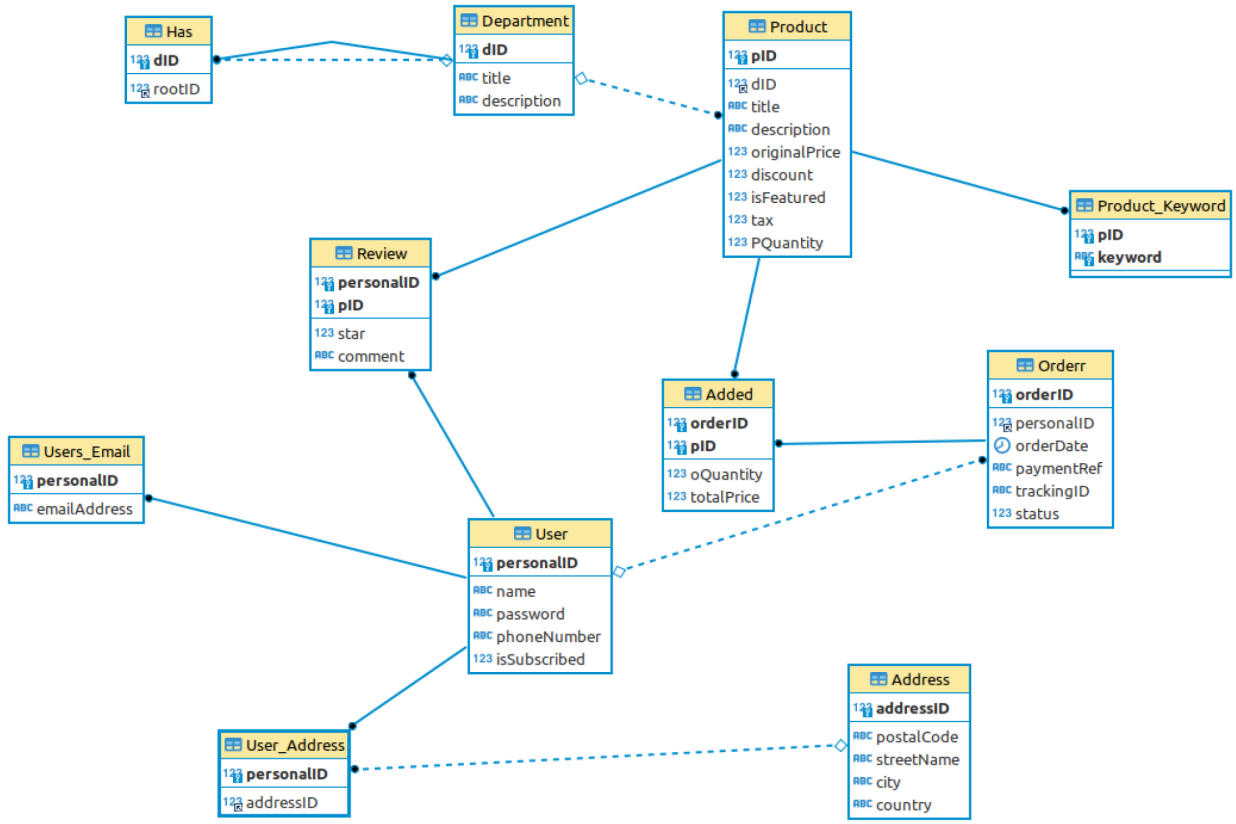
Group3

Jinglin Gao	gaojinglin449@gmail.com
Prashanna Rai	prai931024@gmail.com
Tse-An Hsu	jasonhsutsean@gmail.com

milestone 1,2 are submitted.

Link to source file, milestone 1 and milestone 2.

E. The diagram generated by MySQL Workbench's Reverse Engineer



The notation used in this diagram is IDEF1X

3. Write the SQL commands to create the tables.

We upload a single sql script named milestone.sql. This script can be executed on an empty database only.
This script will create all the tables, populated rows into tables using insert script and projection to find some answer to the question. This single script contains all script for
Source File: milestone3.sql

```
create table User
(
	personalID int,
	name varchar(20),
	password varchar(20),
	phoneNumber varchar(10),
	isSubscribed boolean DEFAULT 0,

	primary key (personalID)
) ENGINE=INNODB;

create table Users_Email
(
	personalID int,
	emailAddress varchar(70),

	primary key (personalID),
	foreign key (personalID) references User(personalID)
		on delete cascade
		on update cascade
)ENGINE=INNODB;

create table Address
(
	addressID int AUTO_INCREMENT,
	postalCode varchar(6),
	streetName varchar(30),
	city varchar(30),
	country varchar(3),

	primary key (addressID)
)ENGINE=INNODB;

create table User_Address
(
	personalID int,
```

```
	addressID int,

	primary key (personalID),
	foreign key (personalID) references User(personalID)
		on delete cascade
		on update cascade,
	foreign key (addressID) references Address(addressID)
		on delete SET NULL
		on update cascade
)ENGINE=INNODB;

create table `Order`
(
	orderID int,
	personalID int,
	orderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
	paymentRef varchar(64),
	trackingID varchar(10),
	status int,

	primary key (orderID),
	foreign key (personalID) references User(personalID)
		on delete cascade
		on update cascade
)ENGINE=INNODB;

create table Department
(
	dID int AUTO_INCREMENT,
	title varchar(50),
	description TEXT,

	primary key (DId)
)ENGINE=INNODB;

create table Has
(
	rootID int,
	dID int,

	primary key (dID),
	foreign key (rootID) references Department(dID)
		on delete SET NULL
```

```
            on update cascade,
      foreign key (dID) references Department(dID)
            on delete cascade
            on update cascade
)ENGINE=INNODB;

create table Product
(
      pID int AUTO_INCREMENT,
      dID int,
      title varchar(50),
      description TEXT,
      originalPrice int,
      discount float,
      isFeatured boolean DEFAULT 0,
      tax float,
      PQuantity int,

      primary key (pID),
      foreign key (dID) references Department(dID)
            on delete SET NULL
            on update cascade
)ENGINE=INNODB;

create table Product_Keyword
(
      pID int,
      keyword varchar(30) NOT NULL,

      primary key (pID, keyword),
      foreign key (pID) references Product(pID)
            on delete cascade
            on update cascade
)ENGINE=INNODB;

create table Review
(
      personalID int,
      pID int,
      star int,
      comment TEXT,

      primary key(personalID, pID) ,
```

```
	foreign key (personalID) references User(personalID)
		on delete cascade
		on update cascade,
	foreign key (pID) references Product(pID)
		on delete cascade
		on update cascade
)ENGINE=INNODB;

create table Added
(
	orderID int,
	pID int,
	oQuantity int,
	totalPrice int,

	primary key(orderID, pID),
	foreign key (orderID) references `Order`(orderID)
		on delete cascade
		on update cascade,
	foreign key (pID) references Product(pID)
		on delete cascade
		on update cascade
)ENGINE=INNODB;
```

4. Create at least 2 top-level departments, each having at least 3 child departments, and at least 10 products in the store. Make some of the products featured on the homepage. Create at least 2 users and add their reviews for the same product. Create one order for one of the users.
Source File: milestone3-task4.sql

(1) Create at least 2 top-level departments, each having at least 3 child departments, and at least 10 products in the store. Make some of the products featured on the homepage.

```
	Insert Department (dID, title, description) values (2, 'Eletronics',
'Electronic');
	Insert Department (dID, title, description) values (13, 'Books',
'Books');
```

Let's say the department id of the 3 child departments for Books are 14, 15, 16.
Same execution but different dID for Electronics.

```
Insert Has (rootID, dID) values (13, 14);
Insert Has (rootID, dID) values (13, 15);
Insert Has (rootID, dID) values (13, 16);
```

For products, let's take Books as example, also create some featured products by making isFeatured to 1.

```
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B1', 'book', 3434, 1, 1, 0.4, 1);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B2', 'book', 1564, 1, 0, 0, 2);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B3', 'book', 2466, 1, 0, 0.4, 20);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B4', 'book', 6671, 1, 0, 0, 16);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B5', 'book', 3967, 1, 0, 0, 1);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B6', 'book', 1835, 0.9, 0, 0, 15);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B7', 'book', 5003, 0.9, 0, 0, 16);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B8', 'book', 4661, 1, 0, 0.3, 9);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B9', 'book', 1982, 0.7, 0, 0.22,
9);
insert Product (dID, title, description, originalPrice, discount,
isFeatured, tax, PQuantity) values (13, 'B10', 'book', 3672, 0.6, 0, 0.2,
11);
```

(2)Create at least 2 users and add their reviews for the same product. Create one order for one of the users.

Users:

```
insert into User (personalID, name, password, isSubscribed,
phoneNumber) values ('72874248', 'RExihW_79', '0f30TCEimND', true,
'4918314759');
insert into User (personalID, name, password, isSubscribed,
phoneNumber) values ('72966582', 'LDslqO_42', '5c76GJIvtS3', true,
'2792994665');
```

Reviews for the same product:

```
insert Review (personalID, pID, star, comment) values (72966582, 2, 2, 'could be better');
insert Review (personalID, pID, star, comment) values (72874248, 2, 2, 'could be better');
```

Create one order for User 72874248:

```
insert `Order` (orderID, personalID, orderDate, paymentRef, trackingID, status) values (79, 72874248, '2022-08-31 04:51:39', '38rny344252tvwzd40cu4u3nz1cn6a2rr040ph1v27rfgudwznyesooxubak3pvw', '79_tracking', 1);
```

5. Write SQL queries to get:
Source File: milestone3-task5.sql
• Welcome text for the homepage

```
SELECT description from Department where dID IN (SELECT dID FROM Has where rootID is NULL);
```

• List of the top level departments with fields needed for the homepage

```
SELECT title ,description  from Department where dID IN (SELECT dID from Has where rootID IN (SELECT dID FROM Has where rootID is NULL));
```

• List of the featured products with fields needed for the homepage

```
SELECT title,originalPrice,discount ,tax,PQuantity  FROM Product as p where
isFeatured =1;
```

• Given a product, list all keyword-related products
When product ID is provided,

```
SELECT pk.keyword from Product_Keyword as pk WHERE pk.pID =36;
```

• Given an department, list of all its products (title, short description, current retail price) with their average rating

Let's say departmentID is provided. The purpose of selecting P.pID is for applying the group by function.

```
select title, description, (originalPrice * discount * (1+tax)) as
retailPrice, avg(star), P.pID
from Product as P left outer join Review as R on P.pID = R.pID
where dID = 5
group by P.pID;
```

• List of all products on sale sorted by the discount percentage (starting with the biggest discount)

The discount attribute will store, for instance 1 for no discount and 0.6 for 40% off.

```
SELECT * FROM Product where discount < 1 order by discount;
```

Optional challenges:

• List 10 best-selling products (in last 30 days)

```
SELECT pID, SUM(oQuantity) AS totalquantity
FROM `Order` natural join Added
WHERE orderDate >= current_date() - interval 30 day
GROUP BY pID
Order by totalquantity DESC
limit 10;
```

• Write an SQL query as complex as possible (but still doing something useful).

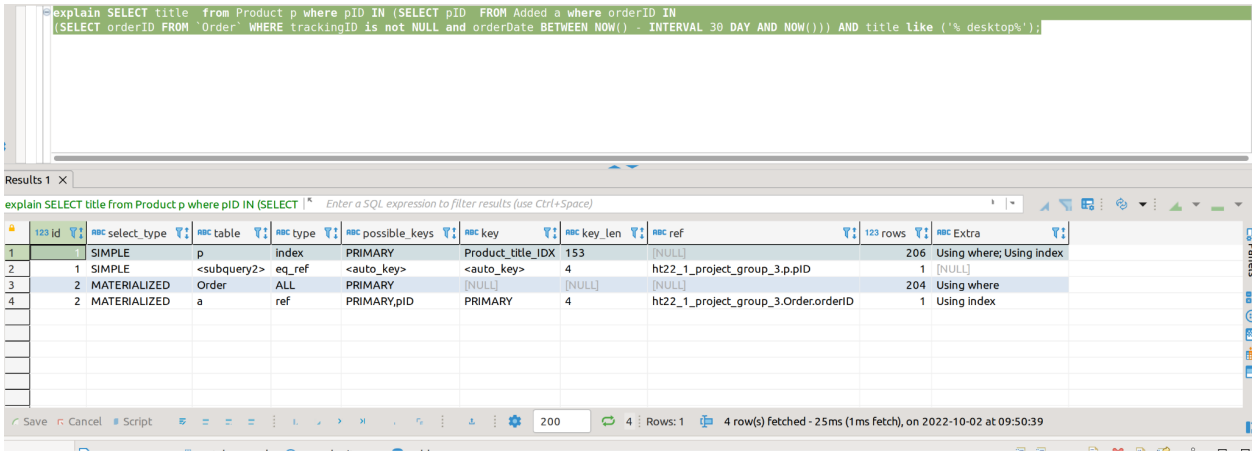
The following SQL script lists all the titles of products whose title contains a naming pattern desktop which are ordered before 30 days from now.

```
SELECT title  from Product p where pID IN (SELECT pID  FROM Added a where
orderID IN
(SELECT orderID FROM `Order` WHERE trackingID is not NULL and orderDate
BETWEEN NOW() - INTERVAL 30 DAY AND NOW())) AND title like ('% desktop%');
```

6. Analyze and optimize the performance of these queries. Identify and choose at least one SQL query which can run faster by adding an index. Write SQL statement(s) for adding this index (these indices). In you report, include the output of the EXPLAIN command before and after adding the index (indices)
as well.

While executing this query on sql server, it took 25 ms but it searched 206 rows to get the result. This is because we are searching based on non-indexed attributes like orderDate.

SELECT title from Product p where pID IN (SELECT pID FROM Added a where orderID IN (SELECT orderID FROM `Order` WHERE trackingID is not NULL and orderDate BETWEEN NOW() - INTERVAL 30 DAY AND NOW())) AND title like ('% desktop%');



So creating an index on orderDate using the following sql script, we noted the time taken was the same but the number of rows scanned were only 66. This means that search space got reduced but because the small table size was approximately 64KB we couldn't observe time improvement by the use of index. It felt that time to fetch 206 rows was the same as time to fetch 66 rows because they tend to lie on the same block.

Source file : milestone3-task6.sql
CREATE INDEX Order_orderDate_IDX USING BTREE ON h `Order` (orderDate);