CS2102 Project (20 marks)

The objective of this team project is for you to apply what you have learned in class to design and develop a web-based database application. The project is to be done in teams of four students.

1 Overview of Application

This section provides a brief specification of the database application. As the specification is not meant to be complete, each team has the freedom to decide on the application's data requirement as well as the application's functionalities/features to be implemented. Your database design and implementation for the application should demonstrate non-trivial usage of SQL and database features.

You are to develop a database application for a food delivery service (FDS). The FDS application allows customers to order food from restaurants and have them delivered to their locations by delivery riders. The application supports four types of users: customers, restaurant staff, delivery riders, and FDS managers.

Each restaurant has a menu listing their food items, availability and prices. Food items are classified into categories (e.g., local, western) to facilitate browsing. Each restaurant could specify a daily limit on the maximum number of orders that it will accept for each food item. Once the number of orders for a food item has reached its daily limit, the FDS application will automatically update the availability status of that item on the menu to be unavailable. The menu information could also be updated dynamically by restaurant staff.

The total cost of each order is the sum of the cost of the ordered food items and a delivery fee. Each restaurant imposes a minimum monetary amount to be spent for each order's cost of the food items; a new order will not be accepted by the restaurant if the order's total cost of the ordered items is below the restaurant's stipulated minimum amount threshold. The delivery fee for each order is determined by the FDS based on some criteria. For each completed order, the customer receives a reward point for each dollar spent and the accumulated points could be used to offset the delivery fees for his/her future orders. Customers could opt to make their payment using a pre-registered credit card or paying cash on delivery. For each completed order, the customer could post reviews on the food and give ratings on the delivery service. The posted reviews could be viewed by all customers.

Each restaurant may offer promotional prices for their menu items for certain periods of time (e.g., a discount of 20% for all orders exceeding $100 during December). The FDS may also launch various promotions throughout the year (e.g., 10% discount for a customer's first order, free delivery during certain time periods) or offer special discount coupons to targeted customers (e.g., customers who have not placed any order for the last three months). For the purpose of tracking, each promotional campaign

is assigned a unique identifier.

Each order's food items must be from a single restaurant, and each order is assigned to a delivery rider who will collect and deliver the ordered items to the customer. The rider assignment is determined by the FDS. For each order, the FDS application records the following timings: the time when the order was placed by the customer, the time when the rider departs for the restaurant to collect the food, the time when the rider arrives at the restaurant, the time when the rider departs from the restaurant for the delivery location, and the time when the rider delivers the order. For each customer, the FDS application keeps track of the five most recent delivery locations for that customer. For the delivery location of each new order, a customer could either provide a delivery location or choose from one of his/her five recent delivery locations.

The FDS operates daily from 10am to 10pm. Each rider works for the FDS as either a full-time or a part-time employee. Each full-time rider maintains a monthly work schedule (MWS) while each part-time rider maintains a weekly work schedule (WWS). In the FDS application, a week is defined to be a duration consisting of seven consecutive days, and a month is defined to be a duration consisting of four consecutive weeks.

Each WWS specifies the hour intervals that the rider is available for work each day. For example, a rider's WWS might specify the following work hours for six work days: for Monday to Thursday, the rider works for the hours 10am to 1pm, 4pm to 6pm, and 7pm to 10pm; and for Saturday and Sunday, the rider works for the hours 10am to 1pm and 5pm to 8pm. In this example, the rider works for a total of $(4 \times 8) + (2 \times 6) = 44$ hours for that week. The WWS for part-time riders must satisfy the following three properties:

- Each hour interval must start and end on the hour, and its duration must not exceed four hours. For example, the hour intervals "12am to 2:30pm" and "2pm to 7pm" are both not allowed.
- There must be at least one hour of break between two consecutive hour intervals.
- The total number of hours in each WWS must be at least 10 and at most 48.

The MWS for full-time riders must satisfy the following three properties:

- The four WWSs in a MWS must be equivalent.
- Each WWS in a MWS must consist of five consecutive work days; i.e., each WWS must belong to one of the following seven options: Monday to Friday, Tuesday to Saturday, Wednesday to Sunday, Thursday to Monday, Friday to Tuesday, Saturday to Wednesday, or Sunday to Thursday.
- Each work day must consist of eight work hours comprising of two four-hour periods with an hour break in between. Specifically, the work hours must belong to one of the following four shifts:
  - Shift 1: 10am to 2pm and 3pm to 7pm.
  - Shift 2: 11am to 3pm and 4pm to 8pm.
  - Shift 3: 12pm to 4pm and 5pm to 9pm.
  - Shift 4: 1pm to 5pm and 6pm to 10pm.

There must be at least five riders (part-time or full-time) working at each hourly interval.

Each part-time rider earns a weekly base salary and each full-time rider earns a monthly base salary. On top of the base salary, each rider also earns a fee for each completed delivery; both base salaries and delivery fees are computed based on some criteria.

2 Application Requirements & Functionalities

Your FDS application should not be limited by the functionalities described in the previous section. You are free to introduce additional functionalities and realistic data constraints to make your application interesting and non-trivial (e.g., requiring complex SQL queries, transactions, triggers, etc.). You are also free to use any of the database's features and other SQL constructs beyond what are covered in class. Your application must contain at least three appropriate applications of triggers.

Your FDS application must provide at least the following functionalities.

- Support the creation/deletion/update of data for the different users (customers, restaurant staff, delivery riders, and FDS managers).
- Support data access for the different users (e.g., customers could view review postings and their past orders, riders could view their past work schedules and salaries).
- Support the browsing/searching of food items by customers.
- Support the browsing of summary information for FDS managers. The summary information could include the following:
    1. For each month, the total number of new customers, the total number of orders, and the total cost of all orders.
    2. For each for each month and for each customer who has placed some order for that month, the total number of orders placed by the customer for that month and the total cost of all these orders.
    3. For each hour and for each delivery location area, the total number of orders placed at that hour for that location area.
    4. For each rider and for each month, the total number of orders delivered by the rider for that month, the total number of hours worked by the rider for that month, the total salary earned by the rider for that month, the average delivery time by the rider for that month, the number of ratings received by the rider for all the orders delivered for that month, and the average rating received by the rider for all the orders delivered for that month.
- Support the browsing of summary information for restaurant staff. The summary information could include the following:
    1. For each month, the total number of completed orders, the total cost of all completed orders (excluding delivery fees), and the top 5 favorite food items (in terms of the number of orders for that item).

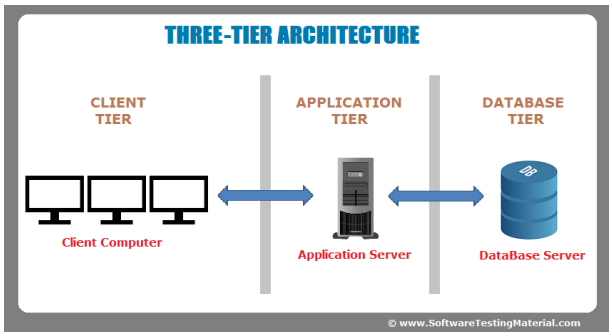

Figure 1: Three-Tier Web-based Application Architecture

2. For each promotional campaign, the duration (in terms of the number of days/hours) of the campaign, and the average number of orders received during the promotion (i.e., the ratio of the total number of orders received during the campaign duration to the number of days/hours in the campaign duration).

- Support the browsing of summary information for delivery riders. The summary information could include weekly/monthly information on the total number of orders delivered by the rider, the total number of hours worked by the rider, and the total salary earned by the rider.

For the final project demo, your application's database should be loaded with reasonably large tables. To generate data for your application, you can use some online data generators (e.g., https://mockaroo.com, https://www.generatedata.com) or write your own program.

3 Software Tools

The architecture of your application follows the typical three-tier (clients, application server, database server) architecture of a Web-based database application shown in Figure 1 with Web browsers as the clients and a Web server as the application server.

You will use PostgreSQL, an open source relational database system, for this project. For an overview of how to install and use PostgreSQL, refer to the file `postgresql.pdf` in LumiNUS's files folder named `SQL`.

You may use any frontend/backend development stack for your web application such as Express, Bootstrap, Angular JS, React JS, and PHP frameworks (e.g, Laravel, Symfony).

Your project must not use any Object-Relational Mapping (ORM) framework (e.g., Hibernate, Sequelize, SQLAlchemy).

4 Project Deadlines & Deliverables

The are two deadlines for this project.

Date	Project Task
March 4, 10pm	Submission of preliminary ER model design
Week 8 (March 9-13)	Review of ER model design (during tutorials & evenings)
April 9, 10pm	Submission of project report & source code to LumiNUS
Week 13 (April 13-17)	Project Demo (during tutorials & evenings)

Each team is to submit a **preliminary ER model design** on March 4, 10pm, which will be reviewed during week 8's tutorial classes and some evenings. The details on the submission process and the registration for the review meetings will be announced later.

The final project deliverables (which are due on April 9, 10pm) consist of a project report and the application's source code. The **project report** (up to a maximum of 20 pages in pdf format with at least 10-point font size) should include the following:

1. Names and student numbers of all team members and project team number (on the first page).
2. A listing of the project responsibilities of each team member.
3. A description of your application's data requirements and functionalities. Highlight any interesting/non-trivial aspects of your application's functionalities/implementation. List down all the application's data constraints.
4. The ER model of your application. If your ER model is too large (spanning more than a page), you may want to include a single-page simplified ER model (with non-key attributes omitted) before presenting the detailed ER model. Provide justifications for any non-trivial design decisions in your ER model. List down all the application's constraints that are not captured by your ER model.
5. The relational schema derived from your ER data model; i.e show the DDL statements of all your database tables. List down all the application's constraints that are not enforced by your relational schema (i.e., the constraints that are enforced using triggers). For each database table, state whether it is in 3NF/BCNF. Provide justifications for tables that are not in 3NF/BCNF.
6. Present the details of three non-trivial/interesting triggers used in your application by providing an English description of the constraint enforced by each trigger and showing the code of the trigger implementation.
7. Show the SQL code of three of the most complex queries implemented in your application. Provide an English description of each of these queries.
8. Specification of the software tools/frameworks used in your project.
9. Two or three representative screenshots of your application in action.
10. A summary of any difficulties encountered and lessons learned from the project.

The **source code submission** should include a README file describing how to deploy and run your application.

The registration for the project demo will be announced later.

5 Evaluation Criteria

The maximum score for the project is 20 marks with the following breakdown:

- Preliminary ER model design: 1 mark
- Project Report & Demonstration: 19 marks
  - Database design (6 marks)
    - ER data model (3 marks)
    - Relational schema (3 marks)
  - Interestingness/Complexity of application (5 marks)
    - Interesting complex queries (2 marks)
    - Apprioriate applications of triggers (3 marks)
  - Application design & functionalities (2 marks)
  - Application user interface (2 marks)
  - Report organization & presentation (2 marks)
  - Project demo (2 marks)

6 Submission of Project Report & Source Code

- Submit your source code by creating a zip file named codeNN.zip, where `NN` is your project team number. Upload this file into LumiNUS file folder named Project-Code-Submission.
- Submit your report by creating a pdf file named reportNN.pdf, where `NN` is your project team number. Upload this file into LumiNUS file folder named Project-Report-Submission.

These deliverables are due on **April 9, 10pm**.

For late submissions, submit to the LumiNUS file folders Late-Project-Code-Submission and Late-Project-Report-Submission. One mark will be deducted for each late day up to two late days; projects submitted after the second late day will receive zero marks and will not be graded.