HW2: DDL Script Assignment

Due Oct 8, 2021 by 11:59pm **Points** 100 **Submitting** a file upload
Available Sep 29, 2021 at 9am - Oct 9, 2021 at 11:59pm

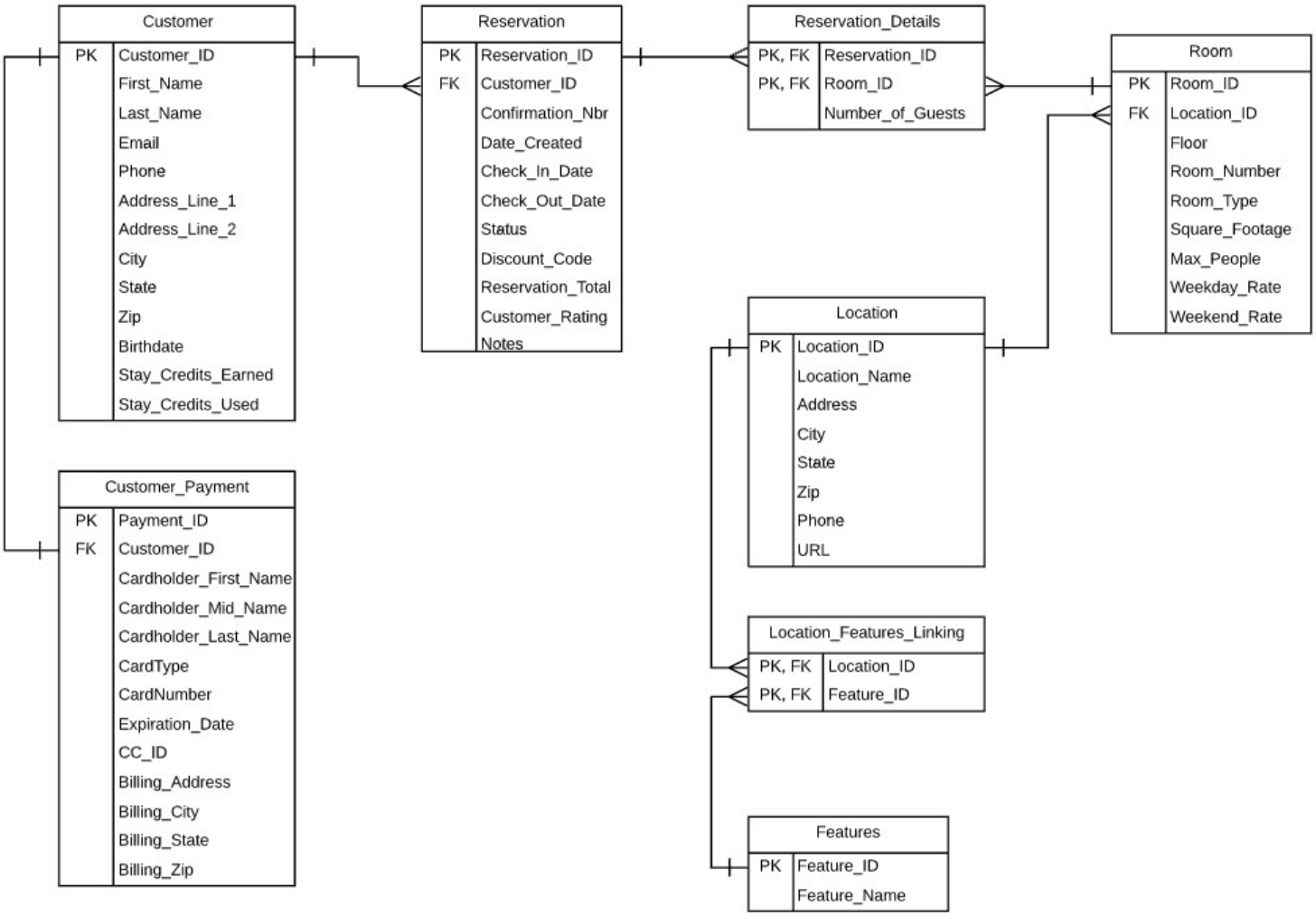
This assignment was locked Oct 9, 2021 at 11:59pm.

Goal: In this assignment, you are tasked with creating a fully functioning DDL script that the TAs could run to create a demo of your Hotel Reservation System database that you designed in the previous Database Design Assignment. Your script must not only build the database according to the requirements below but also seed it with some data for testing purposes. Lastly, your script must be able to run over and over, meaning it can drop database objects (e.g., tables, sequences, indexes), recreate those objects, and seed with test data in a single script.

Assignment Requirements:

Do your own work

- Your team must do your own work and create your own DDL script based on a working (final version) posted below.



Coding Standards

- Naming: Use the final ERD as a reference on how to name tables and columns. For constraints, make the names clear and easy to understand. Avoid abbreviations.
- Data Types / Lengths:
    - NUMBER – Follow the standard of making IDs numeric and any column that involves arithmetic.
    - VARCHAR – All non-numeric fields should be VARCHAR except fields mentioned in CHAR section below.
    - CHAR – Phone numbers will all be defined as CHAR length of 12 to allow for 10 digits and 2 dashes (e.g. 512-999-1234). State is the abbreviation of a state so its CHAR length of 2. Zipcode only needs to store a length of 5. CardType on the payments table will always be 4 character abbreviations of cards. Confirmation number will always be 8 characters of random letters/numbers. Reservation status is used to track a simple U for Upcoming, I for In Progress, C for Completed, N for No-show, R for Refunded so this is always a length of 1. The Room Type will also store a single character for the following room types: D for double beds, Q for single queen, K for single king, S for suite that has two rooms and some form of kitchen, C for cabin.

- DATE – Any dates fields should be DATE format
- Note: Lengths can vary unless specified above.

- Commenting
  - Include comments at least with each section of code (i.e. DROP, CREATE, INSERT, INDEXES)
  - Comments should include a description of section and your name/uteid as the author.
  - If you want to add additional comments to single statements or lines feel free but it's not required. It's just a best practice to comment code well.

Constraints

- Assign primary and foreign keys per the design.
- Only the following can be NULL: Address_line_2 since it's not required. Customers' birthdates since it's not always known. The reservation Checkout Date, Discount Code, Customer Rating, and Notes are also not required since they are allowed to be blank when the reservation is created. Note, if you see other fields that could be nullable, please clarify these assumptions on Slack.
- The following fields should be UNIQUE: Customer email, Feature Name, Location Name, Reservation, and Confirmation Number. Again, if you find other potential non-primary keys, bring this up on Slack.
- DEFAULTS:

o Stay credits earned and used should be set initially to 0 (zero).
o The reservation Date_Created should default to the current date using the SYSDATE function. I know…we didn't cover this in class but you can figure this out if you dig a little in your book or online. You can do it!

- Make sure the following Check constraints are added:
  - Reservation status as of now can only be the following values detailed above: U, I, C, N, or R.
  - The Room Type as of now can only be the following values detailed above: D, Q, K, S, or C.
  - Stay Credits Used should never be greater than the Stay Credits Earned
  - Customer Emailed - emails should have a character length of at least 7 or more. Again, just because we didn't cover it in class doesn't mean you can't google it and figure it out. Name this constraint "email_length_check".

Other

- Create indexes on all foreign keys that are not also part of a primary key. Since primary keys are indexed we won't index a column that is both a PK and FK. Also, index at least 2 other fields in the schema to show you can properly discern which columns should have indexes per design rules discussed in class

- Create sequences that auto-increment the payment_id, reservation_id, room_id, location_id, and feature_id by 1 starting at 1. Create a sequence for customer_id that increments by 1 starting at 100001.

Format

- Easy to Read Code: All your code should be well spaced and indented. If it's not easy to read and messy, you could lose points.
- Sections: You must create your script with the three following sections:
    - Drop Sequence/Tables section - Area of the script that drops all tables and sequences in proper order
    - Create Sequence/Tables section - Area of the script that creates tables/sequences and adds constraints either via CREATE or ALTER TABLE statements
    - Insert Data section - Area of the script that inserts data into the tables using "INSERT INTO"
    - Create Index section – After you seed data, add in indexes for the database to optimize performance
- Commenting: You should add comments before each section that includes a description of what is happening in that section and your name and UTEID, which is a best practice to know who coded what.
- The script must build the database shown in the ERD exactly which means table names, column names, and constraints must match. That being said, we are not going to specify the exact names of constraints that you create unless stated above. Just be sure to use logical, clear names for constraints.

Data Requirements

Seed your tables based on the following requirement: NOTE: Include commits after each group of inserts for a particular table and don't forget to regularly commit to avoid taxing the server and causing NOWAIT error.

- Create the 3 locations mentioned in HW1 and make-up details on address, phone, and URL.
- Create up to 3 features that can be shared or unique to the locations but make sure at least one location has multiple features assigned to it
- Create 2 rooms for each location (even though in reality there should be more)
- Create 2 customers that have payments attached. The first customer should have your first and last name. Their email should be your uteid + "utexas.edu". (e.g. abd123@utexas.edu). The rest of the data about you can be fake. Make up data for the 2nd customer.
- For your customer account, create a single room reservation. For the 2nd, create two separate reservations that are on different dates.
- NOTE: If you need to clarify any data requirements, ask on Slack.

Testing before you turn in your work

- Make sure your script runs without errors before you submit it. Test it by dropping all the tables and running the script. Then run the script to ensure no unexpected errors occur.

Deliverable (What to turn in: two files)

- Turn in your DDL script in a (.sql) file format. If you have issues doing this, at least save it as a .txt file. Do not submit in any other format other than .sql and .txt.

- Write an executive summary (one page) to explain the code, any assumptions your team made to deliver the SQL code to the customer. Save it as a .pdf or .docx file and submit along with the code.