/*
Find the average price of “iPhone Xs” on Shiokee from
1 August 2021 to 31 August 2021.
*/
17

```
--To get average, multiply price by time, and divide by total time.
SELECT SUM(Price * Overlap) / SUM(Overlap)
FROM ( --This subquery returns the "weight" of each price record and the price to
be considered. Longer overlap -> greater consideration in avg.
    SELECT Price, DATEDIFF(day, IIF({d '2021-08-01'} > Start_date, {d
'2021-08-01'}, Start_date), IIF({d '2021-08-31'} < End_date, {d '2021-08-31'},
End_date)) AS Overlap
    FROM ( --This subquery returns all price records which overlap with the
desired timeframe. We only need start date, end date, and price.
        SELECT Start_date, End_date, Price
        FROM PriceRecord
        WHERE Pname = 'iPhone Xs' AND End_date >= {d '2021-08-01'} AND Start_date
<= {d '2021-08-31'}
    ) AS X
) AS Y
```

/*
Find products that received at least 100 ratings of “5” in August 2021,
and order them by their average ratings.
*/
17

```
--Take only those products that satisfy the given condition, and get their name
and average rating.
SELECT Pname, AVG(CAST(Rating AS DECIMAL(10, 2))) AS AverageRating
FROM Feedback
WHERE Pname IN ( --Find the names of the products which satisfy the specified
condition.
    SELECT Pname
```

```
     FROM Feedback AS X
     WHERE Rating = 5 AND Feedback_date_time >= {d '2021-08-01'} AND
Feedback_date_time <= {d '2021-08-31'}
     GROUP BY Pname
     HAVING COUNT(*) >= 10 --Number reduced to 10 for demonstration purposes.
)
GROUP BY Pname
ORDER BY AverageRating
```

/*
For all products purchased in June 2021 that have been delivered,
find the average time from the ordering date to the delivery date.
*/
KSC

```
--Take the average of each TimeInShipping for each product.
SELECT Pname, AVG(CAST(TimeInShipping AS DECIMAL(10, 2))) AS AvgTime
FROM ( --For each product in order, find the time spent in shipping.
    SELECT Pname, DATEDIFF(day, Order_date, Delivery_date) AS TimeInShipping
    FROM ProductsInOrders AS P, Orders AS O
    WHERE Status = 'delivered' AND
                    Order_date >= {d '2021-06-01'} AND
                    Order_date <= {d '2021-06-30'} AND
                    P.OID = O.OID --Join the ProductsInOrders and Orders tables
so that the Order_date and Delivery_date can be compared.
) AS X
GROUP BY Pname
```

/*
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.
*/
KSC

```
--Select the employee with the lowest latency.
SELECT TOP 1 EID, LatencyIndiv
FROM ( --For each employee, find their average latency.
    SELECT EID, AVG(CAST(DATEDIFF(day, Filed_date_time, Handled_date_time) AS
DECIMAL(10, 2))) AS LatencyIndiv
    FROM ( --Get all addressed complaints.
        SELECT CID, Filed_date_time, Handled_date_time, EID
        FROM Complaints
        WHERE Complain_status = 'addressed'
    ) AS AddressedComplaints
    GROUP BY EID
) AS EmployeeLatency
ORDER BY LatencyIndiv
```

/*
Produce a list that contains (i) all products made by Samsung, and
(ii) for each of them, the number of shops on Shiokee that sell the product.
*/
17

```
--part (i) -> Pname, part (ii) -> NumOfShopSelling.
SELECT Pname, COUNT(DISTINCT Sname) AS NumOfShopSelling
```

```
FROM ProductsInShops
WHERE Maker = 'Samsung'
GROUP BY Pname
```

/*
Find shops that made the most revenue in August 2021.
*/
17

```
--Get the shop with the most revenue.
SELECT TOP 1 Sname
FROM ProductsInOrders
WHERE OID IN ( --Consider the orders made during the specified timeframe.
    SELECT OID
    FROM Orders
    WHERE {d '2021-08-01'} <= Order_date AND Order_date <= {d '2021-08-31'}
)
GROUP BY Sname
ORDER BY SUM(OPrice * OQuantity) DESC --Order by revenue per shop.
```

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

The Top Version:

```
SELECT TOP 1 UID, PName
```

```
FROM ProductsInOrders Inner Join Orders
   ON ProductsInOrders.OID = Orders.OID
WHERE UID = (SELECT TOP 1 UID
            FROM Complaints
            GROUP By UID
            ORDER BY COUNT(*) DESC)
ORDER BY OPrice DESC
```

The Non-Top Version:

```
SELECT UID, Pname
FROM (
   SELECT DISTINCT ProductsInOrders.Pname AS Pname, ProductsInOrders.OPrice AS OPrice,
ProductsInOrders.OPID AS OPID, ProductsInOrders.OID AS OID, MaxUsersOIDUID.UID AS UID
   FROM (
       (   SELECT DISTINCT Orders.OID AS OID, Orders.UID AS UID
           FROM (
               (   SELECT UID
                   FROM (SELECT UID, COUNT(*) AS ComplaintNum
                       FROM Complaints
                       GROUP BY UID) AS CountTable
                   WHERE CountTable.ComplaintNum in (
                       SELECT MAX(ComplaintNum2)
                       FROM (SELECT UID, COUNT(*) AS ComplaintNum2
                       FROM Complaints
                       GROUP BY UID) AS CountTable2
                   )
               ) AS MaxUsersTable
           INNER JOIN Orders ON MaxUsersTable.UID = Orders.UID)
       ) AS MaxUsersOIDUID
       INNER JOIN ProductsInOrders ON MaxUsersOIDUID.OID = ProductsInOrders.OID
   )
) AS SemiFinalTable
WHERE SemiFinalTable.OPrice in (
   SELECT MAX(ProductsInOrders.OPrice) AS OPrice
   FROM (
       (   SELECT DISTINCT Orders.OID AS OID, Orders.UID AS UID
           FROM (
               (   SELECT UID
                   FROM (SELECT UID, COUNT(*) AS ComplaintNum3
                       FROM Complaints
```

```
                    GROUP BY UID) AS CountTable3
                WHERE CountTable3.ComplaintNum3 in (
                    SELECT MAX(ComplaintNum4)
                    FROM (SELECT UID, COUNT(*) AS ComplaintNum4
                    FROM Complaints
                    GROUP BY UID) AS CountTable4
                )
            ) AS MaxUsersTable2
        INNER JOIN Orders ON MaxUsersTable2.UID = Orders.UID)
    ) AS MaxUsersOIDUID2
    INNER JOIN ProductsInOrders ON MaxUsersOIDUID2.OID = ProductsInOrders.OID
  )
)
```

/*
Find products that have never been purchased by some users, but are the top 5 most purchased
products by other users in August 2021.
(try 74 or 23)
*/

Joining tables to find the quantity of each product sold in Aug, 2021
Exclude those products that have been ever purchased by user with UID = 'TBD'
Display the top 5 products according to their quantity sold in Aug, 2021
Peilin

```
SELECT TOP 5 Pname
FROM (
    SELECT ProductsInOrders.OQuantity, ProductsInOrders.OID,
ProductsInOrders.Pname
    FROM Orders INNER JOIN ProductsInOrders
    On ProductsInOrders.OID = Orders.OID
    WHERE MONTH(Order_date) = 8 AND YEAR(Order_date) = 2021
) AS OrderProducts
WHERE Pname NOT IN (
    SELECT Pname
    FROM Orders INNER JOIN ProductsInOrders
```

```
    On ProductsInOrders.OID = Orders.OID
    WHERE UID = '100')
GROUP BY Pname
ORDER BY SUM(OQuantity)




-- SELECT TOP 5 OPID
-- FROM (
--     SELECT ProductsInOrders.Pname, ProductsInOrders.OID,
ProductsInOrders.OQuantity
--     FROM Orders INNER JOIN ProductsInOrders
--     On ProductsInOrders.OID = Orders.OID
--     WHERE MONTH(Order_date) = 8 AND YEAR(Order_date) = 2021
-- ) AS OrderProducts
-- WHERE NOT EXISTS (
--     SELECT OPID --OPID
--     FROM Orders INNER JOIN ProductsInOrders
--     On ProductsInOrders.OID = Orders.OID
--     WHERE UID = '10'
-- )
-- GROUP BY Pname
-- ORDER BY SUM(OQuantity)

-- SELECT TOP 5 OPID
-- FROM (
--     SELECT ProductsInOrders.Pname, ProductsInOrders.OID,
ProductsInOrders.OQuantity
--     FROM Orders INNER JOIN ProductsInOrders
--     On ProductsInOrders.OID = Orders.OID
--     WHERE MONTH(Order_date) = 8 AND YEAR(Order_date) = 2021
-- )
-- WHERE NOT EXISTS (
--     SELECT Pname --OPID
--     FROM Orders INNER JOIN ProductsInOrders
--     On ProductsInOrders.OID = Orders.OID
--     WHERE UID = '10'
```

```
-- )
-- GROUP BY Pname
-- ORDER BY SUM(OQuantity)


SELECT Pname --OPID
FROM Orders INNER JOIN ProductsInOrders
On ProductsInOrders.OID = Orders.OID
WHERE UID = '10'
```

/*
Find products that are increasingly being purchased over at least 3 months
*/
Joshua

```
-- SELECT Pname, SUM(OQuantity) AS JuneTotalPurchase
-- FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID = Orders.OID
-- WHERE {d '2021-06-01'} <= Order_date AND {d '2021-06-30'} >= Order_date
-- GROUP BY Pname

SELECT JuneOrder.Pname
FROM (
       (
           (SELECT Pname, SUM(OQuantity) AS JuneTotalPurchase  -- rename sum
           FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID =
Orders.OID
           WHERE {d '2021-06-01'} <= Order_date AND {d '2021-06-30'} >= Order_date
           GROUP BY Pname
           ) AS JuneOrder
       INNER JOIN
           (SELECT Pname, SUM(OQuantity) AS JulyTotalPurchase
           FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID =
Orders.OID
           WHERE {d '2021-07-01'} <= Order_date AND {d '2021-07-31'} >= Order_date
           GROUP BY Pname
           ) AS JulyOrder
       ON JuneOrder.Pname = JulyOrder.Pname)   -- theta join
```

```
   INNER JOIN
       (SELECT Pname, SUM(OQuantity) AS AugustTotalPurchase
       FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID = Orders.OID
       WHERE {d '2021-08-01'} <= Order_date AND {d '2021-08-31'} >= Order_date
       GROUP BY Pname
       ) AS AugustOrder
   ON JuneOrder.Pname = AugustOrder.Pname
)
WHERE JuneTotalPurchase < JulyTotalPurchase AND JulyTotalPurchase <
AugustTotalPurchase
```

/*Fetch data on the name, year, month and sum of quantity sold in that month of the products;
Duplicate three tables T1, T2, T3 based on the data fetched above;
List out the constraints. We have three cases to consider about the “consecutive months”: 1.three months in the same year 2. First month in December the rest two in the next year 3. The first month and the second month in the same year while the third in the next year
The quantity purchased should be increasing*/

New Version: query 9

```
SELECT T1.Pname
FROM (SELECT ProductsInOrders.Pname, YEAR(Order_date)AS Year, MONTH(Order_date)
as Month, SUM(OQuantity) AS Num
        FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID =
Orders.OID
        GROUP BY ProductsInorders.PName, YEAR(Order_date), MONTH(Order_date)
    ) AS T1,
    (SELECT ProductsInOrders.Pname, YEAR(Order_date)AS Year, MONTH(Order_date) as
Month, SUM(OQuantity) AS Num
        FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID =
Orders.OID
        GROUP BY ProductsInorders.PName, YEAR(Order_date), MONTH(Order_date)
    ) AS T2,
    (SELECT ProductsInOrders.Pname, YEAR(Order_date)AS Year, MONTH(Order_date) as
Month, SUM(OQuantity) AS Num
```

```
        FROM ProductsInOrders INNER JOIN Orders ON ProductsInOrders.OID =
Orders.OID
        GROUP BY ProductsInorders.PName, YEAR(Order_date), MONTH(Order_date)
    ) AS T3
WHERE (T1.Pname = T2.Pname AND T2.Pname=T3.Pname)
    AND(
     (T1.Year=T2.Year AND T2.Year=T3.Year AND (T1.Month + 1 = T2.Month) AND
(T2.Month+1=T3.Month))--case1
    OR
    (T1.Year=T2.Year-1 AND T2.Year=T3.Year AND T1.Month=12 AND T2.Month=1 AND
T3.Month=2)--case2
    OR
    (T1.Year = T2.Year AND (T2.Year =T3.Year -1)  AND T1.Month=11 AND T2.Month =
12 AND T3.Month =1 ))--case3

    AND (T1.Num < T2.Num AND T2.Num<T3.Num)
```

==========

Query 7 draft

```
SELECT DISTINCT Orders.OID AS OID, Orders.UID AS UID
FROM (
   (
       SELECT UID
       FROM (SELECT UID, COUNT(*) AS ComplaintNum
           FROM (ComplaintsOnOrders INNER JOIN Complaints ON ComplaintsOnOrders.CID =
Complaints.CID)
           GROUP BY UID) AS CountTable
       WHERE CountTable.ComplaintNum in (
           SELECT MAX(ComplaintNum2)
           FROM (SELECT UID, COUNT(*) AS ComplaintNum2
           FROM (ComplaintsOnOrders INNER JOIN Complaints ON ComplaintsOnOrders.CID =
Complaints.CID)
           GROUP BY UID) AS CountTable2
       )
   ) AS MaxUsersTable
INNER JOIN Orders ON MaxUsersTable.UID = Orders.UID)
```

```
SELECT TOP 1 PName
FROM ProductsInOrders Inner Join Orders
    ON ProductsInOrders.OID = Orders.OID
WHERE UID = (SELECT TOP 1 UID
            FROM Complaints
            GROUP By UID
            ORDER BY COUNT(*) DESC)
ORDER BY OPrice DESC
```