INSY 661 Database Group Project

- McGill Marketplace -

Group #10

260825937 Jenny Yao

260901022 Xenia Sozonoff

261098992 Jeongho Pyo

260795028 Iris Liu

Table of Content

Stage 1

Overview of the Business Scenario

McGill Marketplace is an online trading platform that enables students to purchase or sell their products. It provides students with a safe transaction environment based on a secure login system by using university email addresses. It is expected to have a significant effect on student communities as well. Here is an episode of using McGill Marketplace.

A newly enrolled student at McGill University, Jake, needs a coffee machine. He has had a hard time getting one from a reliable person since he is a newcomer. Also, there was no brand-new machine at an affordable price. Fortunately, he found the university platform “McGill Marketplace.” He was able to sign up easily thanks to his university account (e.g. jake.henderson@mail.mcgill.ca). After the registration, he began to search for the machines.

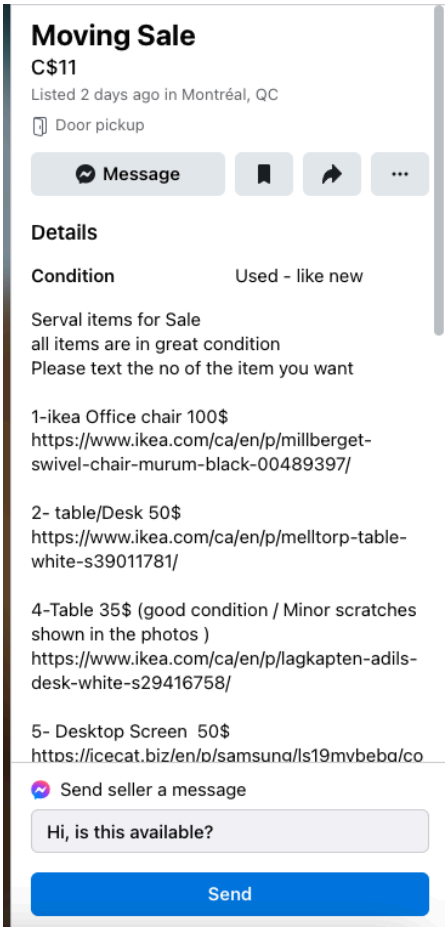
First, he typed “Coffee machine” and got numerous outcomes on the page. After that, he narrowed them down by clicking the "Home appliance" category so that products from other categories cannot be displayed. As a result, a reduced number of results were shown on the page. Subsequently, he browsed every post to check the average rating and the result of the sentimental analysis of reviews. In addition, he was able to see how popular the post is. He made sure the preferred transaction method and posted language be what he wanted. Consequently, He found a fine coffee machine that perfectly satisfied his requirements.

Through this experience, he discovered outstanding features of the McGill Marketplace. As a platform with restricted access, its members can experience secure transactions when they purchase or sell items. This can facilitate the prosperity of the platform. On top of that, the platform specializes in customization so that users are able to easily find what they want. Therefore, McGill University can gain credibility from students by utilizing the McGill Marketplace platform. Students can also benefit from the platform in that it guarantees safe transactions and individualized search results.

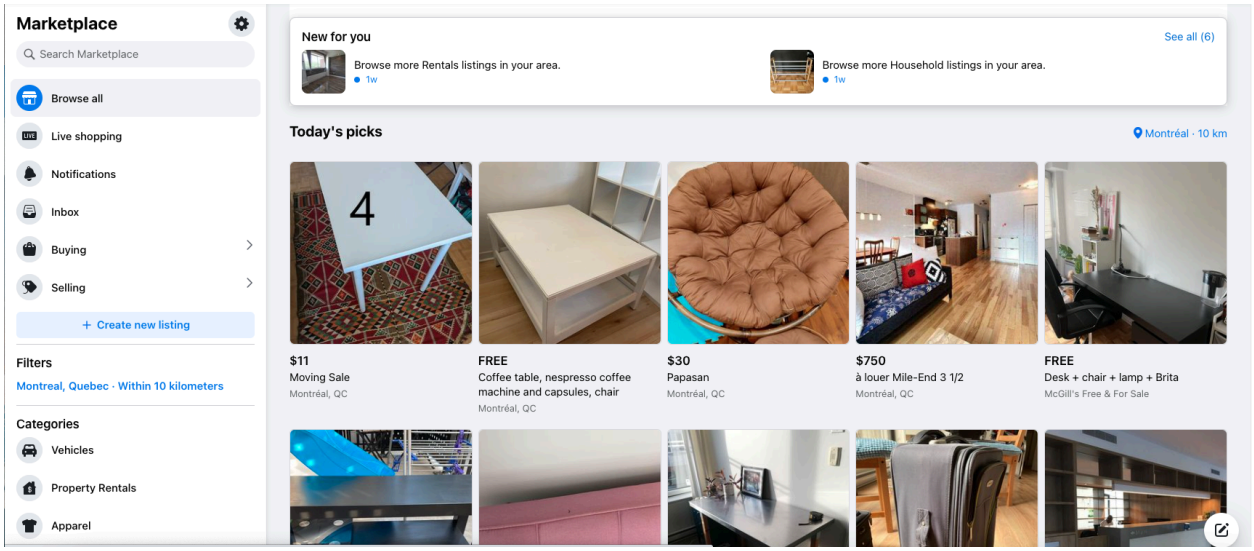
Clarification:

The idea of McGill Marketplace was inspired by Facebook Marketplace, but we did not intend to recreate Facebook Marketplace. McGill Marketplace targets McGill students exclusively and, thus, has different business functionalities. The screenshots below are only for reference purposes.

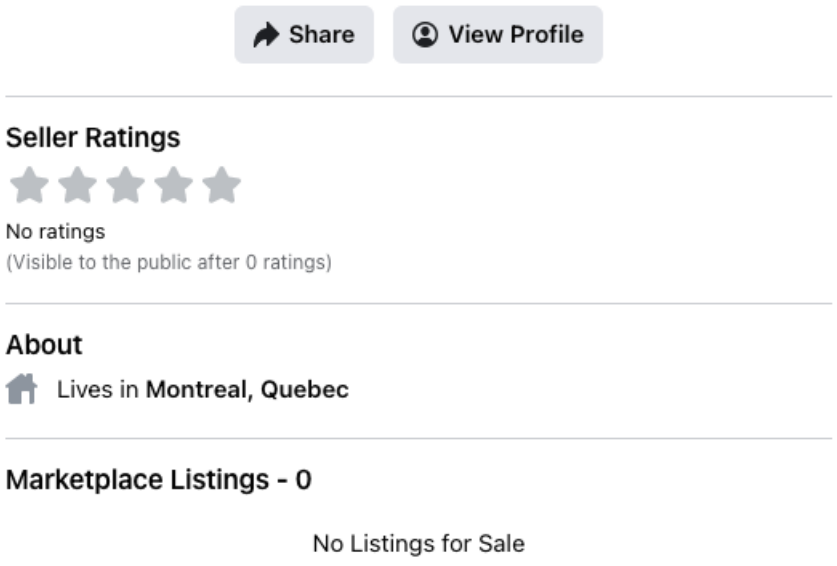
Posting an item for sale:



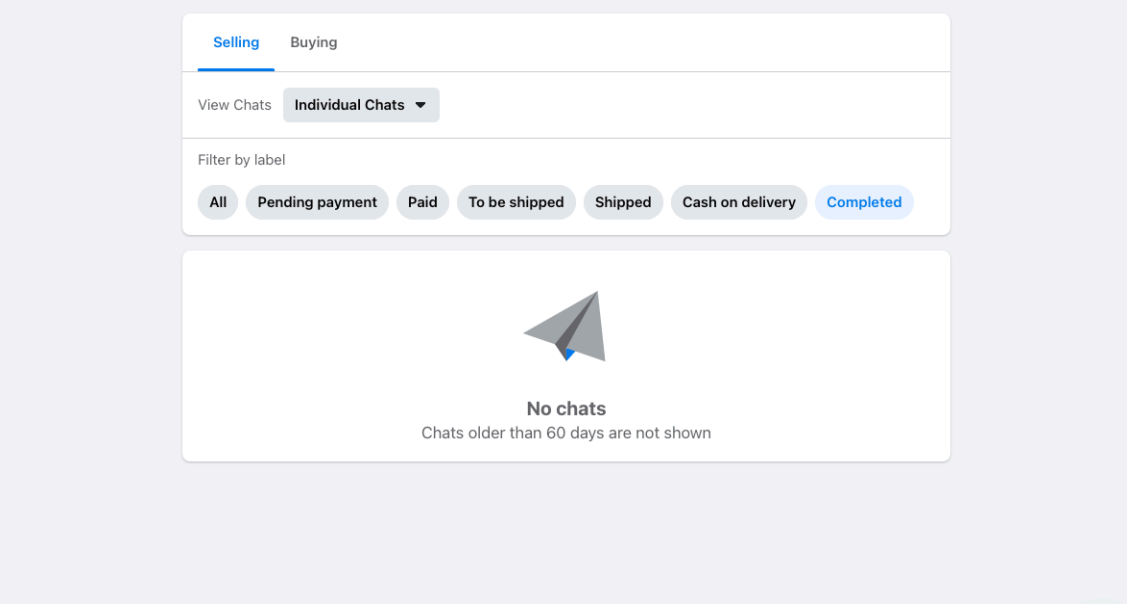
Searching a post based on distance and categories:



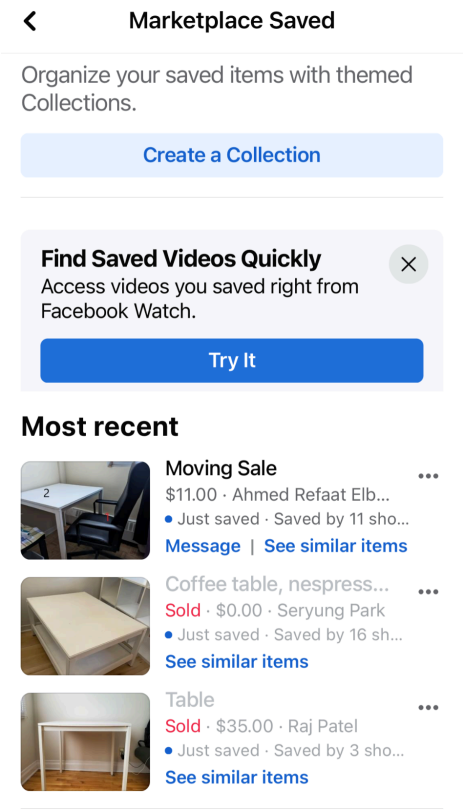
Displaying user's information, location, and review:



Keeping track of the status of transactions:



Saving a post for later purchase:



Mission Statement and Objective

Mission Statement:

The McGill Marketplace is an authorized platform provided for McGill students to trade safely, pertinently, and efficiently while building a sustainable education society.

Objective:

To maintain (enter, update and delete) data on posts.

To maintain (enter, update and delete) data on orders.

To maintain (enter, update and delete) data on students.

To maintain (enter, update and delete) data on students' reviews.

To maintain (enter, update and delete) data on products.

To maintain (enter, update and delete) data on location.

To maintain (enter, update and delete) data on students' favorite posts.

To perform searches on posts.

To perform searches on orders.

To perform searches on students.

To perform searches on students' reviews.

To perform searches on products.

To perform searches on location.

To perform searches on students' favorite posts.

To track the status of orders for students.

To track the status of posts for students.

To track the status of products.

To track products for orders.

To report on posts.

To report on orders.

To report on students.

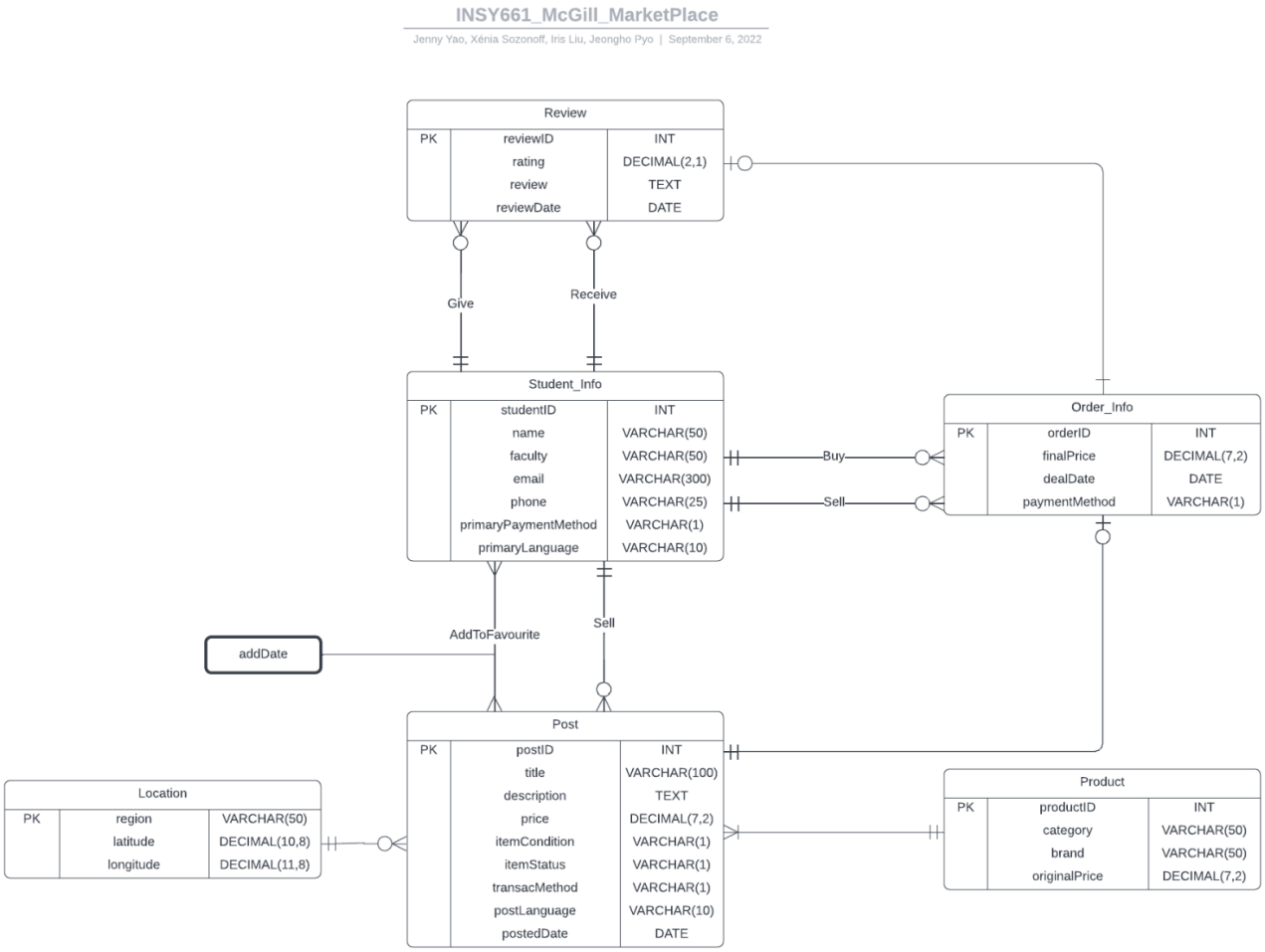
To report on students’ reviews.

To report on products.

To report on location.

To report on students’ favorite posts.

ER Diagram


INSY661_McGill_MarketPlace
Jenny Yao, Xénia Sozonoff, Iris Liu, Jeongho Pyo | September 6, 2022
Review
PK
reviewID
rating
review
reviewDate
INT
DECIMAL(2,1)
TEXT
DATE
Give
Receive
Student_Info
PK
studentID
name
faculty
email
phone
primaryPaymentMethod
primaryLanguage
INT
VARCHAR(50)
VARCHAR(50)
VARCHAR(300)
VARCHAR(25)
VARCHAR(1)
VARCHAR(10)
Buy
Sell
Order_Info
PK
orderID
finalPrice
dealDate
paymentMethod
INT
DECIMAL(7,2)
DATE
VARCHAR(1)
AddToFavourite
Sell
addDate
Post
PK
postID
title
description
price
itemCondition
itemStatus
transacMethod
postLanguage
postedDate
INT
VARCHAR(100)
TEXT
DECIMAL(7,2)
VARCHAR(1)
VARCHAR(1)
VARCHAR(1)
VARCHAR(10)
DATE
Location
PK
region
latitude
longitude
VARCHAR(50)
DECIMAL(10,8)
DECIMAL(11,8)
Product
PK
productID
category
brand
originalPrice
INT
VARCHAR(50)
VARCHAR(50)
DECIMAL(7,2)

Data Dictionary

Entity Name	Description	Aliases	Occurrence
Post	Contains selling information of one product that seller wants to sell	Selling Information	One post can be posted by one student. One post can result in one order. One post contains one product.
Order_Info	An entity that holds information about the purchase when the deal is completed	Dealing Information	One order contains one post. One order can have many reviews.
Review	Contains all the feedback that the students gave to the person who sold them the product	Comment	One review writes by one student for one product.
Student_Info	Contains the informations of the student using McGill Market Place	Personal Information	One student can give and receive many reviews. One student can sell/buy many products. One student can buy/sell one or many orders. One student has one favourite list.
Product	Contains the informations of the product on sale	Item	One product can be posted in one post.
Location	Contains location information of student who sent the post	Geographical Information	One location can belong to many posts.

Entity Name	Attributes	Description	Data Type	Nulls	Multivalued	Derived	Default
Post	postID	Unique ID for each post	int	No	No	No	None

	title	Title of post	varchar(100)	No	No	No	None
	productID	Unique ID for each product (refer to Product.productID)	int				
	description	Description of item	text	Yes	No	No	None
	price	Price of item	decimal(7,2)	No	No	No	None
	itemCondition	Condition of item (b=brand bew, l=like new, u=used, f=fair)	varchar(1)	No	No	No	None
	sellerID	Student ID of seller (refer to Student_Info.studentID)	int				
	itemStatus	Current status of post (a=avaliabe,o=on hold, s=sold)	varchar(1)	No	No	No	None
	transacMethod	Seller's perferred transaction method (c=cash, t=online transfer)	varchar(1)	No	No	No	None
	region	Region in Montreal (refer to Location.region)	varchar(50)				
	postLanguage	Language of post (english, french)	varchar(10)	yes	No	No	None
	postedDate	Date of post	date	No	No	No	None

Order_Info	orderID	Unique ID for each order	int	No	No	No	None
	postID	Unique ID for each post (refer to Post.postID)	int				
	buyerID	Student ID of buyer (refer to Student_Info.studentID)	int				
	sellerID	Student ID of seller (refer to Student_Info.studentID)	int				
	finalPrice	Price when deal is completed	decimal(7,2)	No	No	No	None
	dealDate	Date when deal is completed	date	No	No	No	None
	paymentMethod	Payment method of order (c=cash, t=online transfer)	varchar(1)	No	No	No	None
Review	reviewID	Unique ID for each review	int	No	No	No	None
	toStudent	Student who sends review (refer to Student_Info.studentID)	int				
	orderID	Unique ID for each order (refer to	int				

		Order.OrderID)					
	rating	Rating after student's order completed	decimal(2,1)	No	No	No	None
	review	Review after student's order completed	text	No	No	No	None
	fromStudent	Student who recievesr review (refer to Student_Info.studentID)	int				
	reviewDate	Date of each review posted in yyyy-mm-dd format	date	No	No	No	None
Student_ Info	studentID	Unique ID for each student	int	No	No	No	None
	name	Student Name	varchar(50)	No	No	No	None
	faculty	Student Faculty	varchar(50)	No	No	No	None
	email	Student Email	varchar(300)	No	No	No	None
	phone	Student Phone	varchar(25)	No	No	No	None
	primaryPayme ntMethod	Payment method of each student ((c=cash, t=online transfer))	varchar(1)	No	No	No	None
	primaryLangua ge	First language of each student (english, french)	varchar(10)	No	No	No	None
Product	productID	Unique ID for each product	int	No	No	No	None

	category	The category of each product	varchar(50)	No	No	No	None
	brand	The brand of each product	varchar(50)	No	No	No	None
	originalPrice	The price of each product	decimal(7,2)	No	No	No	None
Location	region	Region in Montreal	varchar(50)	Yes	No	No	None
	latitude	Latitude of location	decimal(10, 8)	Yes	No	No	None
	longtitude	Longtitude of location	decimal(11,8)	Yes	No	No	None
	addDate (Relationship attribute)	Date of post added to favourite	date	No	No	No	None

Relational Schema

Student_Info (studentID, name, faculty, email, phone, primaryPaymentMethod, primaryLanguage)

Primary Key: studentID

Product (productID, category, brand, originalPrice)

Primary Key: productID

Location (region, latitude, longitude)

Primary Key: region

Post (postID, title, productID, description, price, itemCondition, sellerID, itemStatus, transacMethod, region, postLanguage, postedDate)

Primary Key: postID

Foriegn Key: productID References Product(productID)

Foriegn Key: sellerID References Student_Info(studentID)

Foriegn Key: region References Location(region)

Favourite (studentID, postID, addDate)

Primary Key: studentID, postID

Foreign Key: studentID References Student_Info(studentID)

Foreign Key: postID References Post(postID)

Order_Info (orderID, postID, buyerID, sellerID, finalPrice, dealDate, paymentMethod)

Primary Key: orderID

Foreign Key: postID References Post(postID)

Foreign Key: buyerID References Student_Info(studentID)

Foreign Key: sellerID References Student_Info(studentID)

Review (reviewID, toStudent, orderID, rating, review, fromStudent, reviewDate)

Primary Key: reviewID

Foreign Key: toStudent References Student_Info(studentID)

Foreign Key: fromStudent References Student_Info(studentID)

Foreign Key: orderID References Order_Info(orderID)

Stage 2

SQL Scripts for Creating Tables

```
CREATE TABLE Student_Info (
        studentID INT NOT NULL,
        name varchar(50) NOT NULL,
        faculty varchar(50) NOT NULL,
        email varchar(300) NOT NULL,
        phone varchar(25) NOT NULL,
        primaryPaymentMethod varchar(1) NOT NULL,
        primaryLanguage varchar(10) NOT NULL,
        PRIMARY KEY (studentID));

CREATE TABLE Product(
        productID INT NOT NULL,
        category varchar(50) NOT NULL,
        brand varchar(50) NOT NULL,
        originalPrice DECIMAL(7,2) NOT NULL,
        PRIMARY KEY (productID));

CREATE TABLE Location(
        region varchar(50),
        latitude DECIMAL(10,8),
```

```
    longitude DECIMAL(11,9),
    PRIMARY KEY (region)
    );

CREATE TABLE Post(
    postID INT NOT NULL,
    title varchar(100) NOT NULL,
    productID INT NOT NULL,
    description TEXT,
    price DECIMAL(7,2) NOT NULL,
    itemCondition varchar(1) NOT NULL,
    sellerID INT NOT NULL,
    itemStatus varchar(1) NOT NULL,
    transacMethod varchar(1) NOT NULL,
    region varchar(50),
    postLanguage varchar(10),
    postedDate DATE NOT NULL,
    PRIMARY KEY (postID),
    FOREIGN KEY (productID) References Product(productID),
    FOREIGN KEY (sellerID) References Student_Info(studentID),
    FOREIGN KEY (region) References Location(region)
);

CREATE TABLE Favourite (
```

```
    studentID INT NOT NULL,
    postID INT NOT NULL,
    addDate DATE NOT NULL,
    PRIMARY KEY (studentID, postID),
    FOREIGN KEY (studentID) REFERENCES Student_Info(studentID),
    FOREIGN KEY (postID) REFERENCES Post(postID)
);

CREATE TABLE Order_Info(
    orderID INT NOT NULL,
    postID INT NOT NULL,
    buyerID INT NOT NULL,
    sellerID INT NOT NULL,
    finalPrice DECIMAL(7,2) NOT NULL,
    dealDate Date NOT NULL,
    paymentMethod varchar(1) NOT NULL,
    PRIMARY KEY (orderID),
    FOREIGN KEY (postID) References Post(postID),
    FOREIGN KEY (buyerID) References Student_Info(studentID),
    FOREIGN KEY (sellerID) References Student_Info(studentID)
);

CREATE TABLE Review(
    reviewID INT NOT NULL,
```

```
    toStudent INT NOT NULL,
    orderID INT NOT NULL,
    rating DECIMAL(2,1) NOT NULL,
    review TEXT NOT NULL,
    fromStudent INT NOT NULL,
    reviewDate Date NOT NULL,
    PRIMARY KEY (reviewID),
    FOREIGN KEY (toStudent) References Student_Info(studentID),
    FOREIGN KEY (fromStudent) References Student_Info(studentID),
    FOREIGN KEY (orderID) References Order_Info(orderID)
);
```

SQL Scripts for Inserting Data

INSERT INTO Student_Info VALUES

(260000001, ‘Harry Potter’, ‘Education’, ‘harry.potter@mail.mcgill.ca’, ‘514-111-1111’, ‘t’, ‘english’),

(260000002, ‘Hermione Granger’, ‘Law’, ‘hermione.granger@mail.mcgill.ca’, ‘514-111-1112’, ‘c’, ‘english’),

(260000003, ‘Ron Weasley’, ‘Science’, ‘ronald.weasley@mail.mcgill.ca’, ‘514-111-1113’, ‘c’, ‘english’),

(260000004, ‘Draco Malfoy’, ‘Management’, ‘draco.malfoy@mail.mcgill.ca’, ‘514-111-1114’, ‘t’, ‘english’),

(261000001, ‘Luna Lovegood’, ‘Science’, ‘luna.lovegood@mail.mcgill.ca’, ‘438-111-1111’, ‘c’, ‘english’),

(261000002, ‘Cedric Diggory’, ‘Education’, ‘cedric.diggory@mail.mcgill.ca’, ‘438-111-1112’, ‘t’, ‘english’),

(261000003, ‘Fleur Delacour’, ‘Arts’, ‘fleur.delacour@mail.mcgill.ca’, ‘438-111-1113’, ‘c’, ‘french’);

INSERT INTO Product VALUES

(101, ‘furniture’, ‘Ikea’, 512),

(102,’clothing’, ‘Utopia Home’, 99),

(103, ‘home appliance’, ‘Nespresso’, 229),

(104, ‘home appliance’, ‘Bounce’, 13.99),

(105, ‘outdoor’, ‘Intex Explorer ‘, 120),

(106, ‘jewelry’, ‘LILIE&WHITE’, 59.99),

(107, ‘pets’, ‘Kissbark’, 21),

(108, ‘home appliance’, ‘Ikea’, 125.99),

(109, ‘clothing’, ‘Nike’, 200),

(110, ‘clothing’, ‘Aritzia’, 50);

INSERT INTO Location VALUES

(‘Laval’, 45.575480, -73.751221),

(‘Verdun’, 45.446990, -73.568659),

(‘Plateau’, 45.523939, -73.582865),

(‘Downtown’, 45.503480, -73.568489),

(‘Quartier Latin’, 45.515508, -73.563906),

(‘Macdonald Campus’, 45.4078, -73.9388);

INSERT INTO Post VALUES

(100001, ‘Height adjustable desk’, 101, ‘Digital Display Handset – 4 memory preset options for easy adjustment. Electric Lift System – Fully motorized lift from 28 to 45 Inches height.’, 299.87, ‘l’, 260000001, ‘s’, ‘t’, ‘Downtown’, ‘english’, ‘2021-04-22’),

(100002, ‘Garment Rack Freestanding Hanger’, 102, ‘Large storage space : Double-rods design can accommodate double the number of clothes to meet your life needs. The bottom shelf can also hold multiple pairs of shoes.’,39.99, ‘b’, 260000003, ‘s’, ‘c’, ‘Downtown’, ‘english’, ‘2021-12-02’),

(100003, ‘Dosettes de I colombien dorigine unique Starbucks By Nespresso’, 103, ‘Gouts inspires : ce I a torrefaction oyenne a une intensite de niveau de profil de 7 avec des notes equilibrees et de noisette et est fabrique avec des grains de I 100 colombiens.’,40.55, ‘f’, 261000002, ‘s’, ‘t’, ‘Laval’, ‘french’, ‘2022-04-11’),

(100004, ‘Bounce Fabric Softener Dryer Sheets, Outdoor Fresh, 200 Count’, 104, ‘Keep your laundry extra soft and static-free with Outdoor Fresh Bounce Dryer Sheets’,

7.99, ‘b’, 261000002, ‘s’, ‘c’, ‘Downtown’, ‘english’, ‘2022-06-07’),

(100005, ‘Ensemble de kayak gonflable Intex Explorer K2 avec rames’,105, ‘TRANSPORT FACILE : Leger et compact, ce kayak est facile a monter et, grace a la valve Boston’, 132.99, ‘u’, 261000003, ‘s’, ‘c’, ‘Verdun’, ‘french’, ‘2022-03-03’),

(100006, ‘LILIE&WHITE Chunky Gold Hoop Earrings for Women Cute Fashion Hypoallergenic earrings’, 106, ‘Gold Hoop earrings size : Diameter: 34 in 19mm. This gold earring is made with stainless steel needle to prevent allergy.’, 16.99, ‘f’, 261000001, ‘s’, ‘t’, ‘Plateau’, ‘english’, ‘2022-01-17’),

(100007, ‘Cats Dogs ID Tags Personalized Lovely Symbols Pets Collar Name Accessories Simple Custom’, 107, ‘Brass is copper real, shiny brass use hand-polished and 14K gold surface plated.’, 8.99, ‘l’, 261000001, ‘s’, ‘c’, ‘Quartier Latin’, ‘english’, ‘2021-11-12’),

(100008, ‘Air Jordan 1’, 109, ‘DM FOR MORE INFORMATION AND SIZES’, 190, ‘u’, 261000002, ‘s’, ‘t’, ‘Downtown’, ‘english’, ‘2022-08-25’),

(100009, ‘AIR JORDAN 12 RETRO ROYALITY TAXI Size 7 youth’, 109, ‘The Air Jordan 12 Retro Royalty brings a luxe look to its classic basketball construction. The shoes upper is built with tumbled leather, finished primarily in white and contrasted by black on the basketball-textured mudguard. The classic radiating stitching remains intact, while the upper eyelets, TPU insert and branding add metallic gold to the look.’,160, ‘l’, 260000003, ‘s’, ‘t’, ‘Verdun’, ‘english’, ‘2021-12-10’),

(100010, ‘Aritzia TNA long sleeves tshirt’, 110, ‘Tna Ribbed Cropped Longsleeve Size Small Color Cosmo Pink’, 20, ‘f’, 260000002, ‘a’, ‘c’, ‘Plateau’, ‘english’, ‘2022-09-01’);

INSERT INTO Order_Info VALUES

(3678, 100001, 260000002, 260000001, 280, ‘2021-05-01’, ‘t’),

(3679, 100002, 260000004, 260000003, 39.99, ‘2021-12-03’, ‘c’),

(3680, 100006, 261000002, 261000001, 10, ‘2022-03-31’, ‘c’),

(3681, 100005, 260000002, 261000003, 130, ‘2022-04-01’, ‘c’),

(3682, 100007, 261000003, 261000001, 5, ‘2021-11-15’, ‘t’),

(3683, 100004, 261000003, 261000002, 7.99, ‘2022-08-03’, ‘c’),

(3684, 100003, 260000004, 261000002, 100, ‘2022-05-02’, ‘c’),

(3685, 100008, 260000004, 261000002, 185, ‘2022-08-26’, ‘t’),

(3686, 100009, 261000003, 260000003, 160, ‘2021-12-12’, ‘t’);

INSERT INTO Review VALUES

(123, 260000001, 3678, 4, ‘The desk is nice but a bit damaged’, 260000002, ‘2021-05-01’),

(234, 260000003, 3679, 2, ‘Parts are missing, fortunately it still holds’, 260000004, ‘2021-12-14’),

(345, 261000001, 3680, 5, ‘Perfect’, 261000002, ‘2022-01-18’),

(456, 261000003, 3681, 4, ‘Don’t look that used’, 260000002, ‘2022-03-18’),

(678, 260000002, 3678, 5, ‘Great buyer!’, 260000001, ‘2021-05-01’),

(789, 260000002, 3681, 5, ‘Good’, 261000003, ‘2022-03-31’),

(910, 261000002, 3680, 3, ‘Friendly but reply late’, 261000001, ‘2022-01-19’),

(112, 260000001, 3682, 4, ‘A pleasant experience’, 261000003, ‘2021-11-16’),

(121, 261000003, 3682, 1, ‘Description is MISLEADING’, 260000001, ‘2021-11-18’),

(256, 261000002, 3683, 3, ‘Bad, not functioning at all’, 261000003, ‘2021-12-12’),

(247, 261000002, 3684, 4, ‘Good! I love this machine’, 260000004, ‘2022-05-10’),

(248, 261000002, 3685, 4, ‘Pretty good seller, my second time buying his item’, 260000004, ‘2022-09-01’);

INSERT INTO Favourite VALUES

(260000001, 100003, ‘2022-04-13’),

(260000003, 100003, ‘2021-04-14’),

(261000001, 100004, ‘2022-06-16’),

(261000002, 100006, ‘2022-01-18’),

(260000004, 100005, ‘2021-03-10’),

(260000002, 100005, ‘2021-03-09’),

(261000001, 100003, ‘2022-04-11’),

(261000002, 100004, ‘2022-06-08’);

Sample Queries & SQL Scripts

Categorizations:

1. Features: #1 - #5
2. Client: #6 - #9
3. Decision: #10 - #14
4. Retail: #15 - #22

1. Features

#1 Simple:

McGill Marketplace pushes posts with the highest discount to the home page. Please order all posts by discount percentage.

```
SELECT a.postID, b.originalPrice, a.price as postedPrice,
(b.originalPrice-a.price)/b.originalPrice*100 as discount

FROM Post a, Product b

WHERE a.productID = b.productID

ORDER BY discount DESC
```

#2 Simple:

The ranking page of the website shows popular recently added posts according to the times they got added to the Favourite. Please list the top 3 popular posts posted during the previous year (365 days).

```
SELECT p.postID, COUNT(DISTINCT f.studentID) as popularity, p.postedDate

FROM Post p, Favourite f

WHERE p.postID = f.postID AND p.postedDate > SUBDATE(NOW(), INTERVAL 365 DAY)

GROUP BY p.postID
```

```
ORDER BY popularity DESC

LIMIT 3
```

#3 Complex:

The ranking section of the website shows the best users according to the H-index.

The Use of H-index: Students who have posted or bought many items will have lots of reviews, but the ratings are not necessarily high; Students with high average ratings might just be because they only sold or bought one or two items. Thus, it is less reasonable to identify the best users by counting their transactions or calculating their average ratings. Thus, McGill Marketplace wants to use the H-index to rank the best users. H-index equals the number, H, of reviews for this student which have a rating score of at least H.

H-index is calculated by following these steps:

1. For each student, find their ratings and the number of times they got at least that rating.

2. If Ron got at least 4.0 3 times and at least 3.0 4 times, then his H-index should be 3.

3. If Hermione got at least 4.0 4 times and at least 3.0 4 times, her H-index should be 4.

```
SELECT studentID, name, MAX(min_score) AS `H-index`

FROM

(

SELECT temp_a.studentID, temp_a.name, COUNT(temp_a.rating) as numHigherRating, temp_b.rating,

CASE WHEN COUNT(temp_a.rating) < temp_b.rating THEN COUNT(temp_a.rating) ELSE temp_b.rating END AS min_score

FROM(

SELECT s.studentID, s.name, r.rating, COUNT(r.orderID) AS num

FROM Student_Info s, Review r

WHERE s.studentID=r.toStudent
```

```
GROUP BY s.studentID, r.rating

) temp_a

,

Review temp_b

WHERE temp_a.studentID = temp_b.toStudent AND temp_a.rating >= temp_b.rating

GROUP BY temp_a.studentID, temp_b.rating

) temp_c

GROUP BY studentID

ORDER BY `H-index` DESC
```

#4 Simple:

Recommended posts under a post are those that are selling products under the same category. For example, for a post on home appliances, the recommended posts under this post will also be in the home appliance category.

```
SELECT a.postID, a.title

FROM Post a, Product b

WHERE a.productID = b.productID AND b.category = 'home appliance'
```

#5 Simple:

One functionality of McGill Marketplace is giving sellers suggestions on the price according to the conditions and categories of their items so that sellers know what price they should set in order to sell their items quickly.

```
SELECT b.category, a.itemCondition, AVG(price)

FROM Post a, Product b

WHERE a.productID = b.productID
```

```
GROUP BY b.category, a.itemCondition
```

2. Client

#6 Simple:

Jessica is interested in knowing the average final price for each category of products. Then she can compare them with another platform to decide whether she wants to switch to McGill MarketPlace as her primary trading method.

```
SELECT Product.category, AVG(Order_Info.finalPrice) AS averagePrice
FROM Product, Post, Order_Info
WHERE Product.productID=Post.productID AND
        Order_Info.postID=Post.postID
GROUP BY Product.category
```

#7 Simple:

James is in the Education program and wants to search for the posts posted by other students also in the Education department.

```
SELECT Student_Info.studentID, name, faculty, Post.postID, description, price
FROM Student_Info, Post
WHERE Student_Info.studentID = Post.sellerID
AND Student_Info.faculty = 'Education'
```

#8 Simple:

Jack wants to find out the difference between the posted price and the dealt price so that he can better know how to set the price in his post to maximize his earnings.

```
SELECT Student_Info.studentID, name, T.averageRating
FROM Student_Info,
(SELECT toStudent, AVG(rating) AS averageRating
FROM Review
GROUP BY toStudent
HAVING AVG(rating) >=4) AS T
WHERE Student_Info.studentID=T.toStudent
```

#9 Simple:

Jenny is renting an apartment and would like to live in an area where relatively more people are selling second-hand furniture. Please list the top area where most furniture has been or is being sold.

```
SELECT region, COUNT(po.productID) AS furniturePosts
FROM Post po, Product pro
WHERE po.productID = pro.productID AND pro.category = 'furniture'
GROUP BY region
ORDER BY furniturePosts
LIMIT 1
```

3. Decision

#10 Simple:

McGill Marketplace is thinking about hosting an in-person garage sale next year. Please help us decide which month McGill should host this event on campus. (Filter out all on-hold/sold orders located in the downtown area, then find out the months when most orders were dealt.

```
SELECT MONTH(dealDate) as dealMonth, COUNT(orderID) as orders

FROM Order_Info o, Post p

WHERE o.postID = p.postID AND p.region = 'Downtown'

GROUP BY dealMonth

ORDER BY orders DESC
```

#11 Simple:

McGill is considering using student accounts as a supported payment method of Marketplace. If more students use cash rather than online transfer, McGill will have the chance to promote student accounts. Please compare the number of students using cash and online transfer as their primary payment methods.

```
SELECT primaryPaymentMethod, COUNT(studentID)

FROM Student_Info

GROUP BY primaryPaymentMethod
```

#12 Simple:

Since McGill is a bilingual university, McGill Marketplace is mixed with English and French posts. McGill is thinking about embedding an auto translator to the McGill Marketplace if more than 60% of posts are bought by students whose preferred language is the same as the post language. This will prove that students tend to purchase posts with the same language as their preferred ones. Please find out if this is necessary.

```
SELECT

(SELECT COUNT(*)

FROM Order_Info LEFT JOIN Student_Info ON Order_Info.buyerID = Student_Info.studentID

LEFT JOIN Post ON Order_Info.postID = Post.postID

WHERE Student_Info.primaryLanguage = Post.postLanguage
```

```
)/COUNT(*) AS proportion

FROM Order_Info
```

#13 Simple:

The Faculty of Management at McGill is thinking about improving the learning space for Desautels students. In order to target the correct point, the staff would like to find out what categories of goods Desautels students are trading on McGill Marketplace. Please list out the top 3 products that Desautels students ordered the most.

```
SELECT pro.productID, pro.category, pro.brand, COUNT(o.orderID) AS numOrdered

FROM Order_Info o LEFT JOIN Student_Info s ON o.buyerID = s.studentID

LEFT JOIN Post po ON o.postID = po.postID

LEFT JOIN Product pro ON po.productID = pro.productID

WHERE s.faculty = 'Management'

GROUP BY pro.productID

ORDER BY numOrdered

LIMIT 3
```

#14 Complex:

Add a column “Popularity” to Post and set the default value as 0 (“not popular)

After that, replace their Popularity with the corresponding number of Favourite that the Post has.

Let’s suppose the development department of McGill Marketplace wants to free up the storage space of the database. They are to delete three posts based on both their length of posting period (LPP) and popularity.

*You first need to compute the *Adjusted LPP as well as Popularity of each post. After that, you can decide whether to delete the post by computing (Popularity - Adjusted LPP). This formula*

assumes that Popularity has positive effects, whereas LPP has negative ones. Therefore, posts that have low value should be eliminated.

**Adjusted LPP: the number of days between the postedDate and '2022-08-30' divided by 100*

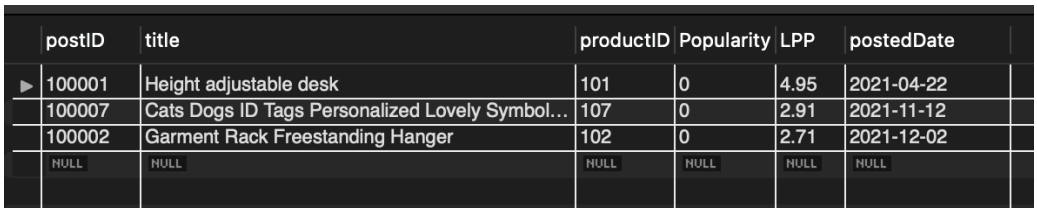
Display the postID, title, productID, popularity, LPP, and postedDate of all the posts to be deleted.

```
ALTER TABLE Post

ADD LPP DECIMAL(4,2) Default 0;


ALTER TABLE Post

ADD Popularity int DEFAULT 0;


CREATE VIEW cnt_pop AS

SELECT p.postID, COUNT(p.postID) cnt

FROM Post p

JOIN Favourite f

ON p.postID = f.postID

GROUP BY p.postID;


UPDATE Post

SET Popularity = (SELECT cnt FROM cnt_pop WHERE cnt_pop.postID = Post.postID)

WHERE postID IN (SELECT postID FROM cnt_pop);


UPDATE Post

SET LPP = DATEDIFF('2022-08-30', Post.postedDate)/100;
```

```
SELECT postID, title, productID, Popularity, LPP, postedDate

FROM Post

ORDER BY (Popularity - LPP)

LIMIT 3;
```

Output:

(Since Infinityfree.net does not support functions like CREATE VIEW and UPDATE, we attached the outcome implemented by MySQL Workbench)



postID	title	productID	Popularity	LPP	postedDate
100001	Height adjustable desk	101	0	4.95	2021-04-22
100007	Cats Dogs ID Tags Personalized Lovely Symbol...	107	0	2.91	2021-11-12
100002	Garment Rack Freestanding Hanger	102	0	2.71	2021-12-02
NULL	NULL	NULL	NULL	NULL	NULL

4. Retail

#15 Simple:

McGill Marketplace wants to assess the usefulness of the new functionality, ADD TO FAVORITE. Find students who eventually purchased the product in the posts that they had added to their favorite.

```
SELECT T.buyerID, Student_Info.name, T.postID

FROM Student_Info,

(SELECT Order_Info.buyerID, dealDate, Order_Info.postID, Favourite.addDate

FROM Order_Info, Favourite

WHERE Order_Info.buyerID = Favourite.studentID AND

Order_Info.postID = Favourite.postID) AS T
```

```
WHERE T.buyerID = Student_Info.studentID
```

#16 Simple:

To monitor the overall performance of McGill Marketplace, we would like to find the total amount of transactions made each year. A decrease in this value will alert us to make some improvements to the website's functionality.

```
SELECT YEAR(dealDate) AS `year`, SUM(finalPrice) as totalTransac
FROM Order_Info
GROUP BY `year`
ORDER BY `year` ASC
```

#17 Simple:

McGill Marketplace wants to know the number of new users joining each month starting from 2022. A user is defined as a new user as long as he/she has posted or ordered something using McGill Marketplace.

```
SELECT MONTH(joinDate) AS `month`, COUNT(studentID) as newUsers
FROM(
SELECT a.studentID,
CASE WHEN postedDate <= dealDate THEN postedDate ELSE dealDate END AS joinDate
FROM(
SELECT s.studentID, s.name, MIN(p.postedDate) AS postedDate, MIN(o.dealDate) AS
dealDate
FROM Student_Info s LEFT JOIN Post p ON s.studentID = p.sellerID
LEFT JOIN Order_Info o ON s.studentID = o.buyerID
GROUP BY s.studentID
) a
```

```
) b

WHERE YEAR(joinDate) = 2022

GROUP BY `month`

ORDER BY `month` ASC
```

#18 Complex:

McGill Marketplace plans to create a WordCloud to show the frequency of words in product reviews. As a part of this plan, they want to know if each review contains positive words such as "Perfect", “Great”, “Good”, or negative words like “Damaged” and “Bad”. Let’s suppose positive words are stored in a set P and a set N is for negative ones.

P = “Perfect”, “Great”, “Good”

N = “Damaged”, “Bad”

If a certain review has at least one word in P, we label it as “positive”. On the other hand, if at least one word is in N, we save it as “negative.” Otherwise, the value “none” will be assigned. One of these three values will be displayed as a result, and it will be included in a column called “Sentiment_label.”

Write SQL codes to display reviewID, rating, and the sentiment label for each review.

```
SELECT reviewID, rating,

(CASE

   WHEN review LIKE '%perfect%' THEN 'positive'

   WHEN review LIKE '%good%' THEN 'positive'

   WHEN review LIKE '%great%' THEN 'positive'

   WHEN review LIKE '%bad%' THEN 'negative'

   WHEN review LIKE '%damaged%' THEN 'negative'

   ELSE 'none'

END) AS 'Sentiment_label'
```

```
FROM Review;
```

#19 Complex:

Let's say we want to label buyers "a" (active), "i" (inactive), or "n" (none) based on their changes in payments (i.e., activity index) between the first and last records.

For example, if buyer A's expenditure has increased from $10 to $100, A is labeled as "a." However, if that of B's has decreased from $100 to $10, we consider B an inactive customer and label B as "i." Otherwise (i.e., if the difference equals 0 or a user doesn't have enough transaction records), we label the user "n."

Among students who have made at least two payments, display studentID, name, an average amount of payments, and activity index.

```
ALTER TABLE Student_Info

ADD Activity_index CHAR(1) Default 'n';


CREATE VIEW Student_Ordered AS

SELECT *

FROM Student_Info s

JOIN Order_Info o

ON s.studentID = o.buyerID;


CREATE VIEW tmp AS

SELECT studentID

FROM Student_Ordered

GROUP BY studentID

HAVING COUNT(orderID) > 1;
```

```
CREATE VIEW First_payment AS
SELECT finalPrice FROM Student_Ordered ORDER BY dealDate ASC LIMIT 1;

CREATE VIEW Last_payment AS
SELECT finalPrice FROM Student_Ordered ORDER BY dealDate DESC LIMIT 1;

UPDATE Student_Info
SET Activity_index = (
CASE
        WHEN studentID = (SELECT * FROM tmp) and ((SELECT * FROM Last_payment) > (SELECT * FROM First_payment)) THEN 'a'
   WHEN studentID = (SELECT * FROM tmp) and ((SELECT * FROM Last_payment) < (SELECT * FROM First_payment)) THEN 'i'
   ELSE 'n'
   END
);

SELECT studentID, name,
(SELECT AVG(finalPrice) Avg FROM Student_Ordered GROUP BY studentID HAVING studentID = Student_Info.studentID) as Avg_exp,
Activity_index
FROM Student_Info;
```

Output:

(Since Infinityfree.net does not support functions like CREATE VIEW and UPDATE, we attached the outcome implemented by MySQL Workbench)

(In this case, NULL values can exist in the Avg_exp (Average Expenditure) column. It is because there could be someone who has not purchased yet)

studentID	name	Avg_exp	Activity_index
260000001	Harry Potter	142.500000	i
260000002	Hermione Granger	NULL	n
260000003	Ron Weasley	39.990000	n
260000004	Draco Malfoy	NULL	n
261000001	Luna Lovegood	10.000000	n
261000002	Cedric Diggory	7.990000	n
261000003	Fleur Delacour	130.000000	n

#20 Simple:

The school of retailing would like to analyze the most easy-to-sell second-hand brands among university students, and they want to request some data from McGill Marketplace. For each category of products, please find out the brand with the shortest average number of days it takes to sell the product.

```
SELECT p.category, p.brand, MIN(o.dealDate-po.postedDate) AS days
FROM Order_Info o LEFT JOIN Post po ON o.postID = po.postID
LEFT JOIN Product p ON po.productID = p.productID
GROUP BY p.category, p.brand
```

#21 Simple:

The Retailing School is researching the relationship between sell/customer's rating and a second-hand item's usage level. Please calculate the average rating for each item condition: brand new, like new, used, and fair.

```
SELECT p.itemCondition, AVG(r.rating) AS avg_rating
FROM Review r LEFT JOIN Order_Info o ON r.orderID = o.orderID
```

```
LEFT JOIN Post p ON o.postID = p.postID

GROUP BY p.itemCondition
```

#22 Complex:

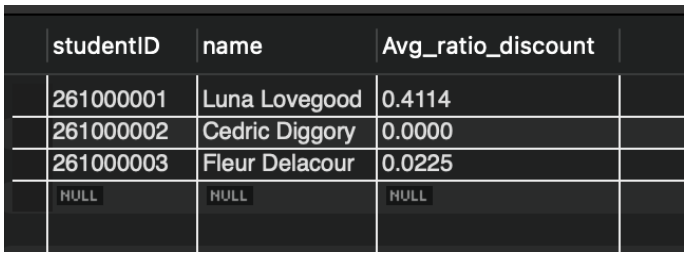
Write SQL codes to list student ID, name, and an average ratio of the discount they got as buyers who made any purchases from Jan 01, 2022 (inclusive). If a user doesn't have transaction history, display NULL.

```
CREATE VIEW s_and_o AS

SELECT s.studentID, s.name, o.dealDate, o.finalPrice, o.postID

FROM Student_Info s

JOIN Order_Info o

ON o.buyerID = s.studentID;


CREATE VIEW avg_discount AS

SELECT s_and_o.studentID, AVG((p.price - s_and_o.finalPrice)/p.price) avg

FROM (SELECT s.studentID, s.name, o.dealDate, o.finalPrice, o.postID FROM Student_Info s
JOIN Order_Info o ON o.buyerID = s.studentID) AS s_and_o

JOIN Post p

ON s_and_o.postID = p.postID

WHERE s_and_o.dealDate >= '2022-01-01'

GROUP BY s_and_o.studentID;


ALTER TABLE Student_Info

ADD Avg DECIMAL(5,4) Default -1;
```

```
UPDATE Student_Info

SET Avg = (SELECT avg FROM avg_discount WHERE avg_discount.studentID =
Student_Info.studentID)

WHERE Student_Info.studentID IN (SELECT studentID FROM (SELECT s_and_o.studentID,
AVG((p.price - s_and_o.finalPrice)/p.price) avg

FROM s_and_o

JOIN Post p

ON s_and_o.postID = p.postID

WHERE s_and_o.dealDate >= '2022-01-01'

GROUP BY s_and_o.studentID) AS avg_discount);


SELECT studentID, name, Avg as Avg_ratio_discount

FROM Student_Info s

WHERE s.Avg != -1;
```

Output:

(Since Infinityfree.net does not support functions like CREATE VIEW and UPDATE, we attached the outcome implemented by MySQL Workbench)



studentID	name	Avg_ratio_discount
261000001	Luna Lovegood	0.4114
261000002	Cedric Diggory	0.0000
261000003	Fleur Delacour	0.0225
NULL	NULL	NULL

Stage 3

Webpage

Mcgillmarketplace.epizy.com