SQL Question

We have:

- a machine learning binary classifier which takes as input an image and outputs the image quality score (from 0 to 1, where scores closer to 0 represent low-quality images, and scores closer to 1 represent high-quality images).
- a SQL table containing 1M unlabeled images. We run each of these images through our machine learning model to get float scores from 0 to 1 for each image.

We want to prepare a new training set with some of these unlabeled images. An example of unlabeled_image_predictions (1M rows) is shown below:

image_id	**score**
242	0.23
123	0.92
248	0.88
...	...

Our sampling strategy is to order the images in decreasing order of scores and sample every 3rd image starting with the first from the beginning until we get 10k positive samples. And we would like to do the same in the other direction, starting from the end to get 10k negative samples.

Task: Write a SQL query that performs this sampling and creates the expected output ordered by image_id with integer columns image_id, weak_label.

Feel free to develop in DB-Fiddle, your own SQL sandbox, or writing the query directly in your submission. If using DB-Fiddle with PostgresSQL v12 (set the database engine on the top-left), you may find this example input table of 50 rows useful (can be pasted into the “Schema SQL” text box). You may also find ROW_NUMBER() or helpers with similar functionality useful.

```
CREATE TABLE IF NOT EXISTS unlabeled_image_predictions (
  image_id int,
  score float
);
INSERT INTO unlabeled_image_predictions (image_id, score) VALUES
('828','0.3149'), ('705','0.9892'), ('46','0.5616'), ('594','0.7670'), ('232','0.1598'), ('524','0.9876'), ('306','0.6487'),
('132','0.8823'), ('906','0.8394'), ('272','0.9778'), ('616','0.1003'), ('161','0.7113'), ('715','0.8921'), ('109','0.1151'),
('424','0.7790'), ('609','0.5241'), ('63','0.2552'), ('276','0.2672'), ('701','0.0758'), ('554','0.4418'), ('998','0.0379'),
('809','0.1058'), ('219','0.7143'), ('402','0.7655'), ('363','0.2661'), ('624','0.8270'), ('640','0.8790'), ('913','0.2421'),
('439','0.3387'), ('464','0.3674'), ('405','0.6929'), ('986','0.8931'), ('344','0.3761'), ('847','0.4889'), ('482','0.5023'),
('823','0.3361'), ('617','0.0218'), ('47','0.0072'), ('867','0.4050'), ('96','0.4498'), ('126','0.3564'), ('943','0.0452'),
('115','0.5309'), ('417','0.7168'), ('706','0.9649'), ('166','0.2507'), ('991','0.4191'), ('465','0.0895'), ('53','0.8169'),
('971','0.9871');
```