Course Project:

Public Bus Transport System (PBTS)

Project Phase 2 (18 Marks)

Due: *22:00hrs, Wednesday March 23, 2022 to eLearn ONLY*

Hours Expected: *6-8 hours per person*

Deliverables

1. **DDL Scripts**
   - **SQL creation script** that creates the tables in a MySQL database for the Logical Design that we have provided in the Appendix.
   - **SQL Load script** that loads sample data into the tables created by your creation script. The sample data is provided in the zip file uploaded on eLearn. Please refer to *MySQL_LoadData_handout.pdf* for the details of loading data from .txt file into MySQL tables. You must include the same sample data in the deliverables as mentioned in the Submission section

     Note: You **ARE NOT** allowed to change the value of any sample data by inserting new records or deleting existing records from the sample data.

2. **DML / Stored Routine Scripts**

   a. The workload for teams with different number of members is different.
      - Teams with only **4 members**, please complete **ALL 8 (i to viii) questions**.
      - Teams with **3 members**, please complete **6 (i to vi) questions**.
      - For those teams with only **2 members**, please complete **4 (i to iv) questions.**

   b. Derive the relevant **DML SQL statements** to support the questions i to viii. A **single SQL statement** is expected for each question except for question iv, for which your team will be writing a stored procedure for your answer. Please include the call to the stored procedure also in the submission.

   c. You can submit SQL statements with the input value hardcoded for DML statements but make sure that it is easy for us to change the input value(s) for testing.

Questions

i. For a given bus plate number and a period defined by a year *Y*, a start month *m1* and an end month *m2* with *m1 <= m2 (e.g., Y=2019, m1=8,* and *m2=10 refers to the duration from 01 August 2019 to 31 October 2019*), list the total number of unique services it has been used for, the total number of trips it has been assigned to, along with the total number of unique drivers who have been assigned to the bus for each month in the specified period. If a bus is not plying in a month, the record for that month will not be listed (note that 10 is missing in the sample shown).

Bus Plate	Month	Total number of unique services	Total number of trips	Total number of drivers
SHA8254Z	8	…	..	..
SHA8254Z	9	…	..	..

ii. Given a string, list all the stops that contain the string in the address or description of a stop (i.e., LocationDesc and/or Address of the returned bus stops must have the string in it). For each returned stop, list the Stop ID, location and address of the stop, service ID served by the stop, and type of the service (I.e., Normal or Express). If the service is normal, list its frequency on weekdays and weekends (Express services will have null in those two columns).

Stop ID	Location	Address	SID	Type	Weekday Freq	Weekend Freq
81131	..	…	77	Normal	15	20
81131	..	…	502	Express	NULL	NULL
…	…	…	…	…	…	…

iii. For all cards that are replacement cards of others, list the card ID, expiry date, number of rides that the card has been used for, the old card it replaced, and the number of rides the old one has been used for. List the cards in the descending order of the expiry date. You cannot assume that all cards have been used (e.g., number of rides corresponding to some cards could be zero).

Replaced Card	Expiry	Number of rides	Old card	Number of Rides of Old card
..				
12	..	1	1	2
..				

iv. Write a **stored procedure** named "findXthPopularStop" that takes in three input parameters (1) Start Date, (2) End Date, and (3) an integer X, and returns the bus stop ID that is ranked Xth position based on the total number of passengers boarding and alighting during the given start date and end date duration (inclusive of both dates). A passenger who boards/alights at one bus stop multiple times is counted multiple times. If there is more than one bus stop with the same total number of passengers boarding and alighting, ranked at the Xth position, all the bus stop IDs are to be returned. For example, given four bus stops A, B, C, and D with total number of passenger boarding/alighting within the given period being 100, 100, 80 and 80 respectively. If X=2, then both bus stops C and D shall be returned. For rides without alighting stops, we assume the alighting happens at the last stop of the service.
Note: you can make use of additional Stored functions or Stored Procedures and call them from inside the Stored procedure findXthPopularStop, if required, to answer the question.

v. For a period defined by a start date and an end date (inclusive of both dates) and a specific CityLink Card ID, list all the rides that it has been used for. For each ride, list the Ride date, SID, Boarding Stop ID and Location description, Alighting Stop ID and Location description, and fare paid **after applying the discount** applicable to the card. For those rides without alighting stop, the fare paid is calculated based on the last stop of the service.

Ride date	SID	Board Stop	Board location	Alight Stop	Alight Location	Fare Paid
…	…					

vi. Given a service ID and a period defined by a start date and an end date (inclusive of both dates), list the total number of passengers ferried on the service, the total number of bus stops passed through by the given service, and the total number of unique stops of the bus service that has at least one passenger boarding/alighting during the interested period. The same passenger travelling 2 (or more) times on the service is counted as 2 (or more). Passenger is identified by his Card ID.

If say for a given period of 2 days, on day 1, the service picks 2 passengers at Stop A, no boarding or alighting passenger at Stop B and Stop C, and drops the 2 passengers at Stop D and Stop E respectively. On day 2, the service picks 1 passenger at Stop A, picks 1 passenger at Stop B, no boarding or alighting passenger at Stop C and Stop D, and drops 2 passengers at Stop E. The total number of unique stops that has passengers boarding/alighting for the service of the given 2-day period is 4 (i.e., Stops A, B, D, and E).

On a specific day or period, a service might not ferry any passengers (e.g., no passengers are using this service during the given period).

SID	**Number of Passengers ferried**	**Total number of stops**	**Total number of unique stops (board or alight)**
…	…		

vii. Given an integer X, find all officers that have been employed for at least X number of years. List their Officer ID, name, years employed, total number of offences discovered, total number of offences that were paid using CitiLink card, total amount of penalty collected and total number of unique bus trips the discovered offences were for.

Officer ID	**Name**	**Years employed**	**Number of offences**	**Number of offences paid using Card**	**Penalty amount collected**	**Number of unique bus trips**
…	…					

viii. Given an integer X, list the top X drivers based on the total number of bus trips he/she has driven throughout his entire career. List the rank, driver's DID, name, total number of bus trips he has driven, the total number of unique bus services the trips are made for and the total number of unique buses (differentiated by different plate numbers) used.
All drivers who drove the same number of bus trips are considered to be at the same rank position and will have to be returned.

For example (note this is not the actual data), Driver 1111 drove 100 trips, Driver 2222 drove 80 trips, Driver 3333 drove 58 trips, Driver 4444 drove 58 trips, Driver 555 drove 50 trips, Driver 6666 drove 50 trips and Driver 7777 drove 45 trips. Drivers 3333 and 4444 drove the same number of trips and are ranked 3rd. Drivers 5555 and 6666 drove the same number of trips and are ranked 4th.

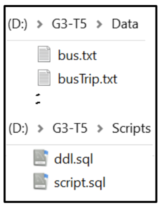
For X = 4, the output should be as shown below:

Rank	Driver ID	Name	Number of trips	Number of unique services	Number of unique buses
1	1111	….	100	….	…..
2	2222	….	80	…..	…..
3	3333	….	58	….	…..
3	4444	….	58	…..	…..
4	5555	….	50	….	…..
4	6666	….	50	…..	…..

Submission

1. Submission must be made via eLearn only

2. The deliverable should be a folder named GX-TY that is compressed as GX-TY.zip and contains the sub folders as shown below, where X is the section (1 – 3) and Y is your team number (1-15). Example if you belong to G3 and Team 5, your zip file should be named G3-T5.zip. You must have two folders, **“Data”** and **“Scripts”** inside the folder named after your team.



```
(D:) > G3-T5 > Data
    bus.txt
    busTrip.txt
    :
(D:) > G3-T5 > Scripts
    ddl.sql
    script.sql
```

3. The **“Data”** folder must contain the data file provided to you in .txt or .csv files. (This is to help us test your scripts easily, if required).

4. The **“Scripts”** folder must contain the following 2 files:
   - The first file named **“ddl.sql”**

     This SQL file must contain **both** the CREATE and LOAD data statements such that it allows us to create tables and load your data with minimum effort using "execute all" command on MySQLWorkBench.

     The first and second statements in this file should be "**create schema GXTY**;" and "**use GXTY**;" where X is your section (1 – 3) and Y is your team number (1-15).

     It will be followed by CREATE table and LOAD data statements in the proper table order. The path of your LOAD statements should be *C:/<folder>/***GXTY/data/***.txt**;** where X is your section (1 – 3) and Y is your team number (1-15)

     Note: The drive and folder can be any folder (or folder path) in your machine but ensure that the data files are placed under a folder named data and within a folder named after your team number. Tester will use ‘find and replace’ function to change the path C:/<folder>/GXTY/data to a path in his laptop for testing.

- Second file named **“script.sql”**

  This file contains SQL statements to all the questions that your team is required to complete. Every solution statement should have a comment tag that states the question number.

```
Example:
#Question 1:
Select * from Table1;
```

  If the question requires input value(s), use set statements for changing the input values with ease.

```
Example:
#Question 2:
set @first_input = '2020-12-12';
set @sec_input = 'value';
Select * from Table2 where col1 = @first_input and col2 = @sec_input;
```

Grading Scheme

Project Phase 2
DDL Scripts: SQL Statements for database creation and data loading Marks will be deducted for - Missing data files - Missing create schema and use schema statements.
DML / Stored Routine Scripts: SQL Statements for the required questions SQL Statements should be complete so that they can be used with different input values. Grades for individual questions might vary slightly based on the number of members in the team and the complexity of the query - Marks will NOT be awarded for misinterpreting the DML questions. - Clarification should be made online via the eLearn **Discussion Forum**.
You will be penalized if your deliverables are not according to our instructions as stated in Submission section.

Appendix

MySQL supports functions and clauses that may come in handy.

IF control function

The **IF** function returns a value based on a condition.

IF(expr, if_true_expression, if_false_expression)
returns "if_true_expersssion" if the "expr" evaluates to **TRUE**, otherwise returns "if_false_expression". The function may be used to return a numeric or string.

For an example, given the following statement
IF(column1 IS NULL, 'N/A', column1)
the output will be N/A if column1 has NULL value; otherwise it will be column1 value

IFNULL control function

The **IFNULL** function accepts two arguments and returns the first argument if the argument is not null; Otherwise, it returns the second argument.

IFNULL(expr1, expr2)
returns expr1 if expr1 is not NULL, otherwise it returns expr2.

For an example;
IFNULL(column1, column2)
The output will be column1 if it is not NULL, otherwise it will be the column2 value.

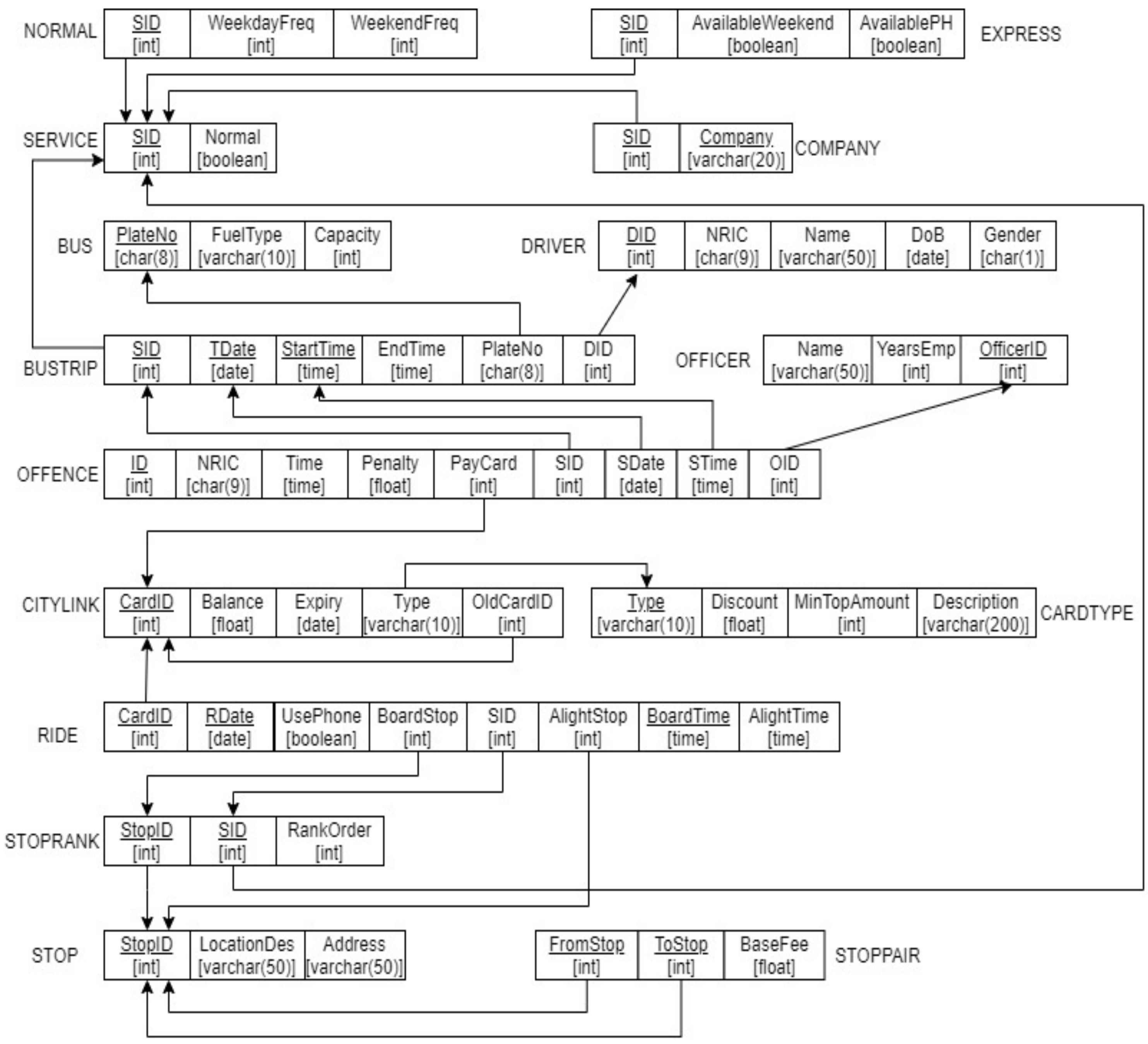
LIMIT

LIMIT can be used to constrain the number of rows returned by the **SELECT** statement and should be placed at the end of the statement after **HAVING** clause.

For an example;
*Select * from DRIVER LIMIT 5;*
This statement retrieves the first 5 rows from DRIVER table.

STORED function
It is a special kind of stored program that returns a single value. Stored function and Stored procedure are called stored routines. Similar to a Stored procedure, Stored function is a set of SQL statements that are stored in the server.
However, Stored function is different from a Stored procedure in the way that Stored function can be used in the WHERE / HAVING / SELECT clause of the SQL statement.
Refer to http://www.mysqltutorial.org/mysql-stored-function/ for more information

Project Logical Design


NORMAL
SID [int]
WeekdayFreq [int]
WeekendFreq [int]
SID [int]
AvailableWeekend [boolean]
AvailablePH [boolean]
EXPRESS
SERVICE
SID [int]
Normal [boolean]
SID [int]
Company [varchar(20)]
COMPANY
BUS
PlateNo [char(8)]
FuelType [varchar(10)]
Capacity [int]
DRIVER
DID [int]
NRIC [char(9)]
Name [varchar(50)]
DoB [date]
Gender [char(1)]
BUSTRIP
SID [int]
TDate [date]
StartTime [time]
EndTime [time]
PlateNo [char(8)]
DID [int]
OFFICER
Name [varchar(50)]
YearsEmp [int]
OfficerID [int]
OFFENCE
ID [int]
NRIC [char(9)]
Time [time]
Penalty [float]
PayCard [int]
SID [int]
SDate [date]
STime [time]
OID [int]
CITYLINK
CardID [int]
Balance [float]
Expiry [date]
Type [varchar(10)]
OldCardID [int]
Type [varchar(10)]
Discount [float]
MinTopAmount [int]
Description [varchar(200)]
CARDTYPE
RIDE
CardID [int]
RDate [date]
UsePhone [boolean]
BoardStop [int]
SID [int]
AlightStop [int]
BoardTime [time]
AlightTime [time]
STOPRANK
StopID [int]
SID [int]
RankOrder [int]
STOP
StopID [int]
LocationDes [varchar(50)]
Address [varchar(50)]
FromStop [int]
ToStop [int]
BaseFee [float]
STOPPAIR