Q1. Consider a 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**s are equal to stu_id, and identifies the two records among them with the largest and smallest **score**s, respectively. (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**, if its **score** is smaller than the largest **score**, then its **cid** is assigned to min_cid; otherwise, min_cid is set to NULL.

A template for **max_min** is provided below.

```
CREATE OR REPLACE FUNCTION max_min( IN stu_id integer, OUT max_cid integer, OUT min_cid integer )
RETURNS RECORD as $$
DECLARE
        max_score integer;
        min_score integer;
BEGIN
        // Write your code here
END;
$$ LANGUAGE plpgsql;
```

Q1 Solution:

```
CREATE OR REPLACE FUNCTION max_min( IN stu_id integer, OUT max_cid integer, OUT min_cid integer )
RETURNS RECORD AS $$
DECLARE
        max_score integer;
        min_score integer;
BEGIN
        SELECT cid, score INTO max_cid, max_score
        FROM exams
        WHERE sid = stu_id AND score =
                (SELECT MAX(score) FROM exams WHERE sid = stu_id);

        SELECT cid, score INTO min_cid, min_score
        FROM exams
        WHERE sid = stu_id AND score =
                (SELECT MIN(score) FROM exams WHERE sid = stu_id);

        IF max_score = min_score THEN
                min_cid := NULL;
        END IF;
END;
$$ LANGUAGE plpgsql;
```

Q2. Consider the **exams** table in Question Q1. Write a function **revised_avg** that returns the "revised average score" of a given student, with the following properties:

- The function has an input parameter stu_id, which is an integer.
- The function has one output parameter r_avg, which is a numeric.
- The function examines the records in **exams** whose **sid**s are equal to stu_id. If there exist at least 3 such records, the function returns the average score of these records, excluding one record with the highest score (with ties broken arbitrarily) and one record with the lowest score (with ties broken arbitrarily). If there exist less than 3 such records, the function returns NULL.

A template for **revised_avg** is provided below.

```
CREATE OR REPLACE FUNCTION revised_avg( IN stu_id integer, OUT r_avg float )
RETURNS float as $$

// Write your code here

$$ LANGUAGE plpgsql;
```

Q2 Solution:

```
CREATE OR REPLACE FUNCTION revised_avg( IN stu_id integer, OUT r_avg float )
RETURNS float AS $$
DECLARE
        max_score integer;
        min_score integer;
        sum_score float;
        count_score float;
BEGIN
        SELECT MAX(score), MIN(score), SUM(score), COUNT(score) INTO
                max_score, min_score, sum_score, count_score
        FROM exams
        WHERE sid = stu_id;

        IF count_score < 3 THEN
                r_avg := NULL;
        ELSE
                r_avg := (sum_score - max_score - min_score) / (count_score - 2);
        END IF;
END;
$$ LANGUAGE plpgsql;
```

Q3. Consider the **exams** table in Question Q1 and the concept of "revised average score" in Question Q2. Write a function **list_r_avg** that returns the **sid** of each student in **exams** along with his/her revised average score. For simplicity, we assume that all **sid**s in **exams** are non-negative integers.

A template for **list_r_avg** is provided below.

```
CREATE OR REPLACE FUNCTION list_r_avg()
RETURNS TABLE ( stu_id integer, ravg float ) AS $$
DECLARE
        curs CURSOR FOR (SELECT sid, score from exams order by sid);
        // write your code here
BEGIN
        // write your code here
END;
$$ LANGUAGE plpgsql;
```

Q3 Solution:

```
CREATE OR REPLACE FUNCTION list_r_avg()
RETURNS TABLE ( stu_id integer, ravg float ) AS $$
DECLARE
	curs CURSOR FOR (SELECT sid, score from exams order by sid);
	r record;
	max_score integer;
	min_score integer;
	sum_score float;
	count_score integer;
BEGIN
	stu_id = -1;
	OPEN curs;
	LOOP
		FETCH curs INTO r;
		IF r.sid <> stu_id OR NOT FOUND THEN
			IF stu_id <> -1 THEN
				IF (count_score < 3) THEN
					ravg := NULL;
				ELSE
					ravg := (sum_score - max_score - min_score) / (count_score - 2);
				END IF;
				RETURN NEXT;
			END IF;
			EXIT WHEN NOT FOUND;
			stu_id := r.sid;
			max_score := r.score;
			min_score := r.score;
			sum_score := r.score;
			count_score := 1;
		ELSE
			sum_score := sum_score + r.score;
			count_score := count_score + 1;
			IF r.score > max_score THEN max_score := r.score;
			END IF;
			IF r.score < min_score THEN min_score := r.score;
			END IF;
		END IF;
	END LOOP;
	CLOSE curs;
	RETURN;
END;
$$ LANGUAGE plpgsql;
```

Q4. Consider the **exams** table in Question Q1. Write a function **list_scnd_highest** that returns the **sid** of each student in **exams** along with his/her 2nd highest score. (Ties are broken arbitrarily.) If the student has fewer than 2 scores, then **list_scnd_highest** returns NULL as his/her 2nd highest score. For simplicity, we assume that all **sid**s and **score**s in **exams** are non-negative integers.

Q4 Solution:

```
CREATE OR REPLACE FUNCTION list_scnd_highest()
RETURNS TABLE ( stu_id integer, scnd_highest integer ) AS $$
DECLARE
        curs CURSOR FOR (SELECT sid, score from exams order by sid);
        r record;
        max_score integer;
        count_score integer;
BEGIN
        stu_id = -1;
        OPEN curs;
        LOOP
                FETCH curs INTO r;
                IF r.sid <> stu_id OR NOT FOUND THEN
                        IF stu_id <> -1 THEN
                                IF (count_score < 2) THEN
                                        scnd_highest := NULL;
                                END IF;
                                RETURN NEXT;
                        END IF;
                        EXIT WHEN NOT FOUND;
                        stu_id := r.sid;
                        max_score := r.score;
                        scnd_highest := -1;
                        count_score := 1;
                ELSE
                        count_score := count_score + 1;
                        IF r.score > max_score THEN
                                scnd_highest := max_score;
                                max_score := r.score;
                        ELSEIF  r.score > scnd_highest THEN
                                scnd_highest := r.score;
                        END IF;
                END IF;
        END LOOP;
        CLOSE curs;
        RETURN;
END;
$$ LANGUAGE plpgsql;
```