize student satisfaction.
* **Output:** Generate a master schedule in a readable format (DataFrame or CSV) and a list of students who did not receive all their requested courses, along with reasons.

**Constraints to Implement:**

* **Scheduling Structure:**
  + Alternating A/B Block Schedule (Red/Gold Days): 4 periods per day, 90-minute periods. Total of 8 periods (4 Red, 4 Gold).
  + Consistency Across Days: No student should have more than 4 classes on a single day.
* **Student Course Load:** Students can take up to 8 courses.
* **Free Period Preferences:**
  + Students can choose free periods where they do not have class. This will be encoded as a “Free First Period” or “Free Fourth Period” class request. Students do not have the choice of whether it’s a gold or red day.
* **Capacity and Sections:** Include course capacity and section constraints to ensure sufficient seats for each grade level.
* **Department-Specific Constraints:**
  + **Social Science:** "Student Government" must be scheduled during Red period 2 and Gold period 3.
  + **World Language:** Group courses together for simplicity in teacher scheduling (e.g., Spanish 1 on one day, Spanish 2 on another).
  + **Medical Department:** "Medical Career" must be scheduled during Red period 1 and Gold period 1. "Heroes Teach" during Red period 2 and Gold period 2.
  + **PE/Health:** Assign a male and female teacher to each PE period for locker room coverage.
  + **Science:** Teachers require prep/lunch breaks between lab-based courses. They can teach back-to-back sections of the same course but need prep time if alternating subjects.
  + **SPED (Special Education):** Co-taught classes can accommodate no more than 12 SPED students.
* **Teacher Constraints:**
  + Teachers are only assigned to courses they are qualified to teach.
  + Honor teachers’ designated free periods for part-time status, coaching, or district meetings.
  + Teachers who teach only two periods must have them scheduled back-to-back.
  + Teachers cannot teach multiple sections during the same period on either Red or Gold days.
* **Student Preferences:**
  + Students can choose a free AM (1st period) or PM (4th period) based on their preferences.
  + Students should only be assigned to courses they have requested.
  + Students can only be in one section per period.
* **Class Balance:** Balance class sizes across periods to avoid overcrowding.

**Section 2: Simulate and Generate CSV Data**

**Purpose:** Simulate and generate realistic data in CSV format for students, teachers, and courses to test the linear program.

**Data Simulation Requirements:**

* **Courses:** Include all specific courses required by the constraints.
* **Consistency:** Ensure course names, IDs, and departments match those used in the constraints.
* **Qualified Teachers:** Generate enough qualified teachers for each course and assign teachers only to courses they are qualified to teach.
* **No Duplicate Course Requests:** Ensure no duplicate course requests in the course\_requests field for any student.
* **Realistic Scenario:** Simulate data for a school with 1000 students and a 16:1 student-teacher ratio.

**Data Requirements:**

* **Students data needed:**
  + Fields: student\_id, grade\_level, gender, free\_period\_preference (1st or 4th), course\_requests (comma-separated), sped\_status (Yes/No).
  + Constraints: No duplicate course requests per student. Include SPED status.
* **Teachers data needed:**
  + Fields: teacher\_id, department, gender, assigned sections, available\_periods, sped\_teacher (Yes/No).
  + Constraints: Teachers are only assigned to qualified courses. Honor designated free periods.
* **Courses information needed:**
  + Fields: course\_id, course\_name, department, capacity, sections, grade\_levels (comma-separated), co\_taught (Yes/No).
  + Constraints: Ensure course names and IDs match the constraints. Include co\_taught indicator for co-taught classes.

**Final Objective:**

* **Code Structure:** Write the code in two sections: one for the linear program and another for simulating CSV data.
* **Code Comments:** Include detailed comments explaining how each constraint is met.
* **Validation:** Ensure the program is functional, efficient, and handles constraints correctly.
* **Output Requirements:** Present the final master schedule and a list of students without requests met, along with reasons.