Assignment 3 - DDL Script & Basic DML Sep 2022

**PART 1 - DDL**

Goal: In this assignment, you are tasked with creating a fully functioning DDL script that the TAs or Clint could run to create a demo of your **Ride Share 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**

* This is an individual assignment and you must do your own work and create your own DDL script based on a working/final version posted on Canvas HW2 instructions.

## **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 could involve math.
  + *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 (e.g. AMEX, VISA, MSTR). All flags (e.g. Discounts.Used\_Flag, Vehicle\_Driver\_Linking. Active\_Vehicle\_Flag, Customer\_Payment.Primary\_Card\_Flag) are used to track either a simple Y for *Yes* and Nfor *No*, therefore it will have a length of 1. Bank Routing numbers are always 9 digits. VIN Numbers are always 17 characters in length. CardNumber on the payments table should allow for 16 digits, and CC\_ID should allow for 3 digits.
  + *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: *Start and End Datetimes* for rides since these will not be defined until the rides start and end. The fares and cost fields for the Ride since those are updated after the ride ends. *Rating* is allowed to be null as riders may chose to not rate their trip. Cardholder *middle name* can be null. Note, if you see other fields that could be nullable, please clarify these assumptions on Slack.
* The following fields should be UNIQUE: *Driver/Rider email, Plate number, Vin number, Driver’s License Number. Driver\_ID* on the bank account table should be unique to force it to be a 1-to-1. If you see a need for more non-primary key constraints, please raise that on Slack.
* DEFAULTS:
  + *Request Datime* 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!
  + *Used flag* in the discounts table should be set to N.
  + *Expiration\_Date* on Discounts will always be 30 days from the current system date to encourage ridership.
* Make sure the following Check constraints are added:
  + All “flag” columns as of now can only be the following values: Y or N.
  + *Final\_Fare* should always be *intial\_fare – discount\_amount*
  + Customer Emails - 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 *bank\_account\_id, vehicle\_id, ride\_id, payment\_id, and discount\_id* by 1 starting at 1. Create sequences for *driver\_id* that increments by 1 starting at 100001 and for rider\_id that increments by 1 starting at 3000001.

## **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](https://stackoverflow.com/questions/4842765/ora-00054-resource-busy-and-acquire-with-nowait-specified-or-timeout-expired).

* Create 3 riders that **each** has **one** payment 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 and 3rd customer as well.
* Create 3 drivers that **each** has **one** bank account attached. Make up details about these drivers and their accounts.
* Create a vehicle for **each** driver. Make sure you are inserting into all necessary tables to account for the link between drivers and vehicles.
* Create 3 different rides using the drivers and the riders that you have already created. Make sure these rides occur on different dates, with one of them starting today (using the SYSDATE function)
* Create a discount for one of the riders.
* 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.

**What to turn in**

* Turn in your DDL script in a .sql or .txt file format.

**PART 2 - DML**

# Instructions:

* **Do your own work:** This is an individual assignment and you must do your own work and create your own SQL statements. If you are caught cheating on this or using someone else’s work, you will receive a zero on this assignment and be reported to the Dean of Students. Also, this homework prepares you for the exam coming up so doing the work now will help you learn and do well when it counts more.
* **What to turn in**
  + Clearly separate your code for each question. Save your code into one SQL file with the naming format: **LastName\_FirstName\_UTEid\_HW3**.
  + **Save your file either as a .sql or .txt file. If you need help doing this, ask on Slack.** Files saved in a different format will be 50% and files in a different format that cannot be read into SQL (example: PDF) will result in a 0%.
  + Submit your .txt file on Canvas before the deadline. Late submissions receive 50% off. No submissions will be accepted 24 hours after the deadline.
  + Do not include the DDL in your submission. If you do, you will lose 5 points. Only provide SQL with comments and nothing else. Do this going forward on all other assignments unless noted.
* The SQL problems below will be based on the DDL script that is posted on the Canvas instructions. Download that script and run it before you start.

# Problems:

1. Write a SELECT statement that returns the following columns from the *Customer\_Payment* table: Cardholder first name, cardholder last name, the card type, and the card expiration date. Then, run this statement to make sure it works correctly.

Add an ORDER BY clause to this statement that sorts the result set by expiration date in ascending order (i.e. oldest to newest). Then, run this statement again to make sure it works correctly. Note, this is a good way to iteratively build and test a statement, one clause at a time.

1. Write a SELECT statement that returns one column from the *Customer* table named customer\_full\_name that combines the first\_name and last\_name columns. Format this column with the first name, a space, and last name like this: **Michael Jordan**

Sort the result set by last name in descending sequence.

Write this in two ways:

**Part 1:** Write a query that gets the results by making use of the LIKE operator to pull the customers whose first name begins with letters of L, M, or O.

**Part 2:** Write another query that gets the same results but by using the SUBSTR function to pull the first character of the first name combined with an “IN” operator.

1. Write a SELECT statement that returns these columns from *Reservation*: customer\_id, confirmation\_nbr, date\_created, check\_in\_date, and number\_of\_guests. Return only the rows for reservations that have a status of “upcoming” that have check\_in\_dates that are today or in the future but only for this year. That means to filter where only the check\_in\_date is greater or equal the current date (note: use SYSDATE here) and on or before Dec 31st 2021. See if you can do this with only the following operators (<, >, <=, or >=). Sort results by check\_in\_date to show the most recent first and the older dates at the bottom.
2. **This is a two-part question:**

**Part A:** Create a duplicate of the previous query but this time update the WHERE clause to use the BETWEEN operator. Keep the rest of the query the same.

**Part B:** Using the MINUS operator, compare the query from #3 to the query from Part A in #4. If you get no rows returned, that means the queries produce the same results. Pretty cool huh!?! 😉

1. Write a SELECT statement that returns these column names and data from the *Reservation*table:

customer\_id The customer\_id column

location\_id The location\_id column

length\_of\_stay This is calculated by subtracting check\_in\_date from the check\_out\_date. Assign an alias of *length\_of\_stay*

Filter the query to only show completed reservations (i.e. status = ‘C’). After you have that running correctly, update filter to use the ROWNUM pseudo column so the result set contains only the first 10 rows from the table.

Sort the result set by the column alias **length\_of\_stay** in descending order and then also by customer\_id ascending

1. Write a SELECT statement that returns the first\_name, last\_name, email from *Customer* and also a fourth column called *credits\_available*. The credits available is calculated by subtracting credits used from credits earned. Once you have this, filter to only show customers with at least 10 or more credits available. Sort results by the column alias *credits\_available* in descending order
2. Write a SELECT statement that returns the first, middle, and last name of a customer’s payment profile on *Customer\_Payment*

Using the NULL operator, return only rows for those customers with a middle name. Sort by column positions 2 and then 3 in ascending order

1. Using the DUAL table write a SELECT statement that uses SYSDATE function to create a row with these columns:

today\_unformatted The SYSDATE function unformatted

today\_formatted The SYSDATE function in this format: MM/DD/YYYY

This displays a number for the month, a number for the day, and a four-digit year. Use a FROM clause that specifies the Dual table. *Hint: You will need to implement the TO\_CHAR function to format the sysdate in the format designated above.*

After you write this add the following columns to the row:

Credits\_Earned 25

Stays\_Earned 25 / 10

Redeemable\_stays (25/10) returned with FLOOR() function

Next\_Stay\_to\_earn (25/10) rounded to nearest whole number

Your result table contains only one row.

1. Write a SELECT statement that pulls *Reservation* records for all reservations that are completed (i.e. status of C) for location 2. Return only the following columns: Customer\_id, Location\_id, and a calculated column called length\_of\_stay which is just checkout date minus check-in date. Sort the results by length\_of\_stay descending and customer\_id ascending. Lastly only pull in the top 20 rows. That means we want you to sort the table before filtering the 20 rows. *Hint:* Do this using FETCH command and not with a subquery since subqueries come later.
2. Write a query that returns all the rows from *Customer* for customers that have a gmail.com address (i.e. an email that contains the word ‘Gmail’. The problem is that some people’s emails are gmail.com while others are Gmail.com or GMAIL.COM etc. We want a query that can pull all gmails regardless of capitalization. So in your where clause use the LOWER() function to compare the lowercase version of *email* to filter only rows that contain the word ‘gmail’