You are developing software for a children's shop.

User story

I want to give a five-star rating to a product.

Notes

- Currently, there is an up/down vote in the system.
- The `vote` table should be renamed to `rating`
- All downvotes should be changed to rating 1
- All upvotes should be changed to rating 5
- `product.votes` should be changed to `product.average_rating`

Available Code

You check the database to find that the following relevant tables exist:

1. `product`
2. `category`
3. `vote`

One interesting thing is, the product table has an integer `votes` field that holds the current approximate votes for the product. A stored procedure (SP) `recalculate_product_votes` is there to calculate the votes from the `vote` table and store them in the `votes` field. A scheduled job executes the SP every midnight.

Now that `vote` is being replaced with `rating`, the product table should have an `average_rating` column instead of the `votes` column. The average rating of a product is simply the average of all ratings given to a product.

An initialization script and seed data are available.

Initial Tasks

The tasks are not in any specific order. You need to figure out in which order you should do the tasks.

- Execute the seed script
- Write a migration script to complete the database end of the user story
- Execute the initialization script

Notes

1. You do not have to deal with the scheduled job
2. Write just one migration script. This is not necessarily best practice or anything. This is only for the easy evaluation of the task.
3. This is a preliminary task for the topic so you do not have to maintain the full workflow of EDD. For example, you do not have to worry about the `changelog` table. Just write the migration script.

The file name should use this template: `your-2-digit-id.sql`

Lab

To complete the task, you will need to know about

1. Migrating a database using Evolutionary Database Design. Keep in mind that both schema and data are part of your database.
2. Denormalization of database with attribute/column redundancy.

Task

You are working on the Kids Shop application. The application is already deployed so there are some migration scripts available. You will need to write some more migration scripts to accommodate requirement changes. Please look into the existing scripts to see what needs to be in the migration scripts.

Please check the naming convention of tables and other elements used in the given script. Use the same convention. Always use the full form of the names. For example, `product_id`, not `pid`. Some names are explicitly mentioned in the tasks, use the same name. Not complying with the naming convention may cause you to lose marks.

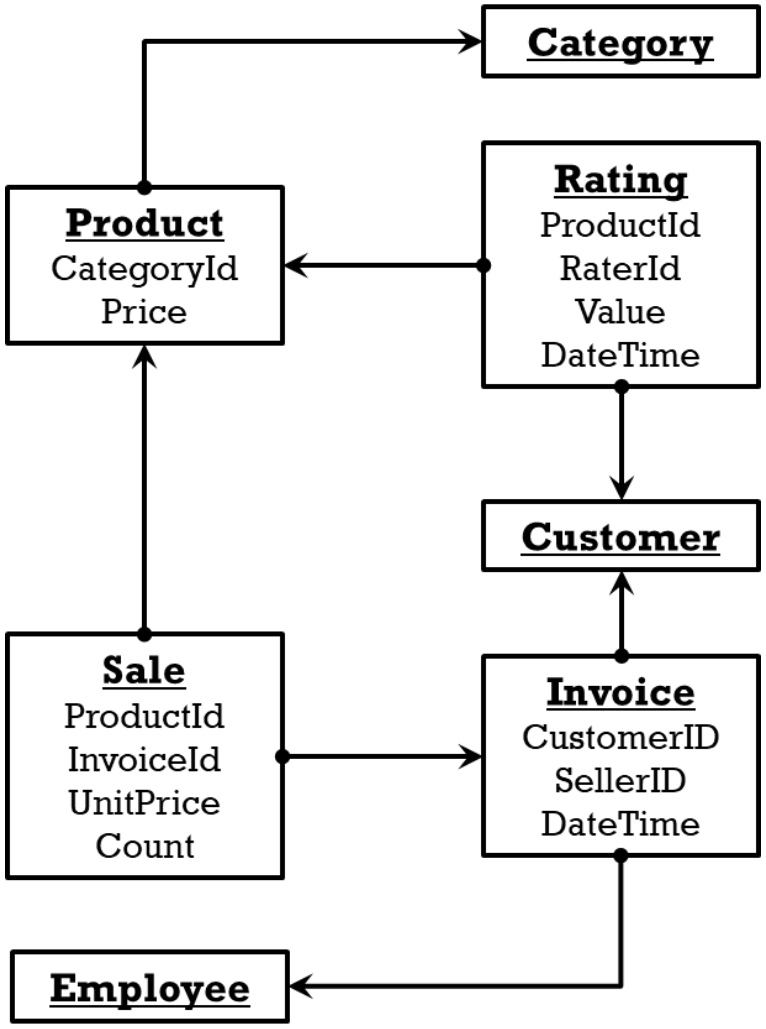
From the previous task, migration scripts will be used to complete the task.

Each of the main tasks should be a separate migration script. For example, Task B1 should be in a migration script that should include subtasks B1a and B1b.

Set A: Add more tables

For this set, please refer to page 5 of the *Database Denormalization* slide. We need the tables and columns as shown in the slide. For simplicity, the figure is given below as well. Use the naming conventions as mentioned above.

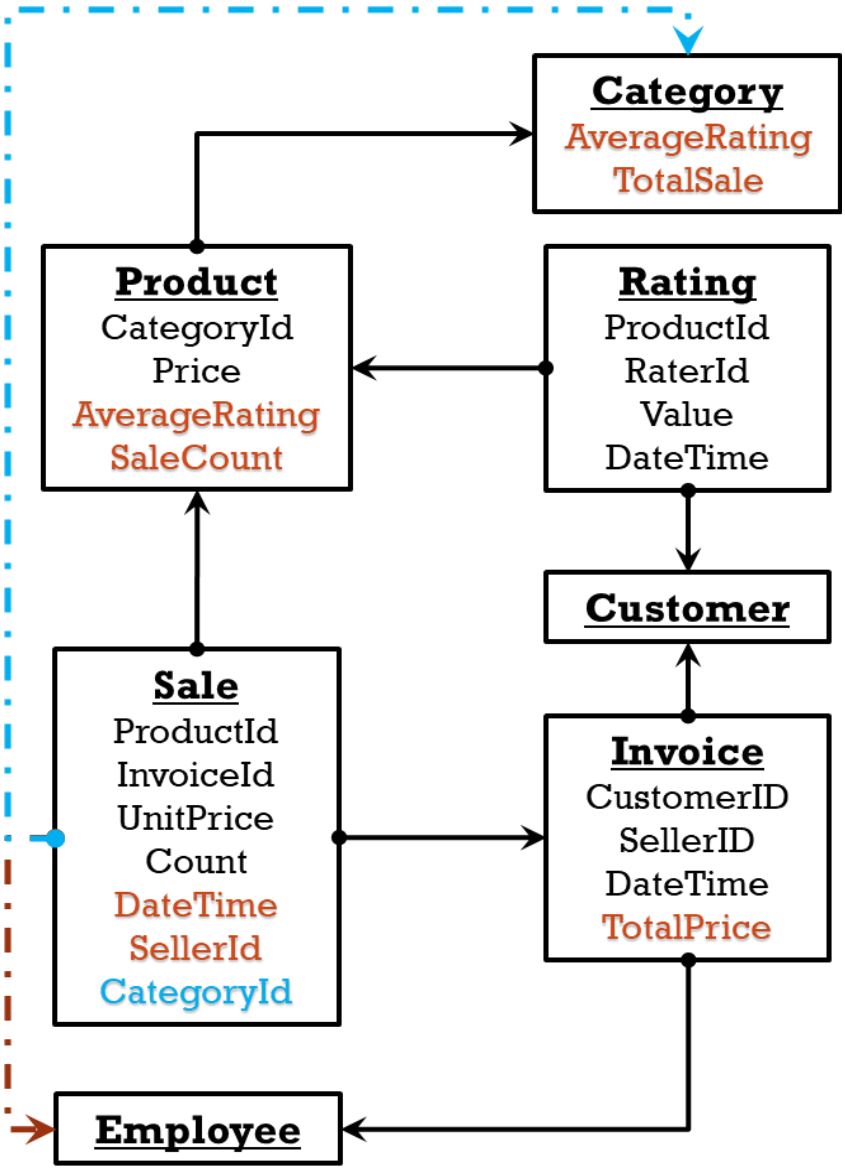
- Task A1: A rating should be given by a customer. A `customer` table is therefore required. A customer should have a `name`. Additionally, the time when the rating was given also needs to be stored (column name: `rating_timestamp`).
- Task A2: Product sale information should be stored (table name: `sale`). Multiple products can be sold together, therefore an `invoice` table should also exist. Also, products are sold by an `employee` and bought by a `customer`.



Set B: Column redundancy

For this set, please refer to pages 6 and 7 of the *Database Denormalization* slide. Please note that the redundancy is somewhat simplified in the presentation. You may want to design differently to address the real scenario.

- Task B1: optimize the database for the query: "find top `n` products by average rating". You do not have to implement the query itself. Just complete the subtasks below.
    - B1a: Add redundant column(s) in the appropriate tables to implement the optimized query.
    - B1b: Write a stored procedure (name `add_rating`) that takes input a `product_id`, a `rating_value`, and a `customer_id` to add a rating to a product accordingly. This is to ensure that the redundant column is consistent with the main data.
    - B1c: Write a stored procedure that takes a `product_id` as input and returns its average rating without joining any tables.
- Task B2: optimize the database for the query: "Find total sale per category for a given employee by joining `sale` and `category` tables only. You do not have to implement the query itself. Just complete the subtasks below.
    - B2a: Add the redundant column(s) in the appropriate tables to implement the optimized query.
    - B2b: Write a stored procedure (name `get_sale_per_category`) that takes an input of an `employee_id` and returns sales by that employee for each category.
    - B2c: Write a stored procedure (name `set_product_category`) that takes an input of a `product_id` and a `category_id` and properly updates the product category.



Required Things

Put all your migration scripts in a folder. The name of the folder should be the same as your full ID. Create a zip file of that folder. Upload it in your Google Drive. Submit it in the classroom as well.

After evaluation of your lab tasks, you should delete the folder from the SQL server.