Scheduling Shenanigans at The Vogon Poetry Corner

Four steps to solve this problem.

1. Group all events information for each client.
   Use ***function*** **info_common_client_ids** (***sample_clients, sample_event_occurrences***) to return an array of objects, each object contains client_id, client_name and events array, which contains event_id, start and end information. For example:
   Output:
   [{client_id: 313,
   client_name: "Prophet",
   events: [{event_id: 1029,
   start: "Mon, 29 May 2017 11:00:00 PDT -07:00",
   end: "Mon, 29 May 2017 12:30:00 PDT -07:00"},
   {event_id: 923,
   start: "Mon, 29 May 2017 12:00:00 PDT -07:00",
   end: "Mon, 29 May 2017 13:00:00 PDT -07:00"},
   {…},
   …
   {…}]},
   {…},
   {…},
   …
   {…}]
   Time complexity: O(m*n). m is the length of ***sample_clients***, n is the length of ***sample_event_occurrences.***

2. Find conflicting event pairs under same client.
   In ***function*** **find_rough_Conflicts**(***events***), traverse event information and use ***function*** **intersection_time**(***start1, end1, start2)*** to check if two events have time conflict, if so, add them to result array and return. For example:
   Input: [{event_id: 1029,
   start: "Mon, 29 May 2017 11:00:00 PDT -07:00",
   end: "Mon, 29 May 2017 12:30:00 PDT -07:00"},
   {event_id: 923,
   start: "Mon, 29 May 2017 12:00:00 PDT -07:00",
   end: "Mon, 29 May 2017 13:00:00 PDT -07:00"},
   {…},
   …
   {…}];
   Output: [[1029, 923], [923, 432], [1032, 1035], …]; *//[1029, 923] is conflicting event_id pairs.*
   Time complexity: O(n^2). n is the length of ***events.***

3. Merge sub-arrays if they contain same element.
   function **merge**(***arr***) returns a unique and largest set.
   Input: [[1,2], [3,4], [1,3], [5,6], [6,5]];
   Output: [[1,2,3,4], [5,6]];
   Time complexity: O(n^2). n is the length of ***arr.***

4. ***function*** **findSchedulingConflicts**(***sample_clients, sample_event_occurrences***)
   Organize the output as required.