Interactive - 29 - m to n relationship (fk to join table to fk)

Our model has a m to n (m:n) relationship in it. This is between ct_homeowrk and ct_tags.

This relationship requires an intermediate table. Each side has a relationship to the intermediate table.

ct_tag associates a tag_word with a tag_id (Shown below - the table was created in the script in homework 26 - you don't need to run it.)

```
CREATE TABLE ct_tag (
    tag_id uuid DEFAULT uuid_generate_v4() not null primary key,
    tag_word text not null
);

CREATE UNIQUE INDEX ct_tag_p1 on ct_tag ( tag_word );
```

To get a list of the words do a select from this table.

```
SELECT tag_word from ct_tag order by tag_word;
```

Note that tag_word has a unique index on it. This prevents any duplicate tags.

The join table has just IDs in it. To make it efficient it has indexes that go in both directions - this allows searching from ct_tags back to ct_homework and from ct_homework to ct_tags. One of the unique indexes is the multi-field primary key with homework_id, tag_id in it.. This is listed at the bottom of the create table. The second one is a unique index.

(shown below - do not run, just for reference)

```
CREATE TABLE ct_tag_homework (
    tag_id      uuid not null,
    homework_id uuid not null,
    primary key ( homework_id, tag_id )
);

CREATE UNIQUE INDEX ct_tag_homework_u1 on ct_tag_homework ( tag_id, homework_id );
```

The ct_homework table (show for reference) is:

```
CREATE TABLE ct_homework (
      homework_id               uuid DEFAULT uuid_generate_v4() not null primary key
    , homework_no               text not null
    , points_avail              int not null default 10
    , video_url                 text not null
    , video_img                 text not null
    , lesson_body               JSONb not null  -- body, html, text etc.
```

```
);

CREATE INDEX ct_homework_p1 on ct_homework ( homework_no );
```

Now do a select from the join table.

```
SELECT * from ct_tag_homework order by 1, 2;
```

Creative Homework - Assignment 02 - program on your computer.

This Interactive assignment has the tables in it for your Assignment 02. The assignments to be written in Python and generate SQL as output.

For this Interactive you get to write the query that will join the tables together. Some data has already been loaded to the tables.

Construct a select that joins all 3 tables together. The select should have `homework_title` and `points_avail` from the ct_homework table in it. It should also have the `tag_word` from ct_tag and it should join from ct_homework to ct_tag_homework to ct_tag. The output should be an ordered list of homework and all the tags that are associated with each homework. For example in homework 29 you should have 3 tags, "m to n", "m:n" and "m:n relationship" as the tags. Also have the tags sorted alphabetically. This means that "m to n" will be the last of the 3 tags for homework 29.

Hints: You need to sort by a column that you do not need to return. This is the homework_no. So you need to have a sub-select with the order by on the inside select. The example of a join is in homework 19 where we use a sub-select and the order by on the inside select.

In homework 11 we have an example of jointing to multiple tables at once.

The homework on inner joins is 18.

It's important that the sort order by homework_no be done as an integer, not as text. You will need to use a `::int` to convert the text field to an integer field.

The correct output data should look similar to:

```
                                    homework_title                                    |
--------------------------------------------------------------------------------------+-
  Interactive - 01 - Create Table                                                     |
  Interactive - 01 - Create Table                                                     |
  Interactive - 01 - Create Table                                                     |
  Interactive - 01 - Create Table                                                     |
  Interactive - 01 - Create Table                                                     |
  Interactive - 02 - Insert data into "name_list"                                     |
  Interactive - 02 - Insert data into "name_list"                                     |
```

```
...
...
...
  Interactive - 08 - create unique id and a primary key                                |
  Interactive - 09 - add a table with state codes                                      |
  Interactive - 09 - add a table with state codes                                      |
  Interactive - 09 - add a table with state codes                                      |
  Interactive - 10 - add a index on the name table                                     |
  Interactive - 11 - add a index on the name table that is case insensitive.           |
...
...
...
```

Tags: m:n,“m to n”,“m:n”,“m:n relationship”

Validate: SQL-Select,“select ‘PASS’ as x”