**Instructions**

Edit Instructions

**Q1. Restaurant Address Cleanup (20 points)**

Use the Restaurants.sql file, which creates three tables Restaurant, Reviewer, and Rating. In this problem, we are concerned with the Restaurant table, which has a single attribute 'Address' of type 'varchar2(100)'. We would like address to be searchable.  So we would like to create another table Restaurant\_Locations with the following attributes:

rID, name, street\_address,  city, state, zipcode, cuisine

a. Create Restaurant\_Locations table. Use the source dataset to determine the data types (and sizes) to use for each of the attributes.

b. Write a cursor (using SQL and PL/SQL) to process each row from the original Restaurant table, extracting information as necessary to populate the new Restaurant\_Locations table. The original address field must be split up and parsed into the new street\_address, city, state, and zipcode fields.

**Submit 1.sql file.**

**Q2. Restaurant JDBC (20 points)**

Create ZipCode table based on attributes in zipcode.csv file (zip, city, state, latitude, longitude, timezone, dst)

Then download the CHIzipcode.csv file provided on D2L.

1. Write a Java program (similar to the one showed in class to connect to your Oracle database using JDBC driver) to convert the zipcode.csv file into a series of SQL insert statements for ZipCode table.
2. Use the program to join zipcodes table with restaurant\_locations table and obtain latitude and longitude of all restaurants in the Restaurants table.
3. Print out the name, zipcode, latitude, longitude using your program.

Output should look like this: Shanghai Inn, 60625, "41.971614","-87.70256"

**Submit your Q2.java file**

**Q3. Triggers. (20 points)**

Consider the TASK and CONTRACT tables defined by the following script, which also populates the JOB table:

DROP TABLE CONTRACT CASCADE CONSTRAINTS;

DROP TABLE TASK CASCADE CONSTRAINTS;

CREATE TABLE TASK (

TaskID CHAR(3),

TaskName VARCHAR(20),

ContractCount NUMERIC(1,0) DEFAULT 0,

CONSTRAINT PK\_TASK PRIMARY KEY (TaskID)

);

CREATE TABLE CONTRACT

(

TaskID CHAR(3),

WorkerID CHAR(7),

Payment NUMERIC(6,2),

CONSTRAINT PK\_CONTRACT PRIMARY KEY (TaskID, WorkerID),

CONSTRAINT FK\_CONTRACTTASK FOREIGN KEY (TaskID) REFERENCES TASK (TaskID)

);

INSERT INTO TASK (TaskID, TaskName) VALUES ('333', 'Security' );

INSERT INTO TASK (TaskID, TaskName) VALUES ('322', 'Infrastructure');

INSERT INTO TASK (TaskID, TaskName) VALUES ('896', 'Compliance' );

SELECT \* FROM TASK;

COMMIT;

The ContractCount attribute of TASK should store a count of how many workers have signed contracts to work on that task, that is, the number of records in CONTRACT with that TaskID, and its value should never exceed 3. Your task is to write three triggers that will maintain the

value of the ContractCount attribute in TASK as changes are made to the CONTRACT table. Write a script file 4.sql containing definitions of the following three triggers:

1. The first trigger, named NewContract, will fire when a user attempts to INSERT a row into CONTRACT. This trigger will check the value of ContractCount for the corresponding task. If ContractCount is less than 3, then there is still room in the task for another worker, so it will allow the INSERT to occur and will increase the value of ContractCount by one. If ContractCount is equal to 3, then the task is full, so it will cancel the INSERT and display an error message stating that the task is full.

2. The second trigger, named EndContract, will fire when a user attempts to DELETE one or more rows from CONTRACT. This trigger will update the values of ContractCount for any affected tasks to make sure they are accurate after the rows are deleted, by decreasing the value of ContractCount by one each time a worker is removed from a task.

3. The third trigger, named NoChanges, will fire when a user attempts to UPDATE one or more rows of CONTRACT. The trigger will cancel the UPDATE and display an error message stating that no updates are permitted to existing rows of CONTRACT.

NewContract and EndContract should be row-level triggers; NoChanges should be a statement-level trigger.

Run the script to define your triggers and test them to make sure they work by doing a few INSERTS, DELETES, and UPDATES on the CONTRACT table.

**Submit one 3.sql file with the trigger code and your insert, delete and update statements.**