Interactive - 30 - 1 to ordered list relationship (fk to uk with sequence)

The model has a 1 to a list of rows. This is a more unusual relationship because the list can be ordered by time. Usually relationships in a SQL database are not inherently ordered. In this case we use a sequence generated number to make the set of rows in a specific order.

Taken from hw26_3.sql that you have already run. You don't have to run it again. The first step is to create a sequence. This one is `t_log_seq`. Each time a value is used from the sequenced it increments.

```
CREATE SEQUENCE t_log_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
  CACHE 1;
```

For the field "seq' in the table we use the sequence. This is the `nextval('t_log_seq'::regclass)` default value. When we insert we allow the default value to fill `seq`. Our join is a 1:n join based on user_id to the user. Each time a security event happens we insert the type of event to this table. In a complete security system this kind of a"log" has a trigger on it and for certain events it would generate an email to the user. An example is the creation of a device/application account or the change of the account password. Some events would just be logged for later statistical usage - like a login or logout event.

Also taken from hw26_3.sql that you have already run. You don't have to run it again.

```
CREATE TABLE "t_ymux_user_log" (
      "id"                 uuid DEFAULT uuid_generate_v4() not null primary key
    , "user_id"            uuid    -- if null then a failed event
    , "seq"                bigint DEFAULT nextval('t_log_seq'::regclass) NOT NULL
    , "activity_name"      text
    , "updated"            timestamp
    , "created"            timestamp default current_timestamp not null
);

create index "t_ymux_user_log_p1" on "t_ymux_user_log" ( "user_id", "seq" );
create index "t_ymux_user_log_p2" on "t_ymux_user_log" ( "user_id", "created" );
```

Run a simple select on t_ymux_user_log. You should have at least 2 rows in it.

```
select * from t_ymux_user_log;
```

Assignment 03 - Create 2 tables.

50 points.

Create a table a “issue” table that has a text body column called “body”. The table needs to have a UUID primary key that is automatically generated. Then create a child table, “issue_note” that has a list of comments with dates associated with the “issue” table. This list of comments needs to be ordered using a “seq” column. Create a sequence to support this column. Call the sequence “t_issue_note_seq”. Both tables need to have created/updated columns that are date stamps.

Turn in the source code for creating the tables with some inserts that put data into the tables and a select that verifies an issue and a set of issue notes.

Don’t loose your source code for these tables. We will use these tables in the assignment on key word search later in the class.

Tags: “1 to list”,“list of rows”

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