1. Consider the table `Exams(sid, cid, score)`, such that:

   - Each `sid` is an integer and represents a student ID.
   - Each `cid` is an integer and represents a course ID.
   - Each `score` is an integer and represents a final exam score of a student in a course.

   Write a function `max_min` with the following properties:

   - It has an input parameter `stu_id`, which is an integer.
   - It has two output parameters `max_cid` and `min_cid`, both of which are integers.
   - It examines the records in `Exams` whose `sid` is equal to `stu_id` and identifies the two records among them with the largest and smallest `score` where *ties are broken arbitrarily*. For the record with the largest `score`, its `cid` is assigned to `max_cid`. For the record with the smallest `score`, its `cid` is assigned to `min_cid` *only if it is smaller than the largest score*. Otherwise, `min_cid` is set to `NULL`.

   A template for `max_min` is provided below:

```
1 CREATE OR REPLACE FUNCTION max_min (IN stu_id INT, OUT max_cid
     INT, OUT min_cid INT)
2 RETURNS RECORD AS $$
3 DECLARE
4   max_score INT;
5   min_score INT;
6 BEGIN
7   /* write your code here */
8 END;
9 $$ LANGUAGE plpgsql;
```

Solution:

```
1  CREATE OR REPLACE FUNCTION max_min (IN stu_id INT, OUT
      max_cid INT, OUT min_cid INT)
2  RETURNS RECORD AS $$
3  DECLARE
4    max_score INT;
5    min_score INT;
6  BEGIN
7    SELECT cid, score INTO max_cid, max_score
8    FROM   Exams
9    WHERE  sid = stu_id
10     AND  score =
11           (SELECT MAX(score) FROM Exams WHERE sid = stu_id);
12
13   SELECT cid, score INTO min_cid, min_score
14   FROM   Exams
15   WHERE  sid = stu_id
16     AND  score =
17           (SELECT MIN(score) FROM Exams WHERE sid = stu_id);
18
```

```
19    IF max_score = min_score THEN
20      min_cid := NULL;
21    END IF;
22  END;
23  $$ LANGUAGE plpgsql;
```

2. Consider the same table `Exams(sid, cid, score)` from the previous question. Write a function `revised_avg` with the following properties:

   - It has an input parameter `stu_id`, which is an integer.
   - It has an output parameter `r_avg`, which is a numeric.
   - It examines the records in `Exams` whose `sid` is equal to `stu_id`. If there are *at least* 3 such records, the function returns the average `score` of these records but *excludes* one record with the highest `score` with *ties broker arbitrarily* as well as one record with the lowest `score` with *ties broker arbitrarily*. If there are fewer than 3 such records, the function returns `NULL`.

   A template for `revised_avg` is provided below:

```
1 CREATE OR REPLACE FUNCTION revised_avg (IN stu_id INT, OUT r_avg
      FLOAT)
2 RETURNS FLOAT AS $$
3   /* write your code here */
4 $$ LANGUAGE plpgsql;
```

Solution:

```
1  CREATE OR REPLACE FUNCTION revised_avg (IN stu_id INT, OUT
       r_avg FLOAT)
2  RETURNS FLOAT AS $$
3  DECLARE
4    max_score INT;
5    min_score INT;
6    sum_score FLOAT;
7    ctx_score INT;
8  BEGIN
9    SELECT MAX(score), MIN(score), SUM(score), COUNT(score)
       INTO
10         max_score , min_score , sum_score , ctx_score
11   FROM   Exams
12   WHERE  sid = stu_id;
13
14   IF ctx_score < 3 THEN
15     r_avg := NULL;
16   ELSE
17     r_avg := (sum_score - max_score - min_score)/(ctx_score
       -2);
18   END IF;
19 END;
20 $$ LANGUAGE plpgsql;
```

3. Consider the same table `Exams(sid, cid, score)` from the first question as well as the concept of "revised average `score`" in the previous question. Write a function `list_r_avg` that returns the `sid` of each student in `Exams` along with their revised average `score`. For simplicity, we assume that all `sid` in `Exams` are non-negative integers.

   A template for `list_r_avg` is provided below:

```
1  CREATE OR REPLACE FUNCTION list_r_avg ()
2  RETURNS TABLE (stu_id INT, ravg FLOAT) AS $$
3  DECLARE
4    curs CURSOR FOR (SELECT sid, score FROM Exams ORDER BY sid);
5    /* add other variables here */
6  BEGIN
7    /* write your code here */
8  END;
9  $$ LANGUAGE plpgsql;
```

Solution:

```
1  CREATE OR REPLACE FUNCTION list_r_avg ()
2  RETURNS TABLE (stu_id INT, ravg FLOAT) AS $$
3  DECLARE
4    curs CURSOR FOR (SELECT sid, score FROM Exams ORDER BY sid)
       ;
5    r RECORD;
6    max_score INT;
7    min_score INT;
8    sum_score FLOAT;
9    ctx_score INT;
10 BEGIN
11   stu_id = -1;
12   OPEN curs;
13   LOOP
14     FETCH curs INTO r;
15     IF r.sid <> stu_id OR NOT FOUND THEN
16       IF stu_id <> -1 THEN
17         IF (ctx_score < 3) THEN
18           ravg := NULL;
19         ELSE
20           ravg := (sum_score - max_score - min_score)/(
     ctx_score-2);
21         END IF;
22         RETURN NEXT;
23       END IF;
24       EXIT WHEN NOT FOUND;
25       stu_id := r.sid;
26       max_score := r.score;
27       min_score := r.score;
28       sum_score := r.score;
29       ctx_score := 1;
30     ELSE
31       sum_score := sum_score + r.score;
32       ctx_score := ctx_score + 1;
33       IF r.score > max_score THEN max_score := r.score; END
     IF;
```

```
34        IF r.score < min_score THEN min_score := r.score; END
      IF;
35       END IF;
36     END LOOP;
37     CLOSE curs;
38     RETURN;
39 END;
40 $$ LANGUAGE plpgsql;
```

4. Consider the same table `Exams(sid, cid, score)` from the first question. Write a function `list_scnd_highest` that returns the `sid` of each student in `Exams` along with their second highest `score` with *ties broken arbitrarily*. If the student only have one `score`, then the second highest `score` is `NULL`. For simplicity, we assume that all `sid` in `Exams` are non-negative integers.

   A template for `list_scnd_highest` is provided below:

```
CREATE OR REPLACE FUNCTION list_scnd_highest ()
RETURNS TABLE (stu_id INT, scnd_highest INT) AS $$
  /* write your code here */
$$ LANGUAGE plpgsql;
```

Solution:

```
CREATE OR REPLACE FUNCTION list_scnd_highest ()
RETURNS TABLE (stu_id INT, scnd_highest INT) AS $$
DECLARE
  curs CURSOR FOR (SELECT sid, score FROM Exams ORDER BY sid)
   ;
  r RECORD;
  max_score INT;
  ctx_score INT;
BEGIN
  stu_id = -1;
  OPEN curs;
  LOOP
    FETCH curs INTO r;
    IF r.sid <> stu_id OR NOT FOUND THEN
      IF stu_id <> -1 THEN
        IF (ctx_score < 2) THEN
          scnd_highest := NULL;
        END IF;
        RETURN NEXT;
      END IF;
      EXIT WHEN NOT FOUND;
      stu_id := r.sid;
      max_score := r.score;
      scnd_highest := -1;
      ctx_score := 1;
    ELSE
      ctx_score := ctx_score + 1;
      IF r.score > max_score THEN
        scnd_highest := max_score;
        max_score := r.score;
      ELSEIF r.score > scnd_highest THEN
        scnd_highest := r.score;
      END IF;
    END IF;
  END LOOP;
  CLOSE curs;
  RETURN;
END;
$$ LANGUAGE plpgsql;
```