Lab 5 - Appendix B

Q1) Find the average price of “iPhone Xs” on Sharkee from 1 August 2020 to 31 August 2020

```
SELECT AVG(PHPrice)
FROM PriceHistory
WHERE ProductName = 'iPhone Xs' AND (PHStartDate <= '2020-08-31
23:59:59' AND PHEndDate >= '2020-08-01 00:00:00');
```

Q2) Find products that received at least 100 ratings of “5” in August 2020, and order them by their average ratings.

```
SELECT ProductName, AVG(Rating) as AVGR
FROM Feedbacks
WHERE ProductName in (
        SELECT ProductName
        FROM Feedbacks
        WHERE Rating = 5 AND FeedbackDateTime >= '2020-08-01
        00:00:00' AND FeedbackDateTime <= '2020-08-31 23:59:59'
        GROUP BY ProductName
        HAVING COUNT(Rating) >= 100)
GROUP BY ProductName
ORDER BY AVGR DESC
```

Q3) For all products purchased in June 2020 that have been delivered, find the average time from the ordering date to the delivery date.

Assumptions: When status is delivered or returned it means that product has been delivered or delivered then returned. Else, the product cannot be on delivered or returned status.

```
SELECT AVG(DATEDIFF(DAY, O.OrderDateTime,
CAST(PiO.PIOdeliverydate AS datetime))) AS 'AVGDate'
FROM ProductsInOrders AS PiO, Orders AS O
WHERE (PiO.PIOstatus = 'delivered' OR PiO.PIOstatus = 'returned')
AND PiO.OrderId = O.OrderId
AND O.OrderDateTime >= '2020-06-01 00:00:00'
AND O.OrderDateTime <= '2020-06-30 23:59:59'
```

Q4) Let us define the "latency" of an employee by the average that he/she takes to process a complaint. Find the employee with the smallest latency.

```
SELECT EmployeeId
FROM Complaints
GROUP BY EmployeeId
HAVING (AVG(DATEDIFF(DAY, ComplaintFiled, HandledDateTime)) = (
        SELECT MIN(LATENCY)
        FROM (
                SELECT EmployeeId AS EmployId, AVG(DATEDIFF(DAY,
                ComplaintFiled, HandledDateTime)) AS LATENCY
                FROM Complaints
                GROUP BY EmployeeId) AS Table1))
```

Q5) 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.

```
SELECT ProductName
FROM Products
WHERE Maker = 'Samsung'

SELECT COUNT(ShopName)
FROM ProductsInShops
WHERE ProductName IN
(SELECT ProductName
FROM Products
WHERE Maker = 'Samsung')
```

Q6) Find shops that made the most revenue in August 2020

Assumption: When product is delivered, shops will then obtain the revenue for selling the product. If returned, then shops will have to refund the money back to the customers

```
SELECT ShopName
FROM ProductsInOrders
WHERE PIOdeliverydate>= '2020-08-01 00:00:00'
AND PIOdeliverydate <= '2020-08-31 23:59:59'
AND PIOstatus = 'Delivered'
AND (PIOquantity * PIOprice) IN (
SELECT MAX(PIOquantity * PIOprice)
FROM ProductsInOrders
WHERE PIOdeliverydate>= '2020-08-01 00:00:00'
AND PIOdeliverydate <= '2020-08-31 23:59:59'
AND PIOstatus = 'Delivered')
```

Q7) For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.

```
WITH TempTable (Counting, Useridd) AS (
SELECT COUNT(ComplaintId), UserId
FROM Complaints
GROUP BY UserId)

SELECT ProductName, UserId
FROM ProductsInOrders, Orders
WHERE Orders.OrderId = ProductsInOrders.OrderId
AND UserId IN (
        SELECT Useridd
        FROM TempTable
        WHERE Counting = (
                SELECT MAX(Counting)
                FROM TempTable)
        )
        AND PIOprice IN (
                SELECT MAX(PIOprice)
                FROM Orders, ProductsInOrders
                WHERE Orders.OrderId = ProductsInOrders.OrderId
                AND UserId IN (
                        SELECT Useridd
                        FROM TempTable
                        WHERE Counting = (
                                SELECT MAX(Counting)
                                FROM TempTable)
                        )
                        GROUP BY UserId
)
```

Q8) Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

```
SELECT t4.ProductName
FROM (
	SELECT TOP 5 ProductName
	FROM ProductsInOrders AS PIO, Orders AS o
	WHERE o.OrderDateTime >='2020-08-01 00:00:00' AND
	o.OrderDateTime <= '2020-08-31 23:59:59' AND PIO.OrderID =
	o.OrderID
	GROUP BY ProductName
	ORDER BY COUNT (*) DESC) AS t4
LEFT JOIN (
	SELECT ProductName
	FROM (
		SELECT ProductName, COUNT(1) AS count_p
		FROM(
			SELECT DISTINCT o.OrderId,ProductName
			FROM ProductsInOrders AS PIO
			LEFT JOIN Orders AS o
			ON PIO.OrderId = o.OrderId) AS t1
		GROUP BY ProductName) AS t2,
	(SELECT COUNT(1) count_c
	FROM Users) t3
WHERE t2.count_p = t3.count_c) AS t5
ON t4.ProductName = t5.ProductName
WHERE t5.ProductName IS NULL
```

Q9) Find products that are increasingly being purchased over at least 3 months.

```
WITH TableOfNumPurchased(ProductName,PMonth,NumPurchased) AS (
        SELECT ProductName, MONTH(PIOdeliverydate) AS PMonth,
        SUM(ProductsInOrders.PIOquantity)
        FROM ProductsInOrders
        GROUP BY ProductName, MONTH(PIOdeliverydate)
)

SELECT DISTINCT t1.ProductName
FROM TableOfNumPurchased AS t1, TableOfNumPurchased AS t2,
TableOfNumPurchased AS t3
WHERE t1.ProductName = t2.ProductName
AND t2.ProductName = t3.ProductName
AND t2.PMonth = t1.PMonth+1
AND t3.PMonth = t1.PMonth+2
AND t2.NumPurchased > t1.NumPurchased
AND t3.NumPurchased > t2.NumPurchased
```