CS2102

Tutorial 05

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #1
  - Find the number of presentation for each student
  - Find the condition for the most often

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #1
  - Find the number of presentation for each student

    ```
    SELECT COUNT(*) FROM Presenters GROUP BY sid
    ```

    - let's call this CTX
  - Find the condition for the most often

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #1
  - Find the number of presentation for each student

    ```
    SELECT COUNT(*) FROM Presenters GROUP BY sid
    ```

    - let's call this CTX
  - Find the condition for the most often
    - `COUNT(*) >= ALL ( CTX )`

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #1

```
SELECT   sid FROM Presenters
GROUP BY sid
HAVING   COUNT(*) >= ALL (
/* CTX */  SELECT COUNT(*) FROM Presenters GROUP BY sid
         );
```

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #2
  - Find the maximum number of presentation
  - Find all students that has present this number of times

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #2
  - Find the maximum number of presentation

```
SELECT MAX(COUNT(*)) FROM Presenters GROUP BY sid
```

    - but aggregate function cannot be nested!
  - Find all students that has present this number of times

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #2
  - Find the maximum number of presentation

```
SELECT MAX(num) FROM (
  SELECT COUNT(*) AS num FROM Presenters GROUP BY sid
) AS numPress
```

    - but aggregate function cannot be nested!
  - Find all students that has present this number of times

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #2
  - Find the maximum number of presentation

```
SELECT MAX(num) FROM (
  SELECT COUNT(*) AS num FROM Presenters GROUP BY sid
) AS numPres
```

    - but aggregate function cannot be nested!
  - Find all students that has present this number of times

```
SELECT COUNT(*) AS num2 ... WHERE num2 = SELECT MAX(num) ... AS numPres
```

    - too complicated!

Question 7(a)

- Question: Find the sid of all students who have presented the most often.
- How? Attempt #2
  - CTE to the rescue!

```
WITH numPres AS
   ( SELECT sid, COUNT(*) AS num FROM Presenters GROUP BY sid )
SELECT sid FROM numPres
WHERE  num = ( SELECT MAX(num) FROM numPres );
```

  - Why is `num = subquery` allowed?
    - because the subquery is a scalar subquery!

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Find what?
  - From which table?
  - What condition?

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Find what?
    - (sid,sid)
  - From which table?
    - Both from Presenters
    - So Presenters P1 × Presenters P2
  - What condition?
    - P1.sid < P2.sid
    - P1.week = P2.week
    - COUNT(*) >= 5

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Attempt #1

```
SELECT   P1.sid, P2.sid
FROM     Presenters P1, Presenters P2
WHERE    P1.sid < P2.sid AND P1.week = P2.week
  AND    COUNT(*) >= 5;
```

    - COUNT(*) cannot appear in WHERE!

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Attempt #2

```
SELECT   P1.sid, P2.sid
FROM     Presenters P1, Presenters P2
WHERE    P1.sid < P2.sid AND P1.week = P2.week
HAVING   COUNT(*) >= 5;
```

    - Does not satisfy condition
      1. Column A appears in the GROUP BY clause
      2. Column A appears in aggregated expression in SELECT (e.g., MIN(A))
      3. The primary (or candidate) key of RRappears in the GROUP BY clause

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Attempt #3

```
SELECT   P1.sid, P2.sid
FROM     Presenters P1, Presenters P2
WHERE    P1.sid < P2.sid AND P1.week = P2.week
GROUP BY P1.sid, P2.sid
HAVING   COUNT(*) >= 5;
```

  - Satisfy the condition!
    1. Column A appears in the GROUP BY clause
    2. Column A appears in aggregated expression in SELECT (e.g., MIN(A))
    3. The primary (or candidate) key of RRappears in the GROUP BY clause

Question 7(b)

- Question: Find all sid pairs (s1,s2) such that s1 < s2 and both students have presented in the same week for at least 5 different weeks.
  - Attempt #3

```
SELECT   P1.sid, P2.sid
FROM     Presenters P1 JOIN Presenters P2
    ON ( P1.sid < P2.sid AND P1.week = P2.week )
GROUP BY P1.sid, P2.sid
HAVING   COUNT(*) >= 5;
```

  - Can also use JOIN

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - If we have a table of students who did not present for every week
  - We can find student in such table T that has three week consecutive entry

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - If we have a table of students who did not present for every week

```
SELECT sid, week
FROM ( SELECT sid FROM Students ) AS st,
FROM ( SELECT week FROM Presenters ) AS wk
EXCEPT SELECT sid, week FROM Presenters
```

  - We can find student in such table T that has three week consecutive entry

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - If we have a table of students who did not present for every week

```
SELECT sid, week
FROM ( SELECT sid FROM Students ) AS st,
FROM ( SELECT week FROM Presenters ) AS wk
EXCEPT SELECT sid, week FROM Presenters
```

  - We can find student in such table T that has three week consecutive entry

```
SELECT DISTINCT T1.sid FROM T T1, T T2, T T3
WHERE  T1.sid = T2.sid AND T2.sid = T3.sid
  AND  T2.week = T1.week + 1 AND T3.week = T1.week + 2;
```

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - WITH T AS (

```
    SELECT sid, week
    FROM ( SELECT sid FROM Students ) AS st,
    FROM ( SELECT week FROM Presenters ) AS wk
    EXCEPT SELECT sid, week FROM Presenters
) -- CTE to the rescue
    SELECT DISTINCT T1.sid FROM T T1, T T2, T T3
    WHERE  T1.sid = T2.sid AND T2.sid = T3.sid
      AND  T2.week = T1.week + 1 AND T3.week = T1.week + 2;
```

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all
      - Students who have had some presentation

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
```

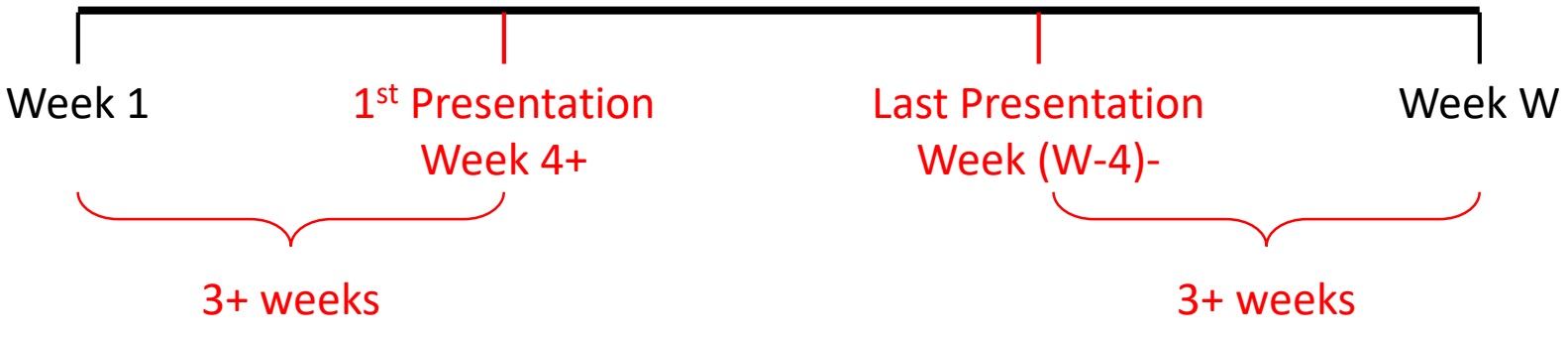
      - Students who have had some presentation

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
```

      - Students who have had some presentation
        - Consider a student S
          - Case 1 & 2: *consider first and last presentation*

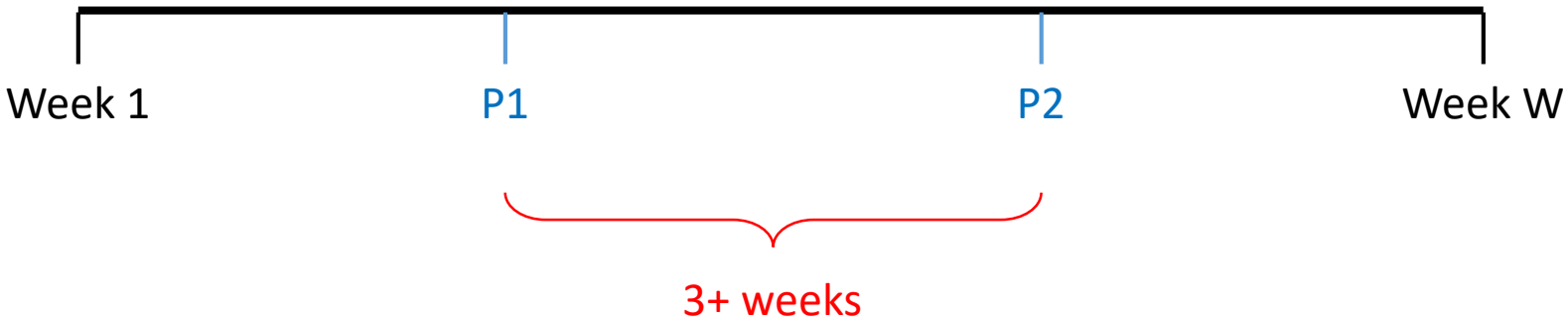


Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
```

      - Students who have had some presentation
        - Consider a student S
          - Case 3: *consider any two presentation P1 and P2 such that P1 < P2*

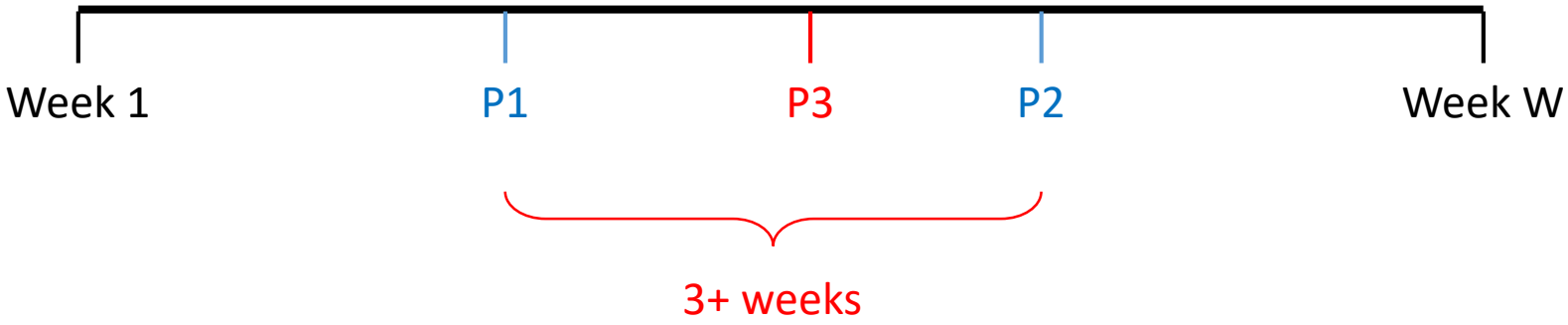


Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
```

      - Students who have had some presentation
        - Consider a student S
          - Case 3: *we check if there's NO P3 in between, then it violates*

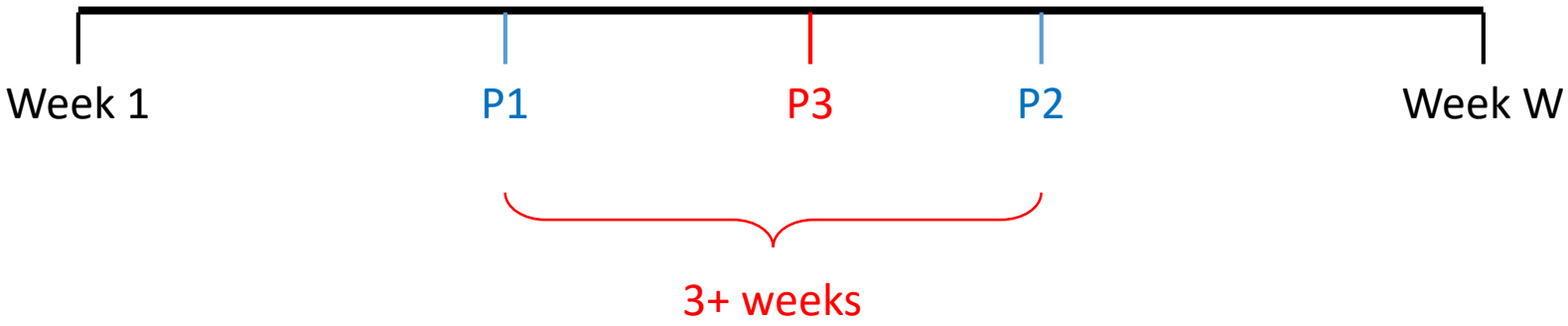


Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2
    - Consider the case that there are at least 3 weeks, we can split into 2 cases
      - Students who have not presented at all

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
```

      - Students who have had some presentation
        - Consider a student S
          - Case 3: *once we check for all pair (P1,P2) and NO violation, we include*



Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Presenters P1 WHERE
```

    - Case 1: *first presentation*
    - Case 2: *last presentation*

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Presenters P1 WHERE
```

  - Case 1: *first presentation*

```
( SELECT MIN(week) FROM Presenters WHERE sid = P1.sid ) >= 4
```

  - Case 2: *last presentation*

```
( SELECT MAX(week) FROM Presenters) –
( SELECT MAX(week) FROM Presenters WHERE sid = P1.sid) >= 4
```

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Presenters P1 WHERE
```

    - Case 3a: *we check if there's NO P3 in between, then it violates*

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Presenters P1 WHERE
```

  - Case 3a: *we check if there's NO P3 in between, then it violates*

```
NOT EXISTS (
  SELECT 1 FROM Presenters P3
  WHERE  P3.sid = P1.sid AND P1.week < P3.week AND P3.week < P2.week
)
```

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Presenters P1 WHERE
```

  - Case 3b: *check for all possible pairs (P1, P2)*

```
EXISTS (
  SELECT 1 FROM Presenters P2
  WHERE  P2.sid = P1.sid AND P2.week – P1.week >= 4
    AND  /* case 3a */
)
```

Question 7(c)

- Question: Find all students who did not present for any *three* consecutive weeks.
  - Solution #2

```
SELECT sid FROM Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
AND    ( SELECT MAX(week) FROM Presenters ) >= 3
UNION
SELECT sid FROM Presenters P1 WHERE
/* Case 1 */ OR /* Case 2 */ OR /* Case 3 */
```

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - How to define priority?
    - Given two students with sid s1 and s2, s1 has a higher priority than s2 if one of the following conditions hold:
      - numQ(s1) < numQ(s2)
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) < lastWk(s2))
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) = lastWk(s2)) and (s1 < s2)

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - How to define priority?
    - Given two students with sid s1 and s2, s1 has a higher priority than s2 if one of the following conditions hold:
      - numQ(s1) < numQ(s2)
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) < lastWk(s2))
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) = lastWk(s2)) and (s1 < s2)
    - We proceed with generating a table for this info first!

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - How to generate this table?
    - Given two students with sid s1 and s2, s1 has a higher priority than s2 if one of the following conditions hold:
      - numQ(s1) < numQ(s2)
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) < lastWk(s2))
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) = lastWk(s2)) and (s1 < s2)

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - How to generate this table?
    - If student has presented, find numQ and lastWk
    - If student has NOT presented, numQ = 0 and lastWk = 0

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - How to generate this table?
    - If student has presented, find numQ and lastWk

```
SELECT sid, COUNT(*) AS numQ, MAX(week) AS lastWk
FROM   Presenters GROUP BY sid
```

    - If student has NOT presented, numQ = 0 and lastWk = 0

```
SELECT sid, 0 AS numQ, 0 AS lastWk
FROM  Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
```

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - Priority table

```
WITH Priority AS (
    SELECT sid, COUNT(*) AS numQ, MAX(week) AS lastWk
    FROM   Presenters GROUP BY sid
    UNION
    SELECT sid, 0 AS numQ, 0 AS lastWk
    FROM  Students WHERE sid NOT IN ( SELECT sid FROM Presenters )
)
```

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - Given Priority table
    - Find the pair such that
      - numQ(s1) < numQ(s2)
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) < lastWk(s2))
      - (numQ(s1) = numQ(s2)) and (lastWk(s1) = lastWk(s2)) and (s1 < s2)

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - Given Priority table
    - s3 has higher priority than s1 if:
      - numQ(s3) < numQ(s1)

      `s3.numQ < s1.numQ`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) < lastWk(s1))

      `s3.numQ = s1.numQ AND s3.lastWk < s1.lastWk`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) = lastWk(s1)) and (s3 < s1)

      `s3.numQ = s1.numQ AND s3.lastWk = s1.lastWk ANS s3.sid < s1.sid`

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - Given Priority table
    - s3 has higher priority than s1 if: s3 must not exists!
      - numQ(s3) < numQ(s1)

      `s3.numQ < s1.numQ`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) < lastWk(s1))

      `s3.numQ = s1.numQ AND s3.lastWk < s1.lastWk`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) = lastWk(s1)) and (s3 < s1)

      `s3.numQ = s1.numQ AND s3.lastWk = s1.lastWk ANS s3.sid < s1.sid`

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - Given Priority table
    - s3 has higher priority than s1 if: then do the same between s3 and s2
      - numQ(s3) < numQ(s1)

      `s3.numQ < s1.numQ`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) < lastWk(s1))

      `s3.numQ = s1.numQ AND s3.lastWk < s1.lastWk`
      - (numQ(s3) = numQ(s1)) and (lastWk(s3) = lastWk(s1)) and (s3 < s1)

      `s3.numQ = s1.numQ AND s3.lastWk = s1.lastWk ANS s3.sid < s1.sid`

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.

```
WITH Priority AS ( ... )
SELECT S1.sid, S2.sid FROM Priority S1, Priority S2
WHERE S1.sid < S2.pid AND NOT EXISTS (
  SELECT 1 FROM Priority S3
  WHERE S3.sid <> S1.sid AND S3.sid <> S2.sid
    AND /* priority(S3) < priority(S1) OR priority(S2) */
); -- it is too long to display here
```

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - A little bit of trickery
    - Note that the priority induces total order
    - in other words, there will be exactly ONLY 2 students that satisfies this condition
    - So we can order them!

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - A little bit of trickery
    - Ordering
      - Order by numQ first
        - because first condition is `numQ(s1) < numQ(s2)`
      - Order by lastWk next
        - because if `numQ(s1) = numQ(s2)` then `lastWk(s1) < lastWk(s2)`
      - Order by sid last
        - and if `lastWk(s1) = lastWk(s2)` too, then `s1 < s2`

Question 7(d)

- Question: Find all sets of two students S to be presenters for the next tutorial such that none of the students in (Students - S) has higher priority than any of the students in S.
  - A little bit of trickery

```
SELECT   sid FROM Priority
ORDER BY numQ, lastWk, sid
LIMIT    2  -- only top 2 priority taken
```

  - Then we can choose the smaller sid as 1st attribute and larger sid as 2nd

```
SELECT MIN(sid) AS sid1, MAX(sid) AS sid2
FROM   The_table_above -- another CTE
```

Question 7(e)

- Question: We say that the Presenters table is *consistent* if it satisfies both the following conditions:
  1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
  2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
  - Write SQL query to output 0 if Presenters is consistent; and 1 otherwise.

Question 7(e)

- Question: We say that the Presenters table is *consistent* if it satisfies both the following conditions:
  1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
  2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
  - Write SQL query to output 0 if Presenters is consistent; and 1 otherwise.
    - Case analysis!

Question 7(e)

1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
   - If we can find the largest value of W
   - it must be equal to the number of distinct week!

Question 7(e)

1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
   - If we can find the largest value of W

     ```
     ( SELECT MAX(week) FROM Presenters )
     ```

   - it must be equal to the number of distinct week!

     ```
     ( SELECT COUNT(DISTINCT week) FROM Presenters )
     ```

Question 7(e)

1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
   - With some magic from scalar subquery

```
( SELECT MAX(week) FROM Presenters )

=
( SELECT COUNT(DISTINCT week) FROM Presenters )
```

Question 7(e)

2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
   - Consider each week w_i, we have a similar condition
     - The largest qnum in w_i
     - must be equal to number of distinct qnum for w_i

Question 7(e)

2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
   - Consider each week w_i, we have a similar condition
     - The largest qnum in w_i

       MAX(qnum)
     - must be equal to number of distinct qnum for w_i

       COUNT(DISTINCT qnum)

Question 7(e)

2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
   - Consider each week w_i, we have a similar condition
     - Consistency check

       MAX(qnum)

       =

       COUNT(DISTINCT qnum)

Question 7(e)

- Question: We say that the Presenters table is *consistent* if it satisfies both the following conditions:
  1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
  2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
  - Write SQL query to output 0 if Presenters is consistent; and 1 otherwise.
  - Given the two checks, we check for *inconsistency!*

Question 7(e)

1. There must exist at least one tuple t in Presenters with t.week = w for each value of w in {1, 2, ..., W}
   - To check inconsistencies

```
( SELECT MAX(week) FROM Presenters )

<>
( SELECT COUNT(DISTINCT week) FROM Presenters )
```

Question 7(e)

2. For each value i in {1, 2, ..., W}, if the maximum qnum value for week i in Presenters is Q, then there must exist at least one tuple t in Presenters with t.week = i and t.qnum = j for each value of j in {1, 2, ..., Q}.
   - Consider each week w_i, we have a similar condition
     - Inconsistency check

       MAX(qnum)

       <>

       COUNT(DISTINCT qnum)

Question 7(e)

- Answer:

```
SELECT 1 AS num FROM Presenters
WHERE  ( SELECT MAX(week) FROM Presenters )
       <>
       ( SELECT COUNT(DISTINCT week) FROM Presenters)
    OR EXISTS (
         SELECT   1 FROM Presenters
         GROUP BY week
         HAVING   MAX(qnum) <> COUNT(DISTINCT qnum)
       );
```

Question 7(e)

- Answer: can be null!

```
SELECT 1 AS num FROM Presenters
WHERE  ( SELECT MAX(week) FROM Presenters )
       <>
       ( SELECT COUNT(DISTINCT week) FROM Presenters)
    OR EXISTS (
         SELECT   1 FROM Presenters
         GROUP BY week
         HAVING   MAX(qnum) <> COUNT(DISTINCT qnum)
       );
```

Question 7(e)

- Answer: if null then 0

```
WITH Status AS ( ... )
SELECT COALESCE(MIN(num), 0) FROM Status;
```