Final Report

WinWin

Ride-hailing platform for local-motorcycle service provider and user

Present to

Assoc. Prof. Dr. Chotirat Ratanamahatana

By G14 Waterfall

6230123921 Thitaree Setwipattanachai

6230252121 Tarm Kalavantavanich

6231301421 Kanokpich Chaiyawan

6231304321 Kittipong Deevee

6231307221 Jirawat Kusalangkurwat

6231333521 Nopdanai Sayamnet

6231353021 Raviporn Akekunanon

6231372021 Atiwat Deepo

Term Project

2110322 Database Systems, Semester 1 of Academic year 2021

Department of Computer Engineering,

Faculty of Engineering, Chulalongkorn University

**Table of Contents**

[Introduction 1](#_Toc88347924)

[Organization background 1](#_Toc88347925)

[Objective of the project 1](#_Toc88347926)

[An overview of the system 2](#_Toc88347927)

[System Functionalities 3](#_Toc88347928)

[ER Diagram / Document-based design schema 5](#_Toc88347929)

[ER Diagram 5](#_Toc88347930)

[Document-based design schema 6](#_Toc88347931)

[Schema diagram 7](#_Toc88347932)

[Normalization of Relation 8](#_Toc88347933)

[Data Dictionary 9](#_Toc88347934)

[Indexing 16](#_Toc88347935)

[Stored procedures 17](#_Toc88347936)

[Get rider from station 17](#_Toc88347937)

[Update Rider Station 18](#_Toc88347938)

[Stored function 19](#_Toc88347939)

[Price Calculation 19](#_Toc88347940)

[Calculate Total Ride Time 20](#_Toc88347941)

[Triggers 21](#_Toc88347942)

[Update Number of riders 21](#_Toc88347943)

[Delete Account Log 23](#_Toc88347944)

[Integrity 24](#_Toc88347945)

[Domain Integrity: Valid value of attribute 24](#_Toc88347946)

[Domain Integrity: Unique attribute 25](#_Toc88347947)

[Referential Integrity: Delete operation 25](#_Toc88347948)

[Execution path 26](#_Toc88347949)

[SQL complex query 28](#_Toc88347950)

[Riders with their manager and special conditions 28](#_Toc88347951)

[Top Spender in November 2021 29](#_Toc88347952)

[Ordering gross subscription sales in 2021 with top spender for each subscription 30](#_Toc88347953)

[Appendix: External Resource 31](#_Toc88347954)

Project name: WinWin

Ride-hailing platform for local-motorcycle service provider and user

# Introduction

Nowadays, motorcycle taxis are a major type of transportation in Bangkok (second only to MRT). However, riders have low income because ride-hailing platforms snatch their market share. Moreover, ride-hailing platforms can make users more satisfied than motorcycle taxis e.g., users can call ride-hailing platforms everywhere.

Therefore, WinWin wants to digitalize the motorcycle taxi system and to utilize route familiarity and locality of local motorcycle taxis to be an advantage that other ride-hailing platforms do not have.

## Organization background

WinWin, an online platform that connects customers with the local motorcycle taxis, has two business partners. Firstly, the Department of Land Transport, Ministry of Transport, which provides WinWin with the information about motorcycle taxis in the Bangkok area. Secondly, Winnonie, a startup founded by Bangchak Corporation group that rents out electric motorcycles.

WinWin believes that employing local motorcycle taxis as service providers is the best option since the riders are familiar with the route and can arrive at the customer's location faster than other riders from other ride-hailing platforms.

WinWin also thinks that building this application would satisfy stakeholders such as the Department of Land Transportation, Ministry of Transport, Winnonie, Motorcycle taxis, and consumers. Because Winnonie would be the market leader in the electric-vehicle rental market, the government and the Ministry of Transportation would receive a lot of positive credit for reorganizing local motorbike service. Riders would have more ways to make money, and consumers would benefit from a low-cost service provided by locals.

# Objective of the project

The aim of this project is to develop a Mobile-based Application that will give customers the advantage of the locality of motorcycle taxis and cheaper prices. Along with increasing income to motorcycle taxis that have their market share taken from various ride-hailing nowadays, it also makes motorcycle taxis more reliable and fairer price. This project will match users with motorcycle taxis, clearly state the fare and let users select their own preference e.g., fast/slow ride.

# An overview of the system

This application is designed to bring benefits to both users and motorcycle taxis. Both customers and riders will use the same application, however, their interface will be different based on their user type.

For rider to get started, they will need to register through filling in their full name, citizen ID and reference number from Department of Land Transport, pay entrance fee including taking a picture of themselves to verify the identity of motorcycle taxis. This is to assure users that all motorcycle taxis in the application will be legal motorcycle taxis and to build confidence to customers that the rider is the approved one.

On the user side to register, it is necessary to verify identity through personal information.

The match making system starts when the user selects the pick-up location and destination they want to go to, selects payment method, and selects rider preferences. The motorcycle taxis in the surrounding location will be notified that there is new user’s ride booking. When a motorcycle taxi accepts a ride from any user, users will see motorcycle plate number and rider name, and then wait for a motorcycle taxi to pick up.

During the service, on motorcycle taxi side, there shall be an update to let the system know that the motorcycle taxi has arrived, on the way to the destination, or arrived at the destination.

When the service is completed. The customer will be charged for the ride fare via the selected payment method and gives a review the rider who serves them with rating and description.

Diagram

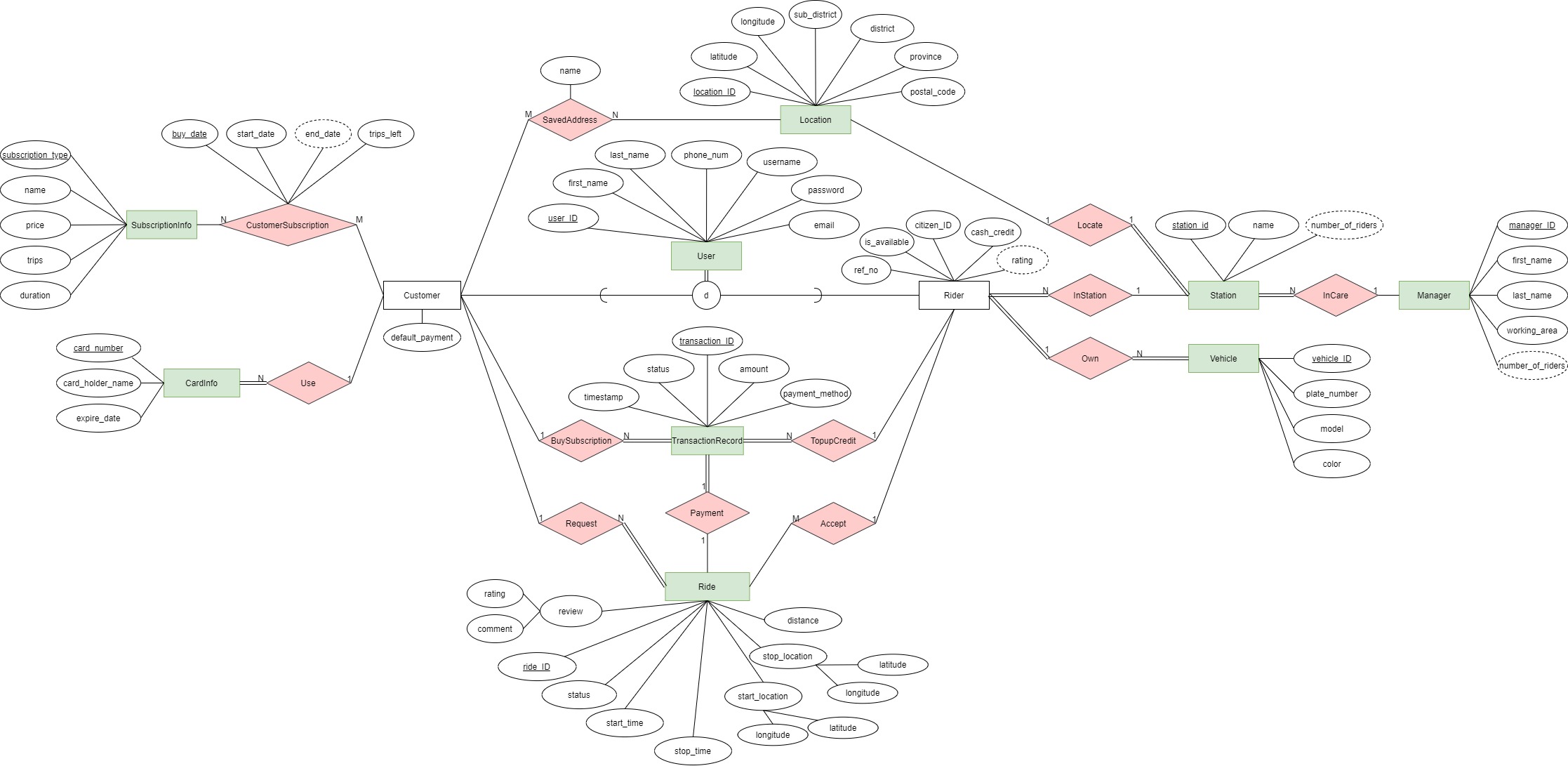
Description automatically generated

# System Functionalities

1. Register and manage account
   1. The system shall allow both user and rider to create a new account with a unique userID on the system.
   2. The system shall allow both user and rider to create a specific password corresponding to the user’s userID.
   3. The system shall allow both user and rider to select account type from two categories including user and rider.
   4. The system shall allow the user to fill-in their profile including citizen ID number, first-name, last-name, and phone number.
   5. The system shall allow the rider to fill-in their profile including Reference number, first-name, last-name, citizen ID number, and phone number.
   6. For rider to start the service, the system shall allow the rider to pay entrance fee via bank transfer, credit card or debit card. This entrance fee will return to rider as rider’s cash credit
   7. The system shall allow both user and rider to update their profile.
   8. The system shall allow the user to save their favorite location.
2. Login/logout system
   1. The system shall allow the user and rider to login/logout the system.
   2. When a user or a rider tries to login, the system shall validate their identity by userID and password.
3. Match riders to user
   1. The system shall allow riders to set their availability to either available or unavailable.
   2. The system shall make notifications to riders about ride requests made by users in their acceptable vicinity.
   3. The system shall allow riders to accept ride requests that are available.
   4. The system shall allow riders to decline available ride requests that are notified to them.
   5. The system shall allow riders to cancel their acceptance of a ride request.
   6. The system shall allow one rider only, at a time, to accept a ride request.
   7. The system shall allow the user to look up available riders by location.
   8. The system shall record every ride every rider has accepted.
4. Book a ride
   1. The system shall allow the user to look up available riders by location.
   2. The system shall allow the user to set their destination for the ride.
   3. The system should allow the user to choose their preference for the ride.
   4. The system shall allow the user to choose between booking a ride right away or booking a ride in advance.
   5. In case of a right-away ride, the system shall allow the user to cancel the ride before the ride is accepted, without any penalty.
   6. In case of an in-advance booked ride, the system shall allow the user to cancel the ride before the scheduled time, without any penalty.
   7. The system shall record every ride users have requested.
   8. The system shall allow the user to see the price rate of the requested ride.
   9. The system should allow users to see the predicted amount of time for the ride.
5. Initiate a ride
   1. The system shall make notification to the user about the acceptance of their ride request.
   2. The system shall show the user the profile of the rider who accepted their ride request.
   3. The system shall allow the user to be able to see the current location of the rider who accepted their ride request.
   4. The system shall make notification to the user of the arrival of the rider who accepted their ride request.
   5. The system shall allow users to cancel their rides that are currently in progress but with a penalty.
6. Make payment
   1. Before the user books a ride, the service shall allow the user to select their desired payment method.
   2. In case the user decides to pay the service by transferring to a bank account, the system shall allow the user to transfer service fee when the user reaches the destination.
   3. In case the user decides to make a payment automatically from their credit or debit card, the system shall automatically make a payment from that credit or debit card after the rider marks the service as done.
   4. In case the user decides to pay by cash, the system shall deduct the rider’s cash credit equivalent to the service fee for that ride after the rider marks the service as done. (User pays the rider when they reach the destination.)
   5. In case the rider’s cash credit is under 50 baht, the system shall allow rider to top-up credit by bank transfer, credit card, and debit card.
7. Make review
   1. In case the service is success, the system shall allow users that use the service to review their rider via anonymous comment and rate them from 0 to 5 after the ride.
   2. The system should allow the user to view comments and ratings they have given to past rides.
   3. The system should allow the rider to view comments and ratings given to them.

# ER Diagram / Document-based design schema

## ER Diagram

[](https://app.diagrams.net/#G1dIbtUo-i9g-JparickuJlgLznPkrMn7I)

click the image to redirect to bigger picture

## Document-based design schema

LOCATION

{

"type": "object",

"bsonType": "object",

"title": "Location",

"description": "Location of customer and station",

"required": [

"\_id", "latitude", "longitude", "sub\_district", "district", "province", "postal\_code"

],

"properties": {

"\_id": { "bsonType": "objectid" },

"latitude": { "bsonType": "double", "minimum": -90, "maximum": 90 },

"longitude": { "bsonType": "double", "minimum": -180, "maximum": 180 },

"sub\_district": { "bsonType": "string", "minLength": 1, "maxLength": 50 },

"district": { "bsonType": "string", "minLength": 1, "maxLength": 50 },

"province": { "bsonType": "string", "minLength": 1, "maxLength": 50 }

},

"patternProperties": {

"postal\_code": {

"bsonType": "string",

"minLength": 5, "maxLength": 5,

"pattern": "(1-9)(0-9)\*"

}

},

"minProperties": 8,

"dependencies": {

"locatedstation": {

"station": { "bsonType": "object" },

"numberofriders": { "bsonType": "int", "minimum": 0, "maxmum": 100 }

},

"customerSavedAddress": {

"bsonType": "array",

"items": {

"name": { "bsonType": "string", "minLength": 1, "maxLength": 50 },

"user\_id": { "bsonType": "string" }

},

"minItems": 0,

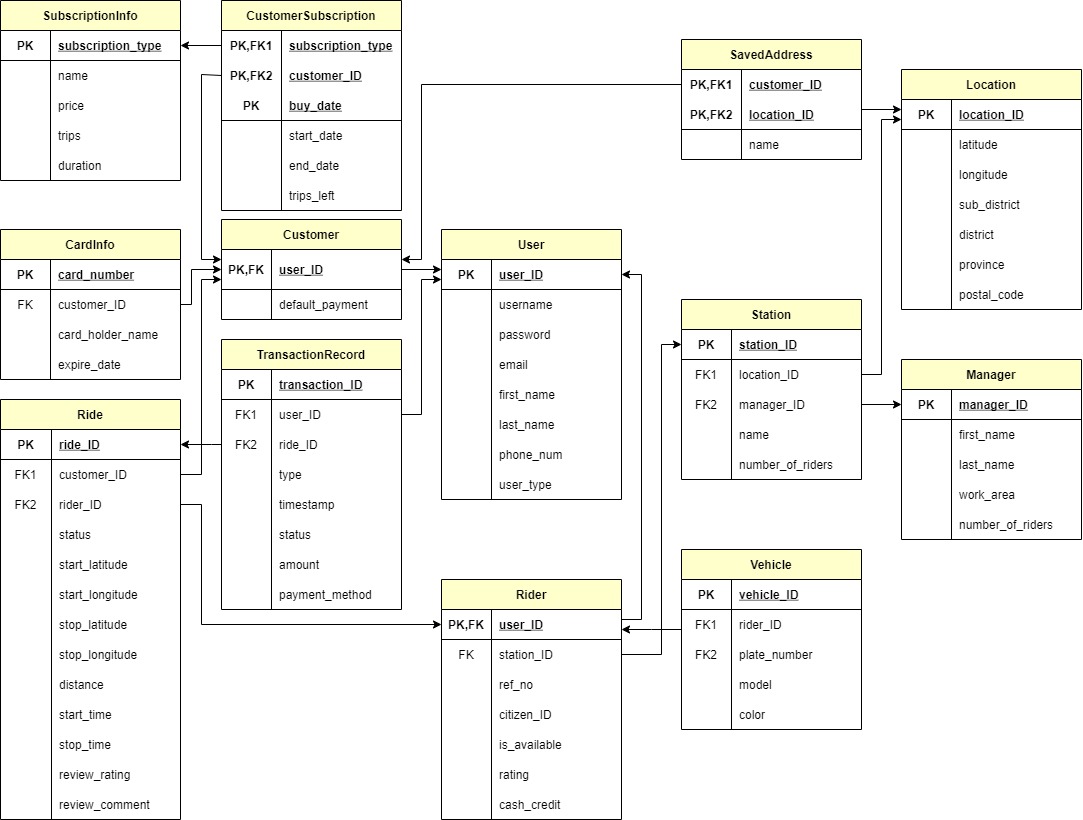
"uniqueItems": true

}

}

}

# Schema diagram

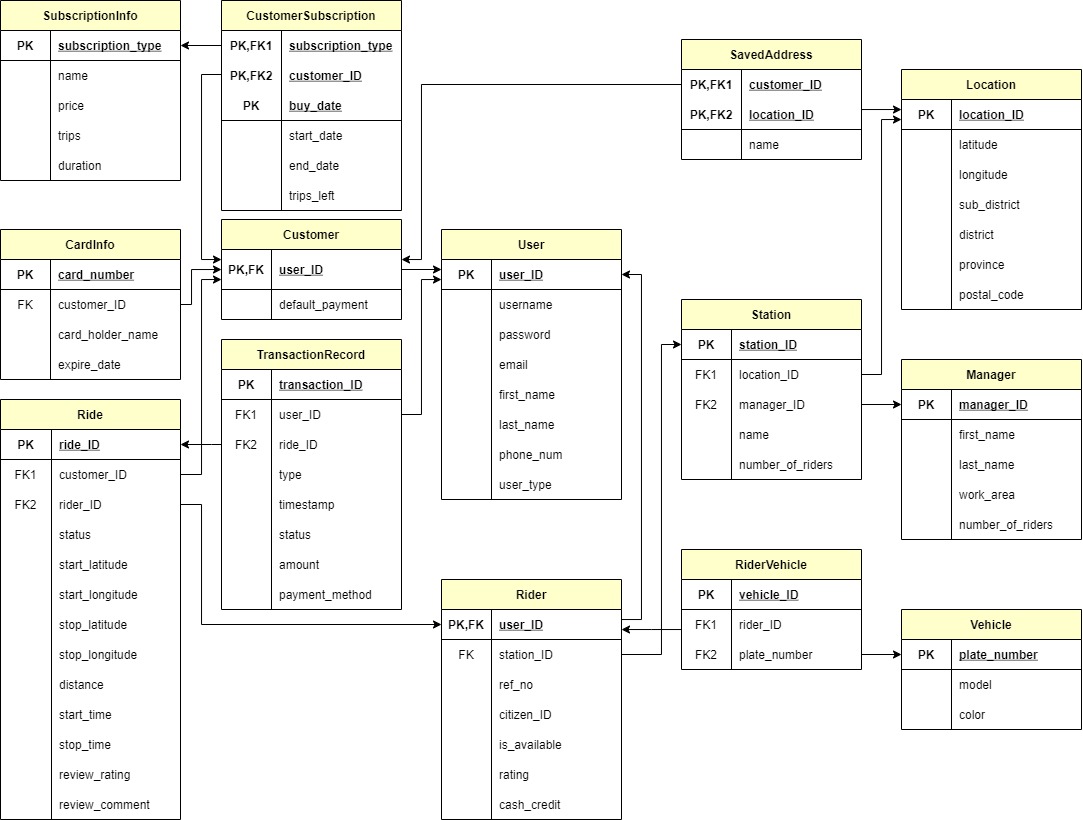


# Normalization of Relation

มีการทำ Functional Dependency ดังนี้

1. Vehicle: vehicle\_id→{rider\_ID,plate\_number} และ plate\_number→{model, color}
2. ส่วน Relation ที่เหลือ มี Functional Dependency เดียว คือ Primary Key → Others

จึงมี Schema diagram ของ Relation ที่มี 3NF แล้ว ดังนี้



# Data Dictionary

**Table of Entities**

|  |  |
| --- | --- |
| **Name** | **Description** |
| User | Generalization of Customer and Rider |
| Customer | Users who want to find a ride |
| CardInfo | Information of credit card |
| SubscriptionInfo | Information of subscription types |
| Location | Location of customer and station |
| Rider | Users who provide a ride |
| Station | Station of rider |
| Vehicle | Vehicle of rider |
| TransactionRecord | Record of transaction in system |
| Ride | Ride from rider that provide to customer |
| Manager | Person who takes care riders in area |

**Table of Relationships**

|  |  |
| --- | --- |
| **Name** | **Description** |
| CustomerSubscription | Customer subscribes to the system |
| Use | Customer uses credit cards |
| SavedAddress | Customer saves their favourite locations |
| Request | Customers request rides |
| Payment | How the ride is paid |
| Accept | Rider accepts rides |
| Locate | Station is located at location |
| InStation | Riders are in station |
| Own | Rider owns vehicles |
| InCare | Manager takes care of riders |
| BuySubscription | Customer pays their subscription fees |
| TopupCredit | Rider buys Top-Up credits |
| RiderVehicle | Rider owns Vehicle |

Note: Underline attribute stands for primary key and asterisk symbol (\*) stands for foreign key.

Entity Type Name: **User**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| user\_ID | CHAR(8) | Customer ID |  | No |
| username | VARCHAR(50) | Username of customer | Unique | No |
| password | TEXT | Hashed password of customer |  | No |
| email | VARCHAR(100) | Customer Email | Email, unique | No |
| first\_name | VARCHAR(50) | First name of customer |  | No |
| last\_name | VARCHAR(50) | Last name of customer |  | No |
| phone\_num | CHAR(10) | Phone Number | Phone number | No |
| user\_type | ENUM | Type of user | 'customer',  'rider' | No |

Entity Type Name: **Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| user\_ID\* | CHAR(8) | Customer ID |  | No |
| default\_payment | ENUM | Default payment of customer | 'cash',  'bank\_transfer',  'credit\_card' | No |

Entity Type Name: **CardInfo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| card\_number | CHAR(16) | Card Number | Credit card number | No |
| customer\_ID\* | CHAR(8) | Customer ID |  | No |
| card\_holder\_name | VARCHAR(16) | Shorten name of card holder |  | No |
| expire\_date | CHAR(4) | Expire date of credit card | Format: “MMYY” | No |

Entity Type Name: **SubscriptionInfo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| subscription\_type | CHAR(8) | Subscription type |  | No |
| name | VARCHAR(30) | Name of subscription type |  | No |
| price | U\_FLOAT | Price of subscription type |  | No |
| trips | U\_INT | Number of trips |  | No |
| duration | U\_INT | How long until expires (days) |  | No |

Entity Type Name: **Location**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| location\_ID | CHAR(8) | Location ID |  | No |
| latitude | FLOAT | Latitude of location | Values from  -90 to 90 | No |
| longitude | FLOAT | Longitude of location | Values from  -180 to 180 | No |
| sub\_district | VARCHAR(50) | Sub district of location |  | No |
| district | VARCHAR(50) | District of location |  | No |
| province | VARCHAR(50) | Province of location |  | No |
| postal\_code | CHAR(5) | Postal code of location |  | No |

Entity Type Name: **Rider**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| user\_ID\* | CHAR(8) | Rider ID |  | No |
| station\_ID\* | CHAR(8) | Station ID |  | No |
| ref\_no | CHAR(16) | Reference Number | Unique | No |
| citizen\_ID | CHAR(13) | Citizen ID | Citizen ID, unique | No |
| is\_available | BOOLEAN | Status available of rider |  | No |
| rating | U\_FLOAT | Rating of rider | Values from 1.0 to 5.0 | Yes |
| cash\_credit | U\_FLOAT | Cash Credit is what will be deducted when the rider is paid by cash; must be topped-up. |  | No |

Entity Type Name: **Station**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| station\_ID | CHAR(8) | Station ID |  | No |
| location\_ID\* | CHAR(8) | Location ID | Unique | No |
| manager\_ID\* | CHAR(8) | Manager ID |  | No |
| name | VARCHAR(50) | Station name | Unique | No |
| number\_of\_riders | U\_INT | Number of riders assigned at the station |  | No |

Entity Type Name: **Vehicle**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| plate\_number | VARCHAR(16) | License plate number of vehicle |  | No |
| model | VARCHAR(50) | Model of vehicle |  | No |
| color | VARCHAR(16) | Color of vehicle |  | No |

Entity Type Name: **TransactionRecord**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| transaction\_ID | CHAR(8) | Transaction ID |  | No |
| type | ENUM | Type for Transaction Record | 'top-up',  'subscription',  'ride' | No |
| timestamp | TIMESTAMP | Timestamp of Transaction |  | No |
| status | ENUM | Status of the transaction | 'Pending', 'Success', 'Declined' | No |
| amount | U\_FLOAT | Amount of payment |  | No |
| payment\_method | ENUM | Payment Method | 'cash',  'bank\_transfer',  'credit\_card' | No |

Entity Type Name: **TopUpTransaction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| transaction\_ID\* | CHAR(8) | Transaction ID |  | No |
| rider\_ID\* | CHAR(8) | Rider ID |  | Yes |

Entity Type Name: **SubscriptionTransaction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| transaction\_ID\* | CHAR(8) | Transaction ID |  | No |
| customer\_ID\* | CHAR(8) | Customer ID |  | Yes |

Entity Type Name: **RideTransaction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| transaction\_ID\* | CHAR(8) | Transaction ID |  | No |
| ride\_ID\* | CHAR(8) | Ride ID |  | Yes |

Entity Type Name: **Ride**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| ride\_ID | CHAR(8) | Ride ID |  | No |
| customer\_ID\* | CHAR(8) | Customer ID |  | Yes |
| rider\_ID\* | CHAR(8) | Rider ID |  | No |
| status | ENUM | Status of ride | 'matching',  'matched',  'cancelled',  'in\_transit',  'arrived',  'paid',  'reviewing',  'completed' | No |
| start\_latitude | FLOAT | Latitude which the service has started | Values from -90 to 90 | No |
| start\_longitude | FLOAT | Longitude which the service has started | Values from -180 to 180 | No |
| stop\_latitude | FLOAT | Latitude which the service has finished | Values from -90 to 90 | No |
| stop\_longitude | FLOAT | Longitude which the service has finished | Values from -180 to 180 | No |
| distance | U\_FLOAT | Real distance from start location to stop location |  | No |
| start\_time | DATETIME | Time the service has started. |  | No |
| stop\_time | DATETIME | Time the service has finished. |  | Yes |
| review\_rating | Small INT | Rating for this ride | Values from  0 to 5 | No |
| review\_comment | TEXT | Comment from customer for this ride |  | Yes |

Entity Type Name: **Manager**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| manager\_ID | CHAR(8) | Manager ID |  | No |
| first\_name | VARCHAR(50) | First name of Manager |  | No |
| last\_name | VARCHAR(50) | Last name of Manager |  | No |
| working\_area | VARCHAR(64) | Working Area of manager |  | No |
| number\_of\_riders | U\_INT | Number of riders under this manager |  | No |

Relationship Type Name: **SavedAddress**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| customer\_ID\* | CHAR(8) | Customer ID |  | No |
| location\_ID\* | CHAR(8) | Location ID |  | No |
| name | VARCHAR(50) | Name of saved location |  | No |

Relationship Type Name: **CustomerSubscription**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| subscription\_type\* | CHAR(8) | Subscription type |  | No |
| customer\_ID\* | CHAR(8) | Customer ID |  | No |
| buy\_date | DATETIME | Buy date of subscription |  | No |
| start\_date | DATE | Start date of subscription | Smaller than end\_date | Yes |
| end\_date | DATE | End date of subscription | If start\_date is null, end\_date = buy\_date + 1.5\*duration  Else, end\_date = min(start\_date + duration, end\_date) | No |
| trips\_left | U\_INT | Amount of trips left |  | No |

Relationship Type Name: **RiderVehicle**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Descriptive Name** | **Valid Values** | **Allow Nulls** |
| vehicle\_ID | CHAR(8) | Vehicle ID |  | No |
| rider\_ID\* | CHAR(8) | Rider ID |  | No |
| plate\_number\* | VARCHAR(16) | License plate number of vehicle | Unique | No |

# Indexing

**Relation and Attributes to do indexing**

|  |
| --- |
| Ride <rider\_ID, ride\_ID, review\_rating> |

**Index structure**

|  |
| --- |
| Clustered B+ tree |

**Reasons**

เนื่องจากเราต้องทำการอัปเดท review\_rating ของ rider ใหม่ทุกครั้งที่ customer ทำการ review เข้ามา ทำให้เราต้องค้นหา records ใน ride ตาม rider\_ID หลายครั้ง จึงทำ indexing โดยให้ rider\_ID อยู่อันดับแรก

เนื่องจากเราต้องการคิด rating ตามจำนวน ride และด้วยข้อจำกัดของ MySQL การทำ Clustered index จำเป็นต้องทำบน search key ที่ unique และ not null จึงต้องเพิ่ม ride\_ID เข้าไปด้วย (rider\_ID ไม่ unique)

นอกจากนี้การที่นำ review\_rating มาทำ indexing ด้วยเพราะว่าในการคำนวณ review\_rating ของ rider เราใช้แค่ column review\_rating เพียง column เดียวในการหาค่า average review\_rating ทำให้ไม่มีความจำเป็นจะต้องอ่านข้อมูลทั้ง record ออกมา ดังนั้น การนำ column review\_rating มาทำ indexing ด้วยจึงทำให้ index นี้เป็นการทำ index แบบ index-only plans ซึ่งช่วยเพิ่มประสิทธิภาพในการ query มากขึ้น

# Stored procedures

## Get rider from station

ทำการค้นหา rider ที่อยู่ใน station โดยใช้ชื่อของ station เป็นตัวค้นหา

|  |
| --- |
| DELIMITER $$  CREATE PROCEDURE GetRiderFromStation(IN stationName varchar(50))  BEGIN  SELECT U.first\_name, U.last\_name, U.phone\_num, R.ref\_no,  R.citizen\_ID, R.is\_available, R.rating  FROM rider R natural join station S natural join user U  WHERE S.name = stationName;  END$$  DELIMITER ; |

**Execution**



**Result**

Table

Description automatically generated with medium confidence

## Update Rider Station

เปลี่ยน station ที่ rider อยู่ โดยจะมีการไปเรียก trigger เพื่อลดจำนวน rider ที่อยู่ใน station และ manager เก่า และ เพิ่มจำนวน rider ที่อยู่ใน station และ manager ใหม่

|  |
| --- |
| DELIMITER $$  CREATE PROCEDURE UpdateRiderStation( IN i\_user\_ID CHAR(8), IN i\_station\_ID CHAR(8) )  BEGIN  DECLARE exit handler for sqlexception  Begin  Rollback;  Resignal;  End;  start transaction;  update Rider  set station\_ID = i\_station\_ID  where user\_ID = i\_user\_ID;  commit;  END$$  DELIMITER ; |

**Execution**



**Result**

Before:

|  |  |
| --- | --- |
| Rider | Station |
| A screenshot of a computer  Description automatically generated with medium confidence | A picture containing text, scoreboard  Description automatically generated |

After:

|  |  |
| --- | --- |
| Rider | Station |
|  |  |

# Stored function

## Price Calculation

คำนวณ และ return ค่าเดินทางของ ride โดยคำนวณจาก distance attribute ของ ride

|  |
| --- |
| DELIMITER $$  CREATE FUNCTION calcPrice(dist float)  RETURNS INT  DETERMINISTIC  BEGIN  DECLARE price INT;  DECLARE remDist float;  IF dist > 5 THEN  BEGIN  set remDist = dist - 5;  IF MOD(remDist,1) < 0.5 THEN set remDist = remDist - MOD(remDist,1);  ELSE set remDist = remDist - MOD(remDist,1) + 1;  END IF;  set price = (48 + remDist\*12);  END;  ELSEIF (dist <= 5 AND dist > 2.5) THEN  BEGIN  set remDist = dist - 2.5;  IF MOD(remDist,1) < 0.5 THEN set remDist = remDist - MOD(remDist,1);  ELSE set remDist = remDist - MOD(remDist,1) + 1;  END IF;  set price = (30 + remDist\*6);  END;  ELSEIF (dist <= 2.5 AND dist > 0) THEN  set price = 30;  END IF;  RETURN (price);  END $$  DELIMITER ; |

**Execution and Result**



Graphical user interface, table

Description automatically generated

## Calculate Total Ride Time

คำนวณ และ return เวลาตั้งแต่ ride เริ่มต้น จนจบ โดยคำนวณจาก start\_time attribute และ stop\_time attribute   
ของ ride

|  |
| --- |
| DELIMITER $$  CREATE FUNCTION calcRideT(startT datetime, stopT datetime)  RETURNS time  DETERMINISTIC  BEGIN  DECLARE totalT INT;  set totalT = abs(timediff(stopT, startT));  RETURN (totalT);  END $$  DELIMITER ; |

**Execution and Result**

Calendar

Description automatically generated with low confidence

# Triggers

## Update Number of riders

ใช้สำหรับแก้ไขจำนวน Rider ใน Station และ Manager

|  |
| --- |
| DELIMITER //  CREATE TRIGGER updateNumberOfRiders  AFTER update ON rider for each row  BEGIN  IF NEW.station\_ID != OLD.station\_ID THEN  BEGIN  DECLARE new\_station\_manager char(8);  DECLARE old\_station\_manager char(8);    select manager\_ID into new\_station\_manager from station  where station\_ID = NEW.station\_ID;    select manager\_ID into old\_station\_manager from station  where station\_ID = OLD.station\_ID;    update station set number\_of\_riders = number\_of\_riders + 1  where station\_ID = NEW.station\_ID;    update station set number\_of\_riders = number\_of\_riders - 1  where station\_ID = OLD.station\_ID;    IF new\_station\_manager != old\_station\_manager THEN  BEGIN  update manager set number\_of\_riders = number\_of\_riders + 1  where manager\_ID = new\_station\_manager;  update manager set number\_of\_riders = number\_of\_riders - 1  where manager\_ID = old\_station\_manager;  END;  END IF;  END;  END IF;  END //  DELIMITER ; |

**Execution and Result**

Before:

|  |  |
| --- | --- |
| Rider | Station |
| A screenshot of a computer  Description automatically generated with medium confidence | A picture containing text, scoreboard  Description automatically generated |

After:

|  |  |
| --- | --- |
| Rider | Station |
| A screenshot of a computer  Description automatically generated with medium confidence | A picture containing text, scoreboard  Description automatically generated |

**Reason**

เนื่องจากการเพิ่ม/ลดจำนวนของ number\_of\_riders ที่อยู่ใน station และ manager เป็นผลกระทบที่เกิดหลังจากเกิดการ update column station\_ID ใน rider จึงนำส่วนของการปรับแก้จำนวน rider มาทำเป็น trigger แทนที่จะใส่ลงไปใน procedure UpdateRiderStation โดยตรง เพื่อให้คำสั่งภายใน procedure มีความชัดเจนและสื่อความหมายตรงตามจุดประสงค์ของการเรียกใช้

## Delete Account Log

เก็บประวัติการลบ Account โดยมีตาราง TriggerTime ซึ่งมี attribute คือ datetime (เก็บเวลาที่ถูกลบ) และ userID

|  |
| --- |
| CREATE TABLE TriggerTime(time datetime NOT NULL, userID char(8) NOT NULL);  CREATE TRIGGER deleteAccountLog  AFTER DELETE  ON User FOR EACH ROW  INSERT INTO TriggerTime VALUES(NOW(), OLD.user\_ID); |

**Execution and Result**

DELETE \* FROM user WHERE user\_ID = 'UID00007';

Before:

|  |  |
| --- | --- |
| User | TriggerTime |
|  |  |

After:

|  |  |
| --- | --- |
| User | TriggerTime |
|  |  |

**Reason**

สามารถนำไปใช้ในแผนกอื่น ๆ ในกรณีฉุกเฉินได้ เช่น การตรวจสอบความถี่ในการลบ Account หากมีการลบหลาย Account ในเวลาอันสั้น ควรจะต้องตรวจสอบความผิดปกติของบริษัท

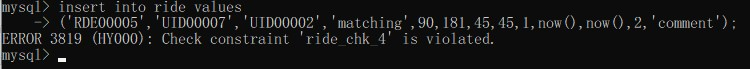
# Integrity

## Domain Integrity: Valid value of attribute

Relation **Ride** schema:

|  |
| --- |
| create table if not exists Ride(  ride\_ID char(8) primary key,  customer\_ID char(8),  rider\_ID char(8),  status enum('matching','matched','cancelled','in\_transit','arrived','paid','reviewing','completed') not null,  start\_latitude float not null,  start\_longitude float not null,  stop\_latitude float not null,  stop\_longitude float not null,  distance float not null,  start\_time datetime not null default(now()),  stop\_time datetime,  review\_rating smallint,  review\_comment text,  foreign key (rider\_ID) references Rider(user\_ID) on update cascade on delete set null,  foreign key (customer\_ID) references Customer(user\_ID) on update cascade on delete set null,  check(distance >= 0),  check(review\_rating >= 1 and review\_rating <= 5),  check(start\_latitude >= -90 and start\_latitude <= 90),  **check(start\_longitude >= -180 and start\_longitude <= 180),**  check(stop\_latitude >= -90 and stop\_latitude <= 90),  check(stop\_longitude >= -180 and stop\_longitude <= 180)  ); |

**Execution and Results:**



## Domain Integrity: Unique attribute

Relation **User** schema:

|  |
| --- |
| create table if not exists User(  user\_ID char(8) primary key,  username varchar(50) not null,  password text not null,  email varchar(100) not null **unique**,  first\_name varchar(50) not null,  last\_name varchar(50) not null,  phone\_num char(10) not null,  user\_type enum('customer','rider') not null  ); |

**Execution and Results:**

Graphical user interface, text, application, email

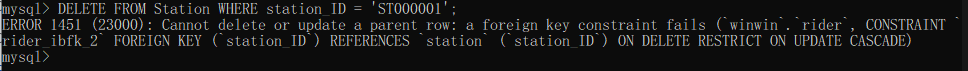
Description automatically generated

## Referential Integrity: Delete operation

Relation **Rider** schema:

|  |
| --- |
| create table if not exists Rider(      user\_ID char(8) primary key,      station\_ID char(8) not null,      ref\_no char(16) not null unique,      citizen\_ID char(13) not null unique,      is\_available boolean not null default true,      rating float,      cash\_credit float not null default 0,      foreign key (user\_ID) references User(user\_ID) on update cascade on delete cascade,      foreign key (station\_ID) references **Station(station\_ID)** on update cascade **on delete restrict**,      check(rating >= 1 and rating <= 5 ),      check(cash\_credit >= 0)  ); |

**Execution and Results:**



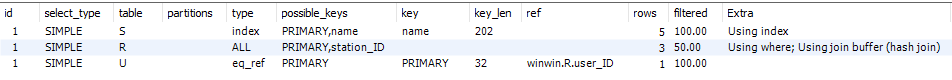
ลบไม่ได้เพราะใน ST000001 มี rider 2 คน



ลบได้เพราะใน ST000001 ไม่มี rider เลย

# Execution path

|  |
| --- |
| SELECT U.first\_name, U.last\_name, U.phone\_num, R.ref\_no, R.citizen\_ID, R.is\_available, R.rating  FROM rider R natural join station S natural join user U  WHERE S.name = 'Station1' or R.user\_ID='UID00005'; |



Diagram

Description automatically generated

**Result**

A picture containing table

Description automatically generated

|  |
| --- |
| (SELECT U.first\_name, U.last\_name, U.phone\_num, R.ref\_no, R.citizen\_ID, R.is\_available, R.rating  FROM rider R, station S, user U  WHERE U.user\_ID=R.user\_ID and R.station\_ID=S.station\_ID and S.name = 'Station1')  UNION  (SELECT U.first\_name, U.last\_name, U.phone\_num, R.ref\_no, R.citizen\_ID, R.is\_available, R.rating  FROM rider R, user U  WHERE U.user\_ID=R.user\_ID and R.user\_ID='UID00005'); |

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

**Result**

A picture containing table

Description automatically generated

# SQL complex query

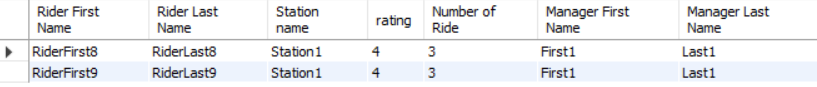
## Riders with their manager and special conditions

Query แสดงข้อมูล Rider ที่มี Rating มากกว่า 1 และมีจำนวนการให้บริการมากกว่า 1 ครั้งโดยจะแสดง Attributes คือ

* Rider full name (firstname and lastname)
* Station name
* rating
* จำนวนการให้บริการ (ในชื่อ Number of Ride)
* Manager full name (firstname and lastname)

|  |
| --- |
| **SELECT**  U.first\_name as "Rider First Name",  U.last\_name as "Rider Last Name",  S.name as "Station name",  rating,  count(ride\_ID) as "Number of Ride",  M.first\_name as "Manager First Name",  M.last\_name as "Manager Last Name"  **FROM**  rider left join ride on rider.user\_ID = ride.rider\_ID  natural join station S natural join user U,  manager M  **WHERE**  rating > 1 and M.manager\_ID =  "MN000001"  and S.manager\_ID = M.manager\_ID  **GROUP BY** user\_ID  **HAVING** count(ride\_ID)>1; |

**Example results**



## Top Spender in November 2021

แสดง First Name, Last Name และผลรวมของยอดค่าใช้จ่ายที่เกิดจาก Ride (ในชื่อ Expense) ของ User 5 คนแรกที่มี Expense มากที่สุดในเดือนพฤศจิกายนปี 2021 โดยเรียงลำดับจากมากไปน้อย

เงื่อนไข:

* นับเฉพาะที่ Trasactionrecord.Status = ‘success’
* หากมีอันดับร่วมจะแสดง Top Spender ที่มีอันดับร่วมด้วย

|  |
| --- |
| **create view** topSpenderView as  **SELECT** SUM(T.amount) as Expense5  **FROM** User U, Customer C, Ride R, ridetransaction RT, transactionrecord T  **WHERE**  U.user\_ID = C.user\_ID AND  C.user\_ID = R.customer\_ID AND  R.ride\_ID = RT.ride\_ID AND RT.transaction\_ID = T.transaction\_ID AND  T.status = 'success' AND  T.timestamp BETWEEN DATE("2021-11-01") AND DATE("2021-11-30")  **GROUP BY** U.user\_ID  **ORDER BY** Expense5 DESC  **limit** 5; |
| **SELECT** U.first\_name AS 'First Name', U.last\_name AS 'Last Name', SUM(T.amount) as Expense  **FROM** User U, Customer C, Ride R, ridetransaction RT, transactionrecord T  **WHERE**  U.user\_ID = C.user\_ID AND  C.user\_ID = R.customer\_ID AND  R.ride\_ID = RT.ride\_ID AND RT.transaction\_ID = T.transaction\_ID AND  T.status = 'success'  AND T.timestamp BETWEEN DATE("2021-11-01") AND DATE("2021-11-30")  **GROUP BY** U.user\_ID  **HAVING** Expense >= any(SELECT Expense5 from topSpenderView)  **ORDER BY** Expense DESC; |

**Example results**

**Table

Description automatically generated**

## Ordering gross subscription sales in 2021 with top spender for each subscription

Query แสดง

* subscription\_type (แทนด้วย SubscriptionType)
* เงินที่ Subscription ทำได้ในปี 2021 (แทนด้วย GrossSale)
* user\_ID ของคนที่ซื้อ Subscription นั้น ๆ มากที่สุดในปี 2021 (แทนด้วย TopSpender)
* เงินที่ user\_ID ข้อที่แล้วนั้นเสียไปกับการซื้อ Subscription ดังกล่าวในปี 2021 (แทนด้วย SpendingAmount)

ข้อกำหนดการเรียงลำดับ

* จะเรียงลำดับ Query จาก SubscriptionType ที่ทำในเงินในปี 2021 ได้มากที่สุดไปน้อยที่สุด  
  (เรียงตาม GrossSale)
* หากมี TopSpender หลายคน ให้เลือกคนที่ซื้อ Subscription นั้นเป็นคนแรกของปี 2021

|  |
| --- |
| **SELECT** subscription\_type as SubscriptionType, sum(Price) as GrossSale,  user\_ID as TopSpender, max(Price) as SpendingAmount  **FROM** (  **SELECT** subscription\_type, user\_ID, sum(price) as Price  **FROM** customersubscription natural join subscriptioninfo  left join customer on customersubscription.customer\_ID = customer.user\_id  natural join user  **WHERE** start\_date between date('2021-1-1') and date('2021-12-31')  and end\_date between date('2021-1-1') and date('2021-12-31')  **GROUP BY** subscription\_type, user\_ID  **ORDER BY** Price DESC, buy\_date  ) T  **GROUP BY** subscription\_type  **ORDER BY** GrossSale DESC; |

**Example results**

Table

Description automatically generated

# Appendix: External Resource

Implemented system <https://github.com/atiwat1113/db_system>

UI design <https://www.figma.com/file/t4T5hITYcZCSgz7ajLSWXO/Waterfall?node-id=2%3A6139>