# **COMP2714 Module 2 Case Study Template**

**Name:** Jiarui Xing

**Student ID:** A01354731

**Name:** YOUR FULL NAME HERE

**Student ID:** YOUR STUDENT ID HERE

**Name:** YOUR FULL NAME HERE

**Student ID:** YOUR STUDENT ID HERE

**Name:** YOUR FULL NAME HERE

**Student ID:** YOUR STUDENT ID HERE

Note: Please edit the name and student ID number to reflect your identity.

## Section A – Analysis and Discussion

|  |  |
| --- | --- |
| Mistake 1 | |
| Mistake | Trip.driverID references Staff.id |
| Correction | Trip.driverID references Driver.id |
| Explanation | Referencing Staff.id allows any staff member to be assigned as a driver for a trip, even if they are not qualified drivers. This could result in trips being assigned to staff members who are not licensed to drive, leading to operational issues and potential safety risks. By correctly referencing Driver.id, we ensure only valid drivers are assigned to trips. |
| Mistake 2 | |
| Mistake | 4WD [vin make model rideHeight wheelType] |
| Correction | 4WD [vin rideHeight wheelType] |
| Explanation | Including make and model in the 4WD table duplicates data already stored in the Vehicles table, leading to data redundancy and potential inconsistencies. If make or model changes in Vehicles but not in 4WD, it can cause discrepancies. Removing these attributes from 4WD ensures data is maintained in a single location. |
| Mistake 3 | |
| Mistake | Using name as the primary key in EmergencyContact table. |
| Correction | Define the primary key of EmergencyContact as (name, userID). |
| Explanation | Using name alone as a primary key assumes all emergency contact names are unique across all users, which is unrealistic. Different users may have emergency contacts with the same name, leading to primary key conflicts. By using a composite primary key (name, userID), we ensure that each contact is uniquely associated with a specific user. |

|  |  |
| --- | --- |
| Mistake 4 | |
| Mistake | The Trip table lacks a proper primary key definition. |
| Correction | Define the primary key of Trip as (userID, driverID, bookingTime). |
| Explanation | bookingTime alone may not uniquely identify a trip if multiple trips are booked at the same time. Without a proper primary key, it is possible to have duplicate trip records or difficulty in uniquely identifying a specific trip, leading to data integrity issues. A composite primary key ensures each trip record is unique. |
| Mistake 5 | |
| Mistake | Overlapping subclasses DRIVER and ADMIN are not properly represented in the mapping. |
| Correction | Implement a mapping strategy that supports overlapping subclasses by creating an Admin table and allowing staff members to be associated with both Driver and Admin roles. |
| Explanation | The current mapping does not accommodate staff members who have multiple roles (overlapping subclasses). Without representing the Admin subclass, administrative staff cannot be properly modeled in the database. This limits the system's ability to accurately represent staff roles and may prevent assigning multiple roles to a single staff member, affecting functionality. |

**SEE NEXT PAGE FOR SECTION B**

## Section B – EER Diagram to Relational Mapping

**YOUR RELATIONAL SCHEMA MUST FIT IN HERE**

** USER**

* **ID *(Primary Key)***
* **DoB**
* **FirstName**
* **MiddleName**
* **LastName**

** EMERGENCY\_CONTACTS**

* **UserID *(Foreign Key to USER.ID)***
* **ContactName**
* **Email**
* **Phone**
* **Primary Key: (UserID, ContactName)**

** STAFF**

* **ID *(Primary Key)***
* **DoB**
* **FirstName**
* **MiddleName**
* **LastName**

** STAFF\_PHONE**

* **StaffID *(Foreign Key to STAFF.ID)***
* **Phone**
* **Primary Key: (StaffID, Phone)**

** DRIVER**

* **ID *(Primary Key & Foreign Key to STAFF.ID)***
* **Licence**

** ADMIN**

* **ID *(Primary Key & Foreign Key to STAFF.ID)***
* **DeskNumber**

** VEHICLES**

* **VIN *(Primary Key)***
* **Make**
* **Model**

** FOUR\_WD**

* **VIN *(Primary Key & Foreign Key to VEHICLES.VIN)***
* **RideHeight**
* **WheelType**

** TWO\_WD**

* **VIN *(Primary Key & Foreign Key to VEHICLES.VIN)***
* **FrontWheelDrive**

** TRIP**

* **BookingTime *(Primary Key)***
* **StartTime**
* **EndTime**
* **VIN *(Foreign Key to VEHICLES.VIN, Nullable)***

** TRIP\_STOP\_LOCATION**

* **BookingTime *(Foreign Key to TRIP.BookingTime)***
* **StopNumber**
* **Address**
* **City**
* **State**
* **Primary Key: (BookingTime, StopNumber)**

** TRIP\_USER**

* **BookingTime *(Foreign Key to TRIP.BookingTime)***
* **UserID *(Foreign Key to USER.ID)***
* **Primary Key: (BookingTime, UserID)**

** TRIP\_DRIVER**

* **BookingTime *(Foreign Key to TRIP.BookingTime)***
* **DriverID *(Foreign Key to DRIVER.ID)***
* **Primary Key: (BookingTime, DriverID)**

** RATES\_DRIVER**

* **UserID *(Foreign Key to USER.ID)***
* **DriverID *(Foreign Key to DRIVER.ID)***
* **RatingValue**
* **RatingDate**
* **Primary Key: (UserID, DriverID)**

** RATES\_VEHICLE**

* **UserID *(Foreign Key to USER.ID)***
* **VIN *(Foreign Key to VEHICLES.VIN)***
* **RatingValue**
* **RatingDate**
* **Primary Key: (UserID, VIN)**

**YOUR FOREIGN KEY CONSTRAINTS MUST FIT IN HERE**

** EMERGENCY\_CONTACTS**

* **UserID references USER(ID)**

** STAFF\_PHONE**

* **StaffID references STAFF(ID)**

** DRIVER**

* **ID references STAFF(ID)**

** ADMIN**

* **ID references STAFF(ID)**

** FOUR\_WD**

* **VIN references VEHICLES(VIN)**

** TWO\_WD**

* **VIN references VEHICLES(VIN)**

** TRIP**

* **VIN references VEHICLES(VIN)**

** TRIP\_STOP\_LOCATION**

* **BookingTime references TRIP(BookingTime)**

** TRIP\_USER**

* **BookingTime references TRIP(BookingTime)**
* **UserID references USER(ID)**

** TRIP\_DRIVER**

* **BookingTime references TRIP(BookingTime)**
* **DriverID references DRIVER(ID)**

** RATES\_DRIVER**

* **UserID references USER(ID)**
* **DriverID references DRIVER(ID)**

** RATES\_VEHICLE**

* **UserID references USER(ID)**
* **VIN references VEHICLES(VIN)**

## Section C – Relational Model and Database Systems

|  |  |
| --- | --- |
| Attribute Type | Answer (Please use format Table.attribute) |
| A superkey | Employee.id, Employee.firstName |
| A minimal key | Employee.id |
| A foreign key | Project.projectLeader (references Employee.id) |

### Give an example of:

### Database Operations

|  |  |  |
| --- | --- | --- |
| Operation 1 | | |
| Operation | | Update the tuple (“Website Setup”, “Get a functional website setup”, 12000, 2019) to (“Website Setup”, “Get a functional website setup”, 20000, 1919) in the relation “Project” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | no |
| IF YES | **Type of constraint violated** |  |
| **Description of violation** |  |
| Operation 2 | | |
| Operation | | Insert the tuple (2014, “Rebecca”, “Zhang”, “Administration”)in the relation “Employee” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | yes |
| IF YES | **Type of constraint violated** | Primary key constraint |
| **Description of violation** | Inserting a tuple with id 2014, which already exists in the Employee table, violates the primary key constraint requiring unique ids. |
| Operation 3 | | |
| Operation | | Update the tuple (2020, “2020 Marketing”, 2/1/2020, 5, true) to (1919, “Overall Marketing”, 2/1/2020, 5, true) in the relation “TimeLog” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | yes |
| IF YES | **Type of constraint violated** | Referential integrity constraint |
| **Description of violation** | Updating projectName to "Overall Marketing", which does not exist in the Project table, violates the referential integrity constraint that TimeLog.projectName must reference an existing Project.name. |

|  |  |  |
| --- | --- | --- |
| Operation 4 | | |
| Operation | | Insert the tuple (, “Test”, “Test”, “Test”) in the relation “Employee” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | yes |
| IF YES | **Type of constraint violated** | Domain constraint |
| **Description of violation** | Inserting non-numeric values into the id attribute violates the domain constraint since id is expected to be numeric. |
| Operation 5 | | |
| Operation | | Delete the tuple (2014, “Daniel”, “Johnson”, “Administration”) in the relation “Employee” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | yes |
| IF YES | **Type of constraint violated** | Referential integrity constraint |
| **Description of violation** | Deleting Employee id 2014 violates the referential integrity constraint because TimeLog entries reference this employee via employeeID. |
| Operation 6 | | |
| Operation | | Insert the tuple (“Talent Recruitment Initiative”, “Get the best and brightest UQ graduates to work for us!”, 10000, 2014) in the relation “Project” |
| Integrity constraint violated?  (Write either “yes” or “no”) | | yes |
| IF YES | **Type of constraint violated** | User-defined constraint (business rule) |
| **Description of violation** | Assigning projectLeader 2014, who is in Administration, violates the business rule prohibiting administration staff from being project leaders. |

### Provide a database operation which:

|  |  |
| --- | --- |
| Operation Requirements | Answer |
| Would result in a domain constraint violation | Insert the tuple (2021, "John", "Doe", "Intern") into the Employee table. |
| Would result in a referential integrity constraint violation | Insert the tuple (9999, "Website Setup", "3/1/2020", 4, true) into the TimeLog table. |