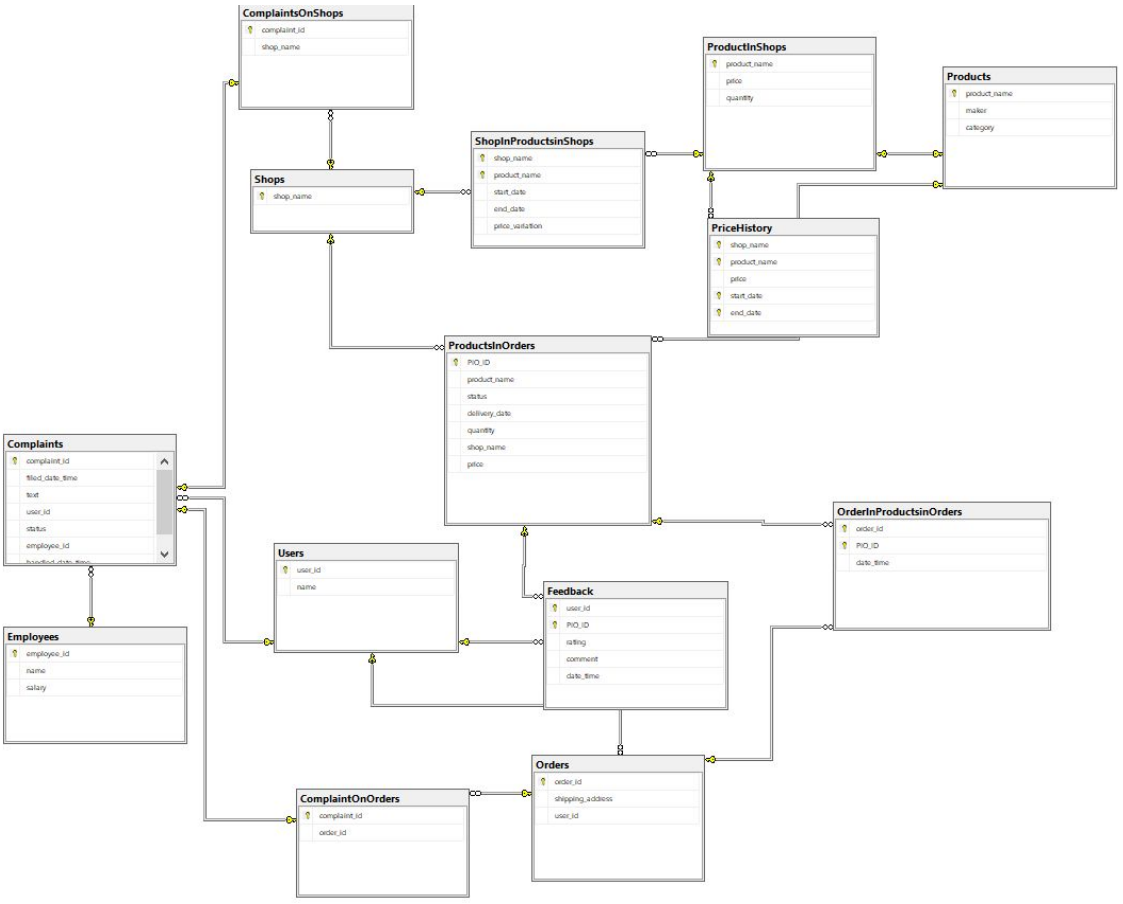

ComplaintsOnShops
complaint_id
shop_name
ProductInShops
product_name
price
quantity
Products
product_name
maker
category
ShopInProductsinShops
shop_name
product_name
start_date
end_date
price_variation
Shops
shop_name
PriceHistory
shop_name
product_name
price
start_date
end_date
ProductsInOrders
PIO_ID
product_name
status
delivery_date
quantity
shop_name
price
Complaints
complaint_id
filled_date_time
text
user_id
status
employee_id
OrderInProductsinOrders
order_id
PIO_ID
date_time
Users
user_id
name
Feedback
user_id
PIO_ID
rating
comment
date_time
Employees
employee_id
name
salary
Orders
order_id
shipping_address
user_id
ComplaintOnOrders
complaint_id
order_id

```
CREATE TABLE Shops(
        shop_name VARCHAR(50) NOT NULL,
        PRIMARY KEY(shop_name)
);
```

| shop_name |
|---|
| Bobs Bottles |
| breadtalk |
| guardian |
| increasingShop1 |
| increasingShop2 |
| iPhone Best |
| iStudio |
| Jshop |
| Logitech Official |
| Phone Shop |
| polar |
| Popular |
| Razer Official |
| Samsung Shop1 |
| Samsung Shop2 |
| Samsung Shop3 |
| sephora |
| watsons |

```
CREATE TABLE Products(
        product_name VARCHAR(50) NOT NULL,
        maker VARCHAR(50) NOT NULL,
        category VARCHAR(50) NOT NULL,
        CHECK(category NOT LIKE '%[^A-Z0-9]%'),
        PRIMARY KEY (product_name)
);
```

| product_name | maker | category |
|---|---|---|
| bagel | idk | food |
| Bottle | Friends | Others |
| iPhone X | Apple | Gadgets |
| iPhone Xs | Apple | Gadgets |
| lipstick | chanel | cosmetics |
| Logitech GPro ... | Logitech | Gadgets |
| power lipstick | sephora | cosmetics |
| Razer Speaker | Razer | Gadgets |
| Samsung earpi... | Samsung | Gadgets |
| Samsung galax... | Samsung | Gadgets |
| Samsung galax... | Samsung | Gadgets |
| stapler | Max Staples | Others |

```
CREATE TABLE Users(
        user_id VARCHAR(50) NOT NULL,
        name VARCHAR(50) NOT NULL,
        PRIMARY KEY(user_id)
);
```

| user_id | name |
|---|---|
| 1 | User_1 |
| 10 | User_10 |
| 100 | User_100 |
| 101 | User_101 |
| 102 | User_102 |
| 103 | User_103 |
| 104 | User_104 |
| 105 | User_105 |
| 106 | User_106 |
| 107 | User_107 |
| 108 | User_108 |
| 109 | User_109 |
| 11 | User_11 |
| 110 | User_110 |
| 111 | User_111 |
| 112 | User_112 |
| 113 | User_113 |
| 114 | User_114 |
| 115 | User_115 |
| 116 | User_116 |

```
CREATE TABLE Orders(
        order_id INT NOT NULL,
        shipping_address VARCHAR(50) NOT NULL,
        user_id VARCHAR(50) NOT NULL,
        PRIMARY KEY(order_id),
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

| order_id | shipping_address | user_id |
|---|---|---|
| 1 | ANG MO KIO | 1 |
| 2 | bishan | 1 |
| 3 | amk | 1 |
| 4 | woodlands | 2 |
| 5 | changi | 1 |
| 55 | changi | 1 |
| 65 | amk | 2 |
| *NULL* | *NULL* | *NULL* |

`

```
CREATE TABLE ProductInShops(
        product_name VARCHAR(50) NOT NULL,
        price FLOAT NOT NULL,
        quantity INT NOT NULL,
        PRIMARY KEY (product_name),
        FOREIGN KEY (product_name) REFERENCES Products(product_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE

);
```

| product_name | price | quantity |
|---|---|---|
| iPhone X | 1500 | 3 |
| iPhone Xs | 1800 | 4 |
| lipstick | 10 | 4 |
| Samsung Earpi... | 200 | 2 |
| Samsung Galax... | 300 | 4 |
| Samsung Galax... | 300 | 3 |

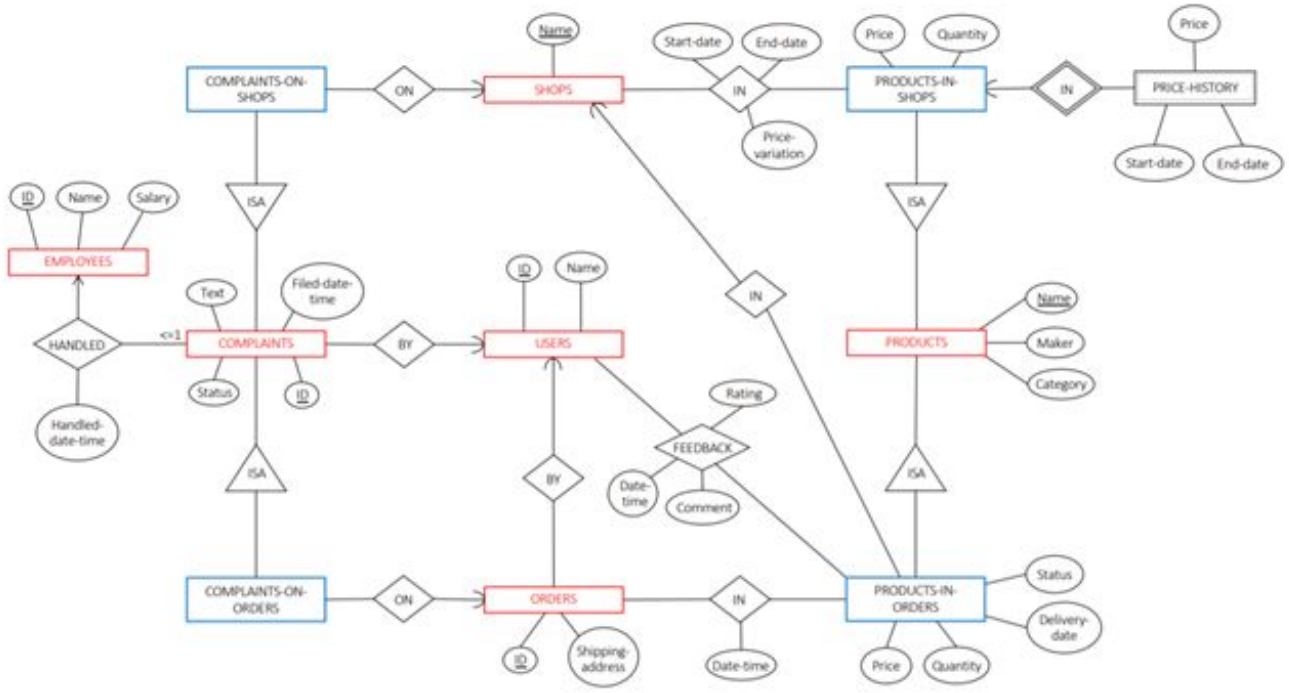
```
CREATE TABLE ShopInProductsinShops(
        shop_name VARCHAR(50) NOT NULL,
        product_name VARCHAR(50) NOT NULL,
        start_date DATE NOT NULL,
        end_date DATE NOT NULL,
        price_variation FLOAT NOT NULL,
        PRIMARY KEY (shop_name, product_name),
        FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (product_name) REFERENCES ProductInShops(product_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE


);

CREATE TABLE ProductsInOrders(
        PIO_ID INT NOT NULL identity(1,1),
        product_name VARCHAR(50) NOT NULL,
        status VARCHAR(20) NOT NULL,
        delivery_date SMALLDATETIME*** NOT NULL,
        quantity INT NOT NULL,
        shop_name VARCHAR(50) NOT NULL,
        price FLOAT NOT NULL,
        PRIMARY KEY (PIO_ID),
        FOREIGN KEY (product_name)  REFERENCES Products(product_name)
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE,
        FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);




CREATE TABLE OrderInProductsinOrders(
        order_id INT NOT NULL,
        PIO_ID INT NOT NULL,
        date_time smalldatetime NOT NULL,
        PRIMARY KEY(order_id, PIO_ID),
        FOREIGN KEY (order_id) REFERENCES Orders(order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
        FOREIGN KEY (PIO_ID) REFERENCES ProductsInOrders(PIO_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Feedback(
        user_id VARCHAR(50) NOT NULL,
        PIO_ID INT NOT NULL,
        rating FLOAT not null,
        comment VARCHAR(100) NOT NULL,
        date_time SMALLDATETIME NOT NULL,
        CHECK(rating>=0 AND rating <= 5),
        PRIMARY KEY(user_id, PIO_ID),
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
        FOREIGN KEY (PIO_ID) REFERENCES ProductsInOrders(PIO_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);




CREATE TABLE Employees(
        employee_id INT NOT NULL,
        name VARCHAR(50) NOT NULL,
        salary FLOAT NOT NULL,
        PRIMARY KEY(employee_id)
```

```
);

CREATE TABLE Complaints(
	complaint_id INT NOT NULL,
	filed_date_time SMALLDATETIME NOT NULL,
	text VARCHAR(200) NOT NULL,
	user_id VARCHAR(50) NOT NULL,
	status VARCHAR(20) NOT NULL,
	employee_id INT NOT NULL,
	handled_date_time SMALLDATETIME NOT NULL,
	CHECK (handled_date_time >= filed_date_time),
	PRIMARY KEY (complaint_id),
	FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
	ON DELETE CASCADE
	ON UPDATE CASCADE,
	FOREIGN KEY (user_id) REFERENCES Users(user_id)
	ON DELETE CASCADE
	ON UPDATE CASCADE

);




CREATE TABLE PriceHistory(
	product_name VARCHAR(50) NOT NULL,
	shop_name VARCHAR(50) NOT NULL,
	price FLOAT NOT NULL,
	start_date SMALLDATETIME NOT NULL,
	end_date SMALLDATETIME NOT NULL,
	PRIMARY KEY (product_name, shop_name, start_date, end_date),
	FOREIGN KEY (product_name) REFERENCES ProductInShops(product_name)
	ON DELETE CASCADE
	ON UPDATE CASCADE
);



CREATE TABLE ComplaintsOnShops(
	complaint_id INT NOT NULL,
	shop_name VARCHAR(50) NOT NULL,
	PRIMARY KEY (complaint_id),
	FOREIGN KEY (complaint_id) REFERENCES Complaints(complaint_id)
	ON UPDATE CASCADE
	ON DELETE CASCADE,
	FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
	ON UPDATE CASCADE
	ON DELETE CASCADE
```

```
);



CREATE TABLE ComplaintOnOrders(
        complaint_id INT NOT NULL,
        order_id INT NOT NULL,
        PRIMARY KEY (complaint_id),
        FOREIGN KEY (complaint_id) REFERENCES Complaints(complaint_id),
        FOREIGN KEY (order_id) REFERENCES Orders(order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Team 3 members:
Khong Farn Ming Nicholas
Toh Jun Jie
Tan Yap Siang
Lee Kai Jie, John
Tan Leng Hwee Gordon
Lee Kai En
Koh Boon Juey




Attributes highlighted in Red, are attributes inherited.

Shops(shop_Name)

PriceHistory( product_Name,Shop_name, start_Date, end_Date, price)

ShopInProductsInShop ( shop_name, product_Name_ start-date, End-date, Price-variation)

ProductInShops( product_name, shop_name,,start-date, End-date, Price, Quantity Price-variation, )

Assumption; each day only one change in price can be made

No bad entries allowed. Eg. if there is an entry with start date 2020-08-01 and end date 2020-08-10 and there will not be another entry with start date before 2020-08-10.

Products(product_Name, maker, category)

~~Productsinorder( product_Name, status, Delivery_date, Quantity, Price, shop_name)~~

ProductsinOrder( PIO_ID, product_Name, status, Delivery_date, Quantity, Price, shop_name)

**Changed primary key to PIO_ID bcuz product_Name cannot be a primary key.**

**IF product_name is a primary key, it will be a unique value and only can have 1 tuple. Hence, a single product_name cannot be sold by multiple shop_names.**

**Note: PIO_ID is also a running number 1,2,3,4…. etc**

OrdersInProductsInOrders (Order_ID, Product_Name, date-time)

Feedback( User-ID, PIO_ID, Rating, Comment, Date-time)

Users( User_ID, Name)

Orders( Order_Id , Shipping_Address, User_ID)

ComplaintsOnOrders (Complaint_ID, order_id)

Complaints(Complaint_ID, filed-date-time, text, status, User_ID, Employee_ID, handled_date_time)

ComplainstOnShops(Complain_ID, Shop_Name)

Employees( Employee_ID, name, salary)

R1(A, B, C, D)
Keys: AB, AD <- Example from lab manual
Primary Key: AB
FDs: AB → CD, A → D
The relation is in 3NF.

*Note:*
*whitespace between key attributes denotes that it is a joint key e.g. Order-ID Product-Name is the same as AB.*
*comma will represent separate keys e.g. Product_Name, start_Date is the same as A, B.*

**Shops (Shop-Name)**

Key: Shop-Name

Primary Key: Shop-Name

No FD.

In 3NF.

**Price history (Product-name, price, start-date, end-date)**

Key: Product-Name

Primary Key: Product-Name

FD: Product-Name -> price, start-date, end-date

Since product-name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Orders (Order-ID, User-ID¸ shipping address)**

Key: Order-ID

Primary Key: Order-ID

FD: Order-ID -> User-ID, shipping address

Since Order-ID User-ID is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Users (User-ID, name)**

Key: User-ID

Primary Key: User-ID

FD: User-ID -> name

Since it’s a 2 attribute FD, it is in 3NF.

**Employees (Employees-ID, name, salary)**

Key: Employee-ID

Primary Key: Employee-ID

FD: Employee-ID -> name, salary

Since Employee-ID is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Products (Product-Name, maker, category)**

Key: Product-Name

Primary Key: Product-Name

FD: Product-Name -> maker, category

Since Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Products-In-Orders (product-name, status, delivery-date, qty, price, shop_name)**

Key: Product-Name

Primary Key: Product-Name

FD: Product-Name -> status, delivery-date, qty, price, shop_name

Since Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Products-In-Shop (Product-name, price, qty)**

Key: Product-Name

Primary Key: Product-Name

FD: Product-Name -> price, qty

Since Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Complaints (Complaint-ID, Employee-ID, User-ID, status, text, filled-date-time, handled-date-time)**

Key: Complaint-ID

Primary Key: Complaint-ID

FD: Complaint-ID -> Employee-ID, User-ID, status, text, filled-date-time, handled-date-time

Since Complaint-ID is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Complaints-on-Order (Complaint-ID, order-ID)**

Key: Complaint-ID

Primary Key: Complaint-ID

FD: Complaint-ID -> order-ID

Since Complaint-ID is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Complaints-On-Shops (Complaint-ID, Shop-Name)**

Key: Complaint-ID

Primary Key: Complaint-ID

FD: Complaint-ID -> Shop-Name

Since Complaint-ID is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**ShopIn-ProductsInShop(Shop-Name, Product-Name, Start-date, End-date, price-variation)**

Key: Shop-Name Product-Name

Primary Key: Shop-Name Product-Name

FD: Shop-Name Product-Name -> Start-date, End-date, price-variation

Since Shop-Name Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Orders-In-ProductsInOrders (Order-ID, product-name, date-time)**

Key: Order-ID Product-Name

Primary Key: Order-ID Product-Name

FD: Order-ID Product-Name -> date-time

Order-ID and Product-Name determine date-time as for each product added to an order, there will be a date-time recorded hence needing Order-ID to first identify the order the product is in and then, using Product-Name to determine date-time for that specific product.

Since Order-ID Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

**Feedback (User-ID, Product-Name, date-time, comment, rating)**

Key: User-ID Product-Name

Primary Key: User-ID Product-Name

FD: User-ID Product-Name -> date-time, comment, rating

Since User-ID Product-Name is a key and is in the LHS of every non-trivial FD, this relation is in 3NF.

SQL Queries used:

**TABLES**

```
CREATE TABLE Users(
        user_id VARCHAR(50) NOT NULL,
        name VARCHAR(50) NOT NULL,
        PRIMARY KEY(user_id)
);
CREATE TABLE Orders(
        order_id INT NOT NULL,
        shipping_address VARCHAR(50) NOT NULL,
        user_id VARCHAR(50) NOT NULL,
        PRIMARY KEY(order_id),
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE ProductInShops(
        product_name VARCHAR(50) NOT NULL,
        price FLOAT NOT NULL,
        quantity INT NOT NULL,
        PRIMARY KEY (product_name),
        FOREIGN KEY (product_name) REFERENCES Products(product_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE

);

CREATE TABLE ShopInProductsinShops(
```

```
	shop_name VARCHAR(50) NOT NULL,
	product_name VARCHAR(50) NOT NULL,
	start_date DATE NOT NULL,
	end_date DATE NOT NULL,
	price_variation FLOAT NOT NULL,
	PRIMARY KEY (shop_name, product_name),
	FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
	ON DELETE CASCADE
	ON UPDATE CASCADE,
	FOREIGN KEY (product_name) REFERENCES ProductInShops(product_name)
	ON DELETE CASCADE
	ON UPDATE CASCADE
);

CREATE TABLE ProductsInOrders(
	PIO_ID INT NOT NULL identity(1,1),
	product_name VARCHAR(50) NOT NULL,
	status VARCHAR(20) NOT NULL,
	delivery_date SMALLDATETIME*** NOT NULL,
	quantity INT NOT NULL,
	shop_name VARCHAR(50) NOT NULL,
	price FLOAT NOT NULL,
	PRIMARY KEY (PIO_ID),
	FOREIGN KEY (product_name)  REFERENCES Products(product_name)
	ON DELETE CASCADE
	ON UPDATE CASCADE,
	FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
	ON DELETE CASCADE
```

```
        ON UPDATE CASCADE
);


CREATE TABLE OrderInProductsinOrders(
        order_id INT NOT NULL,
        PIO_ID INT NOT NULL,
        date_time smalldatetime NOT NULL,
        PRIMARY KEY(order_id, PIO_ID),
        FOREIGN KEY (order_id) REFERENCES Orders(order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
        FOREIGN KEY (PIO_ID) REFERENCES ProductsInOrders(PIO_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Feedback(
        user_id VARCHAR(50) NOT NULL,
        PIO_ID INT NOT NULL,
        rating INT not null,
        comment VARCHAR(100) NOT NULL,
        date_time SMALLDATETIME NOT NULL,
        CHECK(rating>=0 AND rating <= 5),
        PRIMARY KEY(user_id, PIO_ID),
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
```

```
        FOREIGN KEY (PIO_ID) REFERENCES ProductsInOrders(PIO_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);



CREATE TABLE Employees(
        employee_id INT NOT NULL,
        name VARCHAR(50) NOT NULL,
        salary FLOAT NOT NULL,
        PRIMARY KEY(employee_id)
);

CREATE TABLE Complaints(
        complaint_id INT NOT NULL,
        filed_date_time SMALLDATETIME NOT NULL,
        text VARCHAR(200) NOT NULL,
        user_id VARCHAR(50) NOT NULL,
        status VARCHAR(20) NOT NULL,
        employee_id INT NOT NULL,
        handled_date_time SMALLDATETIME NULL,
        CHECK (handled_date_time >= filed_date_time),
        CHECK(status = 'Fulfilled' OR status = 'Pending'),
        PRIMARY KEY (complaint_id),
        FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
```

```
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE

);


CREATE TABLE PriceHistory(
        product_name VARCHAR(50) NOT NULL,
        shop_name VARCHAR(50 NOT NULL,
        price FLOAT NOT NULL,
        start_date SMALLDATETIME NOT NULL,
        end_date SMALLDATETIME NOT NULL,
        PRIMARY KEY (product_name, shop_name, start_date, end_date),
        FOREIGN KEY (product_name) REFERENCES ProductInShops(product_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
        FOREIGN KEY (shop_name) REFERENCES ProductInShops(product_name)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);


CREATE TABLE ComplaintsOnShops(
        complaint_id INT NOT NULL,
        shop_name VARCHAR(50) NOT NULL,
        PRIMARY KEY (complaint_id),
```

```
        FOREIGN KEY (complaint_id) REFERENCES Complaints(complaint_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
        FOREIGN KEY (shop_name) REFERENCES Shops(shop_name)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);



CREATE TABLE ComplaintOnOrders(
        complaint_id INT NOT NULL,
        order_id INT NOT NULL,
        PRIMARY KEY (complaint_id),
        FOREIGN KEY (complaint_id) REFERENCES Complaints(complaint_id),
        FOREIGN KEY (order_id) REFERENCES Orders(order_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

**Query 1:**

```
SELECT AVG(price) AS AverageiPhoneXsPrice
FROM PriceHistory, ShopInProductsinShops
WHERE PriceHistory.product_name = 'iPhone Xs'

AND ((PriceHistory.start_date >= '2020-08-01' AND PriceHistory.start_date <= '2020-08-30')

OR (PriceHistory.end_date >= '2020-08-01 00:00:00' AND PriceHistory.end_date <= '2020-08-30
00:00:00')

OR (PriceHistory.start_date < '2020-08-01 00:00:00' AND PriceHistory.end_date > '2020-08-30
00:00:00'));
```

//Assumption; each day only one change in price can be made
No bad entries allowed. Eg. if there is an entry with start date 2020-08-01 and end date 2020-08-10 and there will not be another entry with start date before 2020-08-10.

To find average price we are interested in the price of Iphone Xs between 1 Aug and 31 Aug. However in PriceHistory, it depends on when we enter our record, there are 3 cases that the record entered will affect computation of average price

1. When start date after 1 Aug but before 31
2. When end date after 1 Aug but before 31
3. When start date before 1 Aug and end date after 31

**Query 2:**

```
WITH table1 AS(SELECT product_name, AVG(rating) AS avg_rating

FROM Feedback f

JOIN ProductsInOrders p ON f.PIO_ID = p.PIO_ID

GROUP BY p.product_name),



tablenoOf5 AS (SELECT p.product_name, COUNT(rating) AS noOf5

FROM Feedback f

JOIN ProductsInOrders p ON f.PIO_ID = p.PIO_ID

WHERE rating = 5 AND date_time >= '08/01/2020' AND date_time <= '08/30/2020'

GROUP BY p.product_name)



SELECT tablenoOf5.product_name, noOf5   //, avg_rating

FROM tablenoOf5, table1
```

```
WHERE noOf5 >= 100 AND tablenoOf5.product_name = table1.product_name

ORDER BY table1.avg_rating;
```

**Query 3:**

Average of all delivered

```
WITH junePurchasedDelivered

        AS (SELECT p.PIO_ID, o.date_time, p.delivery_date, p.product_name

        FROM OrderInProductsinOrders o

        JOIN ProductsInOrders p ON o.PIO_ID = p.PIO_ID

        WHERE o.date_time >= '06/01/2020' AND o.date_time <= '06/30/2020' AND p.status =
'delivered')


        SELECT AVG(DATEDIFF(day, jpd.date_time, jpd.delivery_date)*1.0) AS duration

        FROM junePurchasedDelivered jpd
```

Average of all delivered by product

```
WITH junePurchasedDelivered

        AS (SELECT p.PIO_ID, o.date_time, p.delivery_date, p.product_name

        FROM OrderInProductsinOrders o

        JOIN ProductsInOrders p ON o.PIO_ID = p.PIO_ID

        WHERE o.date_time >= '06/01/2020' AND o.date_time <= '06/30/2020' AND p.status =
'delivered')


        SELECT product_name, AVG(DATEDIFF(day, jpd.date_time, jpd.delivery_date)*1.0) AS
duration

        FROM junePurchasedDelivered jpd
```

```
GROUP BY product_name
```

**Query 4:**

```
WITH employeeLatencyTable(employee_id, latency) AS(
        select employee_id, datediff(MINUTE,filed_date_time,handled_date_time) AS latency
                from Complaints
                where status ='fulfilled'),
        minimumLatencyTable(employee_id, avg_Latency)AS
                (select employee_id, avg(latency*1.0)
                from employeeLatencyTable
                group by employee_id)

        select employee_id as EmployeeWithLowestLatency
        from minimumLatencyTable
        where avg_Latency = (select min(avg_Latency) from minimumLatencyTable);
```

**Query 5:**

```
        WITH samsungPROD AS(SELECT product_name
        FROM Products
        WHERE maker = 'Samsung')

        SELECT sp.product_name, COUNT(s.shop_name) AS noOfShop
        FROM samsungPROD sp
        JOIN ShopInProductsinShops s ON sp.product_name = s.product_name
        GROUP BY sp.product_name
```

**Query 6:**

```
WITH Revenue AS
(SELECT shop_name, SUM(price*quantity) AS shopRevenue
FROM ProductsInOrders pio, OrderInProductsinOrders o
WHERE pio.PIO_ID = o.PIO_ID
AND o.date_time >= '08/01/2020'
AND o.date_time <= '08/31/2020'
GROUP BY shop_name)

SELECT shop_name
FROM Revenue
WHERE shopRevenue >= ALL(SELECT shopRevenue
                         FROM Revenue);
```

```
SELECT top 1 shop_name, SUM(price*quantity) AS REVENUE
FROM ProductsInOrders pio
        JOIN OrderInProductsinOrders o ON pio.PIO_ID = o.PIO_ID
WHERE date_time >= '08/01/2020'  AND date_time <= '08/31/2020'
GROUP BY shop_name
ORDER BY REVENUE DESC;
```

(method 2)

```
WITH augOrders
AS (SELECT PIO_ID
```

```
FROM OrderInProductsinOrders
WHERE date_time >= '08/01/2020' AND date_time <= '08/31/2020'),

shopRevenue
AS(SELECT shop_name, SUM(price*quantity) AS REVENUE
FROM ProductsInOrders pio, augOrders ao
WHERE pio.PIO_ID = ao.pio_id
GROUP BY shop_name)

SELECT *
FROM shopRevenue
WHERE REVENUE = (SELECT max(sr.REVENUE) as REVENUE
                 FROM shopRevenue sr)
```

**Query 7:**

```
WITH TotalComplaints(user_id, complaints)  AS
(SELECT user_id, count(complaint_id)
FROM Complaints
GROUP BY user_id),

maxComplaints(user_id) AS
(Select c.user_id
from TotalComplaints c
where c.complaints = (SELECT MAX(complaints)
                      FROM TotalComplaints)),

mostComplaintUserProducts(product_name, price) as(
```

```
SELECT PRODUCT_NAME, price

from ProductsInOrders

where PIO_ID in(select pio_id

                from feedback f, maxComplaints c

                where f.user_id = c.user_id))


select max(price) as MostExpensiveProduct

from mostComplaintUserProducts;
```

**Query 8:**

8. Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

It's intended to return 3 products. Steps: find the top 5 products first, then find products that have never been purchased by some users, then take overlap of the 2.

Rationale: The wording, cos they didnt say "top 5 most purchased products by other users in August 2020 that have never been purchased by some users". So we took it to be, those products that have never been purchased by some users, and are within the top 5 of August

```
SELECT Top5Products

FROM

    (SELECT P2 AS Top5Products

    FROM

        (SELECT PC1.Product_name P1, PC1.counter C1, PC2.Product_name P2,
PC2.counter C2

        FROM (SELECT Product_name, COUNT(product_name) as counter
```

```
                    FROM OrderInProductsInOrders, ProductsInOrders

                    WHERE date_time >= '2020-08-01 00:00:00' AND date_time
<='2020-08-30 00:00:00'

                    GROUP BY Product_name)PC1,

                    (SELECT Product_name, COUNT(product_name) as counter

                    FROM OrderInProductsInOrders, ProductsInOrders

                    WHERE date_time >= '2020-08-01 00:00:00' AND date_time
<='2020-08-30 00:00:00'

                    GROUP BY Product_name) PC2

                WHERE PC1.counter >= PC2.counter) TABLE1

        GROUP BY P2

        HAVING COUNT(P2) <=5) AS TABLE2

        LEFT OUTER JOIN

        (SELECT product_name, OrderInProductsInOrders.order_id

        FROM (ProductsInOrders JOIN OrderInProductsInOrders ON ProductsInOrders.PIO_ID
= OrderInProductsinOrders.PIO_ID)

        JOIN Orders ON OrderInProductsInOrders.order_id = Orders.order_id) AS TABLE3

        ON TABLE2.Top5Products = TABLE3.product_name

WHERE order_id is NULL;
```

//Assumption; what the question is looking for is first finding the top 5 products in the month of August, then of these top 5 products find the products that have not been purchased by every user (these are the products that have never been purchased by some users)

9. Find products that are increasingly being purchased over at least 3 months.

```
WITH ProductByMonth(product_name, month, qty) AS

(SELECT product_name, month(delivery_date) AS month, sum(quantity) AS qty
```

```
FROM ProductsInOrders

GROUP BY product_name, month(delivery_date))



SELECT DISTINCT p1.product_name

FROM ProductByMonth p1, ProductByMonth p2, ProductByMonth p3

WHERE p1.product_name = p2.product_name AND p1.month= p2.month-1 AND
p1.product_name = p3.product_name and p2.month=p3.month-1

and p1.qty<p2.qty and p2.qty< p3.qty;
```

5. Produce a list that contains (i) all products made by Samsung, and (ii) for each of them, the number of shops on Sharkee that sell the product.

```
WITH samsungPROD AS

(SELECT product_name

FROM Products

WHERE maker = 'Samsung')


SELECT sp.product_name, COUNT(s.shop_name) AS noOfShop

FROM samsungPROD sp, ShopInProductsinShops s

WHERE sp.product_name = s.product_name

GROUP BY sp.product_name
```

| | product_name | maker | category |
|---|---|---|---|
| | bagel | idk | food |
| | Bottle | Friends | Others |
| | iPhone X | Apple | Gadgets |
| | iPhone Xs | Apple | Gadgets |
| | lipstick | chanel | cosmetics |
| | Logitech GPro mouse | Logitech | Gadgets |
| | Pen | Stabilo | Others |
| | power lipstick | sephora | cosmetics |
| | Razer Speaker | Razer | Gadgets |
| ▸ | Samsung earpiece | Samsung | Gadgets |
| | Samsung galaxy 1 | Samsung | Gadgets |
| | Samsung galaxy 2 | Samsung | Gadgets |
| | stapler | Max Staples | Others |
| * | NULL | NULL | NULL |

| | product_name |
|---|---|
| 1 | Samsung earpiece |
| 2 | Samsung galaxy 1 |
| 3 | Samsung galaxy 2 |

| | shop_name | product_name | start_date | end_date | price_variation |
|---|---|---|---|---|---|
| | iStudio | iPhone Xs | 2020-06-08 | 2020-06-08 | 50 |
| ▸ | iStudio | Samsung Galaxy 1 | 2020-06-01 | 2020-08-08 | 20 |
| | Samsung Shop1 | Samsung Galaxy 1 | 2020-06-01 | 2020-08-01 | 30 |
| | Samsung Shop2 | Samsung Galaxy 2 | 2020-06-02 | 2020-07-02 | 30 |
| | Samsung Shop3 | Samsung Earpiece | 2020-06-03 | 2020-08-08 | 30 |
| | sephora | lipstick | 2020-06-08 | 2020-08-08 | 23 |
| * | NULL | NULL | NULL | NULL | NULL |

| | product_name | noOfShop |
|---|---|---|
| 1 | Samsung earpiece | 1 |
| 2 | Samsung galaxy 1 | 2 |
| 3 | Samsung galaxy 2 | 1 |

6. Find shops that made the most revenue in August 2020.

```
WITH Revenue AS

(SELECT shop_name, SUM(price*quantity) AS shopRevenue

FROM ProductsInOrders pio, OrderInProductsinOrders o

WHERE pio.PIO_ID = o.PIO_ID

AND o.date_time >= '08/01/2020'

AND o.date_time <= '08/31/2020'

GROUP BY shop_name)


SELECT shop_name

FROM Revenue

WHERE shopRevenue >= ALL(SELECT shopRevenue

                         FROM Revenue);
```

| PIO_ID | product_name | status | delivery_date | quantity | shop_name | price |
|---|---|---|---|---|---|---|
| 1 | Power LipStick | delivered | 2020-08-01 00:0... | 2 | sephora | 15 |
| 2 | lipstick | delivered | 2020-08-04 00:0... | 3 | sephora | 200 |
| 4 | lipstick | delivered | 2020-08-05 00:0... | 1 | watsons | 18 |
| 5 | bagel | delivered | 2020-08-22 00:0... | 6 | breadtalk | 3 |
| 6 | bagel | pending | 2020-08-12 00:0... | 4 | polar | 4 |
| 13 | iPhone Xs | delivered | 2020-08-01 00:0... | 3 | jShop | 1800 |
| 14 | IPhone Xs | delivered | 2020-08-30 00:0... | 4 | iStudio | 1600 |
| 15 | iPhone Xs | delivered | 2020-09-15 00:0... | 10 | Phone Shop | 1400 |
| 19 | iPhone Xs | delivered | 2020-08-20 00:0... | 4 | iPhone Best | 1550 |
| 23 | Logitech GPro ... | delivered | 2020-06-05 00:0... | 20 | Logitech Official | 180 |
| 25 | Razer Speaker | delivered | 2020-06-03 00:0... | 3 | Razer Official | 100 |
| 28 | Logitech GPro ... | delivered | 2020-04-06 00:0... | 10 | Logitech Official | 180 |
| 30 | Razer Speaker | delivered | 2020-04-01 00:0... | 20 | Razer Official | 100 |
| 31 | Logitech GPro ... | pending | 2020-08-08 00:0... | 100 | Logitech Official | 180 |
| 32 | Razer Speaker | delivered | 2020-05-08 00:0... | 100 | Razer Official | 100 |
| 33 | bagel | pending | 2020-09-22 12:0... | 20 | breadtalk | 4 |
| 34 | bagel | delivered | 2020-10-02 13:0... | 30 | breadtalk | 4 |
| 35 | bagel | delivered | 2020-07-02 13:0 | 5 | breadtalk | 5 |

| | order_id | PIO_ID | date_time |
|---|---|---|---|
| ▶ | 1 | 1 | 2020-08-01 00:0... |
| | 2 | 4 | 2020-08-02 00:0... |
| | 3 | 2 | 2020-08-02 00:0... |
| | 3 | 4 | 2020-02-18 00:0... |
| | 4 | 4 | 2020-08-15 00:0... |
| | 4 | 5 | 2020-08-12 00:0... |
| | 5 | 2 | 2020-04-11 00:0... |
| | 5 | 4 | 2020-10-09 00:0... |
| | 5 | 6 | 2020-08-05 00:0... |
| | 5 | 34 | 2020-09-02 00:0... |
| | 5 | 36 | 2020-09-02 00:0... |
| | 5 | 38 | 2020-09-02 00:0... |
| | 55 | 23 | 2020-06-01 00:0... |
| | 65 | 1 | 2020-10-24 00:0... |
| | 65 | 4 | 2020-11-18 00:0... |
| | 65 | 5 | 2020-08-16 00:0... |
| | 65 | 25 | 2020-06-02 00:0... |

| | PIO_ID | product_name | status | delivery_date | quantity | shop_name | price | order_id | PIO_ID | date_time |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Power LipStick | delivered | 2020-09-01 00:00:00 | 2 | sephora | 15 | 1 | 1 | 2020-08-01 00:00:00 |
| 2 | 4 | lipstick | delivered | 2020-08-05 00:00:00 | 1 | watsons | 18 | 2 | 4 | 2020-08-02 00:00:00 |
| 3 | 2 | lipstick | delivered | 2020-08-04 00:00:00 | 3 | sephora | 200 | 3 | 2 | 2020-08-02 00:00:00 |
| 4 | 4 | lipstick | delivered | 2020-08-05 00:00:00 | 1 | watsons | 18 | 4 | 4 | 2020-08-15 00:00:00 |
| 5 | 5 | bagel | delivered | 2020-08-22 00:00:00 | 6 | breadtalk | 3 | 4 | 5 | 2020-08-12 00:00:00 |
| 6 | 6 | bagel | pending | 2020-08-12 00:00:00 | 4 | polar | 4 | 5 | 6 | 2020-08-05 00:00:00 |
| 7 | 5 | bagel | delivered | 2020-08-22 00:00:00 | 6 | breadtalk | 3 | 65 | 5 | 2020-08-16 00:00:00 |

| | shop_name | shopRevenue |
|---|---|---|
| 1 | breadtalk | 36 |
| 2 | polar | 16 |
| 3 | sephora | 630 |
| 4 | watsons | 36 |

| | shop_name |
|---|---|
| 1 | sephora |

IF two shops with top revenue

| | shop_name | shopRevenue |
|---|---|---|
| 1 | breadtalk | 36 |
| 2 | polar | 16 |
| 3 | sephora | 33 |
| 4 | watsons | 36 |

| | shop_name |
|---|---|
| 1 | breadtalk |
| 2 | watsons |