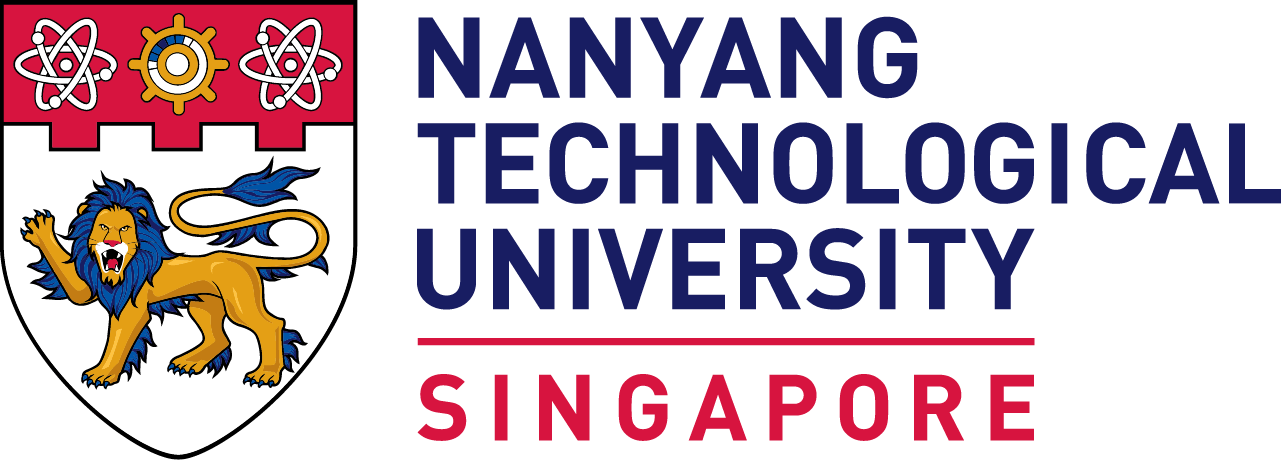
****

**CZ2007 SS3 Group 2 Lab 5 Report**

Members:

Eugene Poh Yang Quan

Wong Yi Pun

Roy Lau Run-Xuan

Chua Zi Jian

Ryan

Koh Jun Kai

# 

# **Contents**

[**Contents**](#_82x5pixpvtmw) **2**

[**Table creation commands**](#_mn96cie9v2cx) **3**

[**Queries**](#_43en2nsunv61) **7**

[Query 1:](#_2jlm53ouno76) 7

[Find the average price of “iPhone Xs” on Shiokee from 1 August 2021 to 31 August 2021.](#_wml3o75bbf7k) 7

[Query 2:](#_t8yv7m8uxoy6) 8

[Find products that received at least 100 ratings of “5” in August 2021, and order them by their average ratings.](#_z6b8g7m0c8on) 8

[Query 3:](#_p0chi0yui6ru) 9

[For all products purchased in June 2021 that have been delivered, find the average time from the ordering date to the delivery date.](#_4z8yaenvi4oa) 9

[Query 4:](#_hr21t5da66p4) 10

[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.](#_wfp4aytznir5) 10

[Query 5:](#_cju78k4r6lk8) 11

[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.](#_l9ceck9woqt) 11

[Query 6:](#_g4uoayxibu74) 12

[Find shops that made the most revenue in August 2021.](#_u7zyctlnu6kt) 12

[Query 7:](#_43iz3j5hlayk) 13

[For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.](#_9tn4ecihyloi) 13

[Query 8:](#_z1qnesdfyzpc) 14

[Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2021.](#_ostqu584382m) 14

[Query 9:](#_mnrr73xjppa8) 15

[Find products that are increasingly being purchased over at least 3 months.](#_irm02ueze76r) 15

[Additional Query 1:](#_6q181u3j7y39) 17

[Frequent shoppers are shoppers who have purchased more than 2 items per shop for at least 2 shops in the last 730 days. Who are the top 3 frequent shoppers in terms of the total cost of the items they have purchased?](#_fsoufunt3w3k) 17

[Additional Query 2:](#_3jcwcv2phdhw) 19

[Popular shops are shops which have sold more than 3 items in the last 365 days. Who are the top three shoppers in these popular shops in terms of the number of items they have purchased?](#_b9uh4i7uvlox) 19

[**Printout of table records:**](#_e4gl456zyu2x) **21**

[**APPENDIX C: INDIVIDUAL CONTRIBUTION FORM**](#_1dgbhj51k8oi) **86**

# 

# 

# **Table creation commands**

| **CREATE** **TABLE** USERS(  UID INT **IDENTITY**(1,1),  UName varchar(255),  PRIMARY **KEY** (UID), );  **CREATE** **TABLE** PRODUCTS (  PName varchar(255),  Maker varchar(255),  **Category** varchar(255),   PRIMARY **KEY** (PName) );  **CREATE** **TABLE** SHOPS(  SName varchar(255),    PRIMARY **KEY** (SName), );  **CREATE** **TABLE** EMPLOYEE(  EmployeeID INT **IDENTITY**(1,1),  EName varchar(255),  Salary **double** **PRECISION**,    PRIMARY **KEY** (EmployeeID), );  **CREATE** **TABLE** ORDERS(  OID INT **IDENTITY**(1,1),  UID INT,  ODate\_time DATE,  Shipping\_address varchar(255),    PRIMARY **KEY** (OID),    FOREIGN **KEY** (UID) **REFERENCES** **USERS**(UID) **ON** **UPDATE** **CASCADE** );  **CREATE** **TABLE** COMPLAINTS(  CID INT **IDENTITY**(1,1),  UID INT,  CText varchar(255),  CStatus varchar(15) **NOT** NULL **CHECK** (CStatus **IN** ('pending', 'being handled', 'addressed')),  Filed\_DATE DATE,   PRIMARY **KEY** (CID),  FOREIGN **KEY** (UID) **REFERENCES** **Users**(UID) **ON** **UPDATE** **CASCADE** );  **CREATE** **TABLE** PRICE\_HISTORY (  PName varchar(255),  SName varchar(255),  StartDate DATE,  EndDate DATE,  Price **DOUBLE** **PRECISION**,   PRIMARY **KEY** (PName, SName, StartDate, EndDate),  FOREIGN **KEY** (PName) **REFERENCES** PRODUCTS(PName) **ON** **UPDATE** **CASCADE**,  FOREIGN **KEY** (SName) **REFERENCES** SHOPS(SName) **ON** **UPDATE** **CASCADE** );  **CREATE** **TABLE** PRODUCTS\_IN\_ORDERS(  PName varchar(255),  SName varchar(255),  OID INT,  OPID INT,  OPrice **DOUBLE** **PRECISION**,  OQuantity INT,  Delivery\_date DATE,  OStatus varchar(10) **NOT** NULL **CHECK** (OStatus **IN** ('Shipped', 'Delivered', 'Returned')),   PRIMARY **KEY** (PName, SName, OID),  FOREIGN **KEY** (PName) **REFERENCES** PRODUCTS(PName) **ON** **UPDATE** **CASCADE**,  FOREIGN **KEY** (SName) **REFERENCES** SHOPS(SName) **ON** **UPDATE** **CASCADE**,  FOREIGN **KEY** (OID) **REFERENCES** ORDERS(OID) **ON** **UPDATE** **CASCADE** );  **CREATE** **TABLE** PRODUCTS\_IN\_SHOPS\_R2(  PName varchar(255),  SName varchar(255),  SPID varchar(255),   **CONSTRAINT** USPID **UNIQUE** (SPID),    PRIMARY **KEY** (PName, SName),  FOREIGN **KEY** (PName) **REFERENCES** PRODUCTS(PName) **ON** **UPDATE** **CASCADE**,  FOREIGN **KEY** (SName) **REFERENCES** SHOPS(SName) **ON** **UPDATE** **CASCADE** );  **CREATE** **TABLE** PRODUCTS\_IN\_SHOPS\_R1(  SPID varchar(255),  SPrice **DOUBLE** **PRECISION**,  SQuantity INT,    PRIMARY **KEY** (SPID),  FOREIGN **KEY** (SPID) **REFERENCES** PRODUCTS\_IN\_SHOPS\_R2(SPID) **ON** **UPDATE** **CASCADE**, );  **CREATE** **TABLE** COMPLAINTS\_ON\_SHOPS(  CID INT,  SName varchar(255),    PRIMARY **KEY** (CID),    FOREIGN **KEY** (CID) **REFERENCES** COMPLAINTS(CID),  FOREIGN **KEY** (SName) **REFERENCES** SHOPS(SName), );  **CREATE** **TABLE** COMPLAINTS\_ON\_ORDERS(  CID INT,  OID INT,    PRIMARY **KEY** (CID),    FOREIGN **KEY** (CID) **REFERENCES** COMPLAINTS(CID),  FOREIGN **KEY** (OID) **REFERENCES** ORDERS(OID), );  **CREATE** **TABLE** FEEDBACK\_R2(  UID INT,  OID INT,    PRIMARY **KEY** (OID),    FOREIGN **KEY** (UID) **REFERENCES** USERS(UID),  FOREIGN **KEY** (OID) **REFERENCES** ORDERS(OID), );  **CREATE** **TABLE** FEEDBACK\_R1(  PName varchar(255),  SName varchar(255),  OID INT,  FDate\_time DATE,  Rating INT **NOT** NULL **CHECK** (Rating **IN** (1,2,3,4,5)),  Comment varchar(255),    PRIMARY **KEY** (PName,SName,OID),    FOREIGN **KEY** (PName) **REFERENCES** PRODUCTS(PName),  FOREIGN **KEY** (SName) **REFERENCES** SHOPS(SName),  FOREIGN **KEY** (OID) **REFERENCES** FEEDBACK\_R2(OID) );  **CREATE** **TABLE** HANDLED(  CID INT,  handled\_DATE DATE,  EmployeeID INT,   PRIMARY **KEY** (CID),    FOREIGN **KEY** (CID) **REFERENCES** COMPLAINTS(CID),  FOREIGN **KEY** (EmployeeID) **REFERENCES** EMPLOYEE(EmployeeID) ); |
| --- |
|  |

# 

# 

# 

# **Queries**

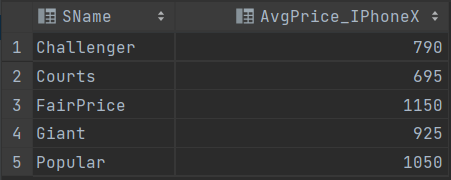
## **Query 1:**

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

**SQL Query:**

| **SELECT** p1.SName,**AVG**(p1.Price) **as** AvgPrice\_IPhoneX **FROM** PRICE\_HISTORY p1 **WHERE** ((p1.EndDate >= '2021-08-01' **AND** p1.EndDate <= '2021-08-31')  **OR** (p1.StartDate >= '2021-08-01' **AND** p1.StartDate <= '2021-08-31')  **OR** (p1.StartDate < '2021-08-01' **AND** p1.EndDate > '2021-08-31'))  **AND** p1.PName **LIKE** 'Apple iPhone X' **GROUP** **BY** p1.SName; |
| --- |

**Output:**



**Explanation:**

The price of IPhone X in August 2021 can be found in price history under these conditions:

(i) Price change start before August 2021 and end at August 2021.

(ii) Price change start during August 2021 and end after August 2021.

(iii) Price change start during August 2021 and end at August 2021.

(iv) Price change start before August 2021 and end after August 2021.

(p1.EndDate >= '2021-08-01' AND p1.EndDate <= '2021-08-31') cover condition (i).

(p1.StartDate >= '2021-08-01' AND p1.StartDate <= '2021-08-31') cover condition (ii).

These two conditions together will cover condition (iii).

Lastly, (p1.StartDate < '2021-08-01' AND p1.EndDate > '2021-08-31') cover condition (iv).

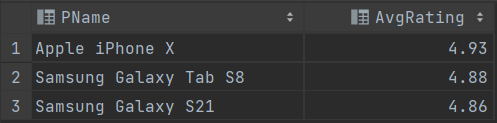
## Query 2:

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

**SQL Query:**

| **SELECT** PName,**ROUND**(**AVG**(**CAST**(Rating **AS** FLOAT)),2) **AS** AvgRating **FROM** FEEDBACK\_R1 **WHERE** PName **IN** (**SELECT** PName  **FROM** FEEDBACK\_R1  **WHERE** [FDate\_time] >= '2021-08-01'   **AND** [FDate\_time] < '2021-09-01'   **GROUP** **BY** PName, Rating  **HAVING** Rating = 5 **AND** **COUNT**(\*) >= 100) **GROUP** **BY** PName **ORDER** **BY** AvgRating **DESC** |
| --- |

**Output:**



**Explanation:**

| **SELECT** PName **FROM** FEEDBACK\_R1 **WHERE** [FDate\_time] >= '2021-08-01' **AND** [FDate\_time] < '2021-09-01'  **GROUP** **BY** PName, Rating  **HAVING** Rating = 5 **AND** **COUNT**(\*) >= 100 |
| --- |

The subquery above first finds those products that have at least 100 ratings of 5 in August 2021.

Next, we group by PName and find the average rating of each product. Lastly, we order the products based on the average ratings.

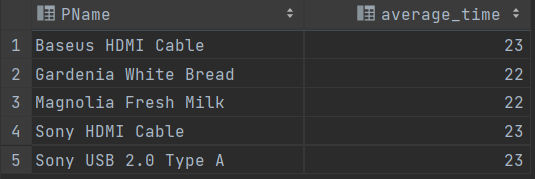
## Query 3:

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

**SQL Query:**

| **SELECT** PO.PName, **AVG**(**DATEDIFF**(**day**, O.ODate\_time, PO.Delivery\_date)) **as** average\_time **FROM** PRODUCTS\_IN\_ORDERS **AS** PO, ORDERS **AS** O **WHERE** PO.Ostatus = 'delivered'   **AND** (O.ODate\_time >= '2021-06-01' **AND** O.ODate\_time <= '2021-06-30')  **AND** PO.OID = O.OID **GROUP** **BY** PO.PName; |
| --- |

**Output:**



**Explanation:**

The ORDERS table is joined with PRODUCTS\_IN\_ORDERS to get the delivery date and the order date for a particular order. We then check if the product is ordered in June 2021 and check if it has been delivered. Then, we group it by PName and find the average difference between the order date and delivery date.

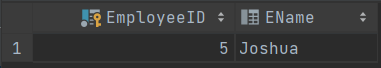
## Query 4:

## 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.

**SQL Query:**

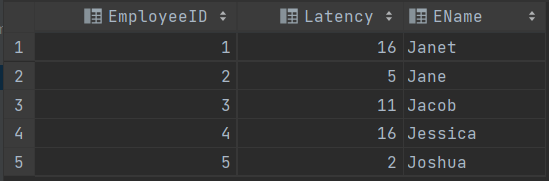
| WITH A AS (**SELECT** E.EmployeeID, **AVG**(**DATEDIFF**(**day**, C.Filed\_DATE, H.handled\_DATE)) **AS** Latency, E.EName  **FROM** COMPLAINTS **AS** C, EMPLOYEE **AS** E, HANDLED **AS** H  **WHERE** C.CID = H.CID **AND** E.EmployeeID = H.EmployeeID  **GROUP** **BY** E.EmployeeID, E.EName)  **SELECT** A.EmployeeID, A.EName **FROM** A **WHERE** Latency = (**SELECT** **MIN**(Latency) **FROM** A) |
| --- |

**Output:**



**Explanation:**

We first do a join on COMPLAINTS, EMPLOYEE and HANDLED tables. From this table, we group by the employee and find the latency which is the average of the date the complaint is made and the date the complaint is handled.



Lastly, we can just select the record with the lowest latency.

## 

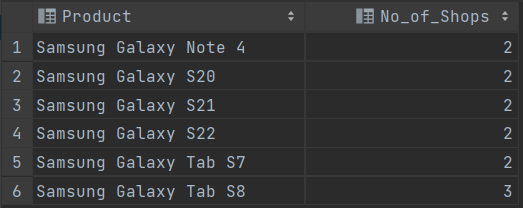
## Query 5:

## 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.

**SQL Query:**

| **SELECT** PRODUCTS.PName **AS** Product, **COUNT**(PRODUCTS\_IN\_SHOPS\_R2.PName) **AS** No\_of\_Shops **FROM** PRODUCTS **LEFT** **JOIN** PRODUCTS\_IN\_SHOPS\_R2 **ON** PRODUCTS.PName = PRODUCTS\_IN\_SHOPS\_R2.PName **WHERE** maker = 'Samsung' **GROUP** **BY** PRODUCTS.PName |
| --- |

**Output:**



**Explanation:**

PRODUCTS table is left joined with PRODUCTS\_IN\_SHOPS\_R2 based on PName.

Next, the where clause will retrieve products whose maker is ‘Samsung’.

Lastly, we group by PName and count the number of times the record appears in PRODUCTS\_IN\_SHOPS\_R2 which will give us the number of shops that sell the product on Shiokee.

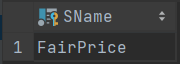
## Query 6:

## Find shops that made the most revenue in August 2021.

**SQL Query:**

| **SELECT** TOP 1 SName **FROM** PRODUCTS\_IN\_ORDERS **AS** PO **JOIN** ORDERS **AS** O **ON** O.OID = PO.OID **WHERE** **MONTH**(ODate\_time) = 8 **AND** **YEAR** (ODate\_time) = 2021 **GROUP** **BY** PO.SName **ORDER** **BY** **SUM**(OPrice \* OQuantity) **DESC** |
| --- |

**Output:**



**Explanation:**

First, PRODUCTS\_IN\_ORDERS was joined with ORDERS based on the OID.

The where clause allows us to find orders made in August 2021.

Lastly, we group by SName and sort in descending order of SUM(OPrice \* OQuantity) which is the revenue that the shops had made in these PRODUCTS\_IN\_ORDERS. The first tuple will be the shop that made the most revenue.

## 

## 

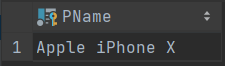
## Query 7:

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

**SQL Query:**

| **SELECT** TOP 1 PName **FROM** PRODUCTS\_IN\_ORDERS  **INNER** **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID **WHERE** UID = (**SELECT** TOP 1 UID  **FROM** COMPLAINTS   **GROUP** **BY** UID  **ORDER** **BY** **COUNT**(CID) **DESC**) **ORDER** **BY** OPrice **DESC** |
| --- |

**Output:**



**Explanation:**

The PRODUCTS\_IN\_ORDERS table is joined with ORDERS to get all the products that the user with the most complaints has purchased.

| **SELECT** TOP 1 UID **FROM** COMPLAINTS  **GROUP BY** UID **ORDER BY COUNT(**CID**) DESC** |
| --- |

The subquery above will find UID that made the most complaints. It first groups the complaints by UID and sorts them in descending order based on the total complaints that the user had made.

Lastly, the table is sorted in descending order by the price of the products that he/she had purchased and the first tuple is the most expensive product that the user had purchased.

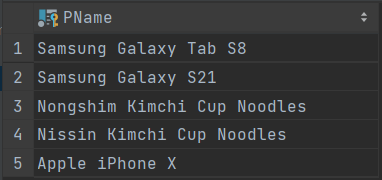
## Query 8:

## Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2021.

**SQL Query:**

| **SELECT** TOP 5 PName **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS O **on** PRODUCTS\_IN\_ORDERS.OID = O.OID **WHERE** PName **IN** (**SELECT** PName  **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID  **GROUP** **BY** PName  **HAVING** **COUNT**(**DISTINCT** UID) < (**SELECT** **COUNT**(UID) **FROM** **USERS**))  **AND** **MONTH**(ODate\_time) = 8  **AND** **YEAR**(ODate\_time) = 2021 **GROUP** **BY** PName **ORDER** **BY** **SUM**(OQuantity) **DESC** |
| --- |

**Output:**



**Explanation:**

| **SELECT** PName **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID **GROUP** **BY** PName **HAVING** **COUNT**(**DISTINCT** UID) < (**SELECT** **COUNT**(UID) **FROM** **USERS**) |
| --- |

The above subquery finds the products that have not been purchased by all users. The PRODUCTS\_IN\_ORDERS table is joined with ORDERS to get all the users who had purchased the products.

Within the outer query, we filter the tuples where the order was made in August and then we then group using PName and sort in descending order by the sum of the quantity that has been sold. The first 5 tuples are the top 5 most purchased products that have never been purchased by all users.

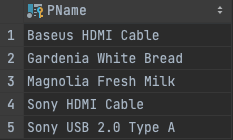
## Query 9:

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

**SQL Query:**

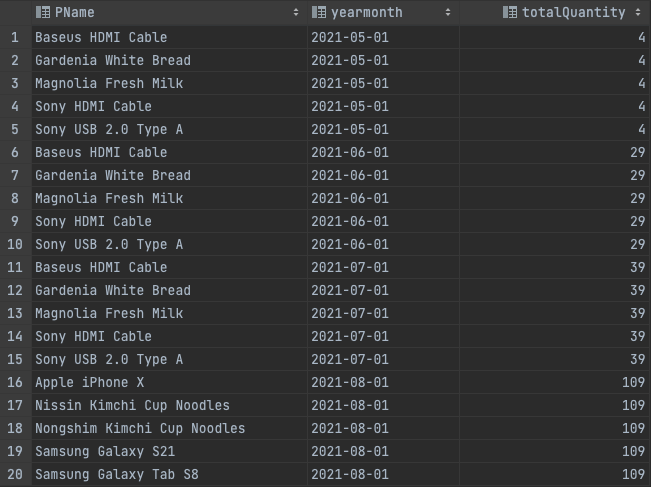
| WITH A AS (**SELECT** PName, **DATEFROMPARTS**(**YEAR**(ODate\_time), **MONTH**(ODate\_time), 1) **AS** yearmonth, **SUM**(OQuantity) **AS** totalQuantity  **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID  **GROUP** **BY** PName, **DATEFROMPARTS**(**YEAR**(ODate\_time), **MONTH**(ODate\_time), 1) )  **SELECT** A.PName **FROM** A, A **AS** B, A **AS** C **WHERE** (  A.PName = B.PName **AND**   B.PName = C.PName **AND**   **DATEDIFF**(**month**, A.yearmonth, B.yearmonth) = 1 **AND**   **DATEDIFF**(**month**, B.yearmonth, C.yearmonth) = 1 **AND**  A.totalQuantity <= B.totalQuantity **AND**  B.totalQuantity <= C.totalQuantity ) |
| --- |

**Output:**



**Explanation:**

The PRODUCTS\_IN\_ORDERS table is joined with the ORDERS table to get the date that the products have been purchased.The new table is grouped by month and year of the order date:



| **SELECT** A.PName **FROM** A, A **AS** B, A **AS** C **WHERE** (  A.PName = B.PName **AND**   B.PName = C.PName **AND**   **DATEDIFF**(**month**, A.yearmonth, B.yearmonth) = 1 **AND**   **DATEDIFF**(**month**, B.yearmonth, C.yearmonth) = 1 **AND**  A.totalQuantity <= B.totalQuantity **AND**  B.totalQuantity <= C.totalQuantity ) |
| --- |

The above statement is to find the products that are increasingly being purchased over the months.

## Additional Query 1:

## Frequent shoppers are shoppers who have purchased more than 2 items per shop for at least 2 shops in the last 730 days. Who are the top 3 frequent shoppers in terms of the total cost of the items they have purchased?

Note: Originally “at least 5 shops in the last 30 days.”

**SQL Query:**

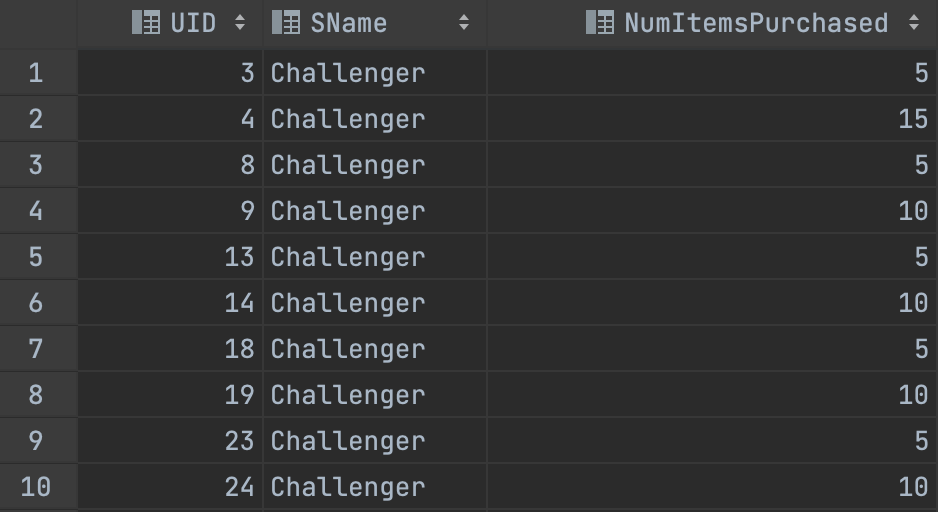
| WITH A AS (   **SELECT** UID, SName, **SUM**(OQuantity) **AS** NumItemsPurchased  **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **on** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID  **WHERE** **DATEDIFF**(**day**, ODate\_time, **GETDATE**()) < 730  **GROUP** **BY** UID, SName  **HAVING** **SUM**(OQuantity) >= 2 ), B **AS** (  **SELECT** UID, **COUNT**(SName) **AS** ShopsVisited  **FROM** A  **GROUP** **BY** UID  **HAVING** **COUNT**(SName) >= 2 )  **SELECT** TOP 3 UID **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID **WHERE** UID **IN** (**SELECT** UID **FROM** B) **GROUP** **BY** UID **ORDER** **BY** **SUM**(OQuantity \* OPrice) |
| --- |
|  |

**Output:**



**Explanation:**

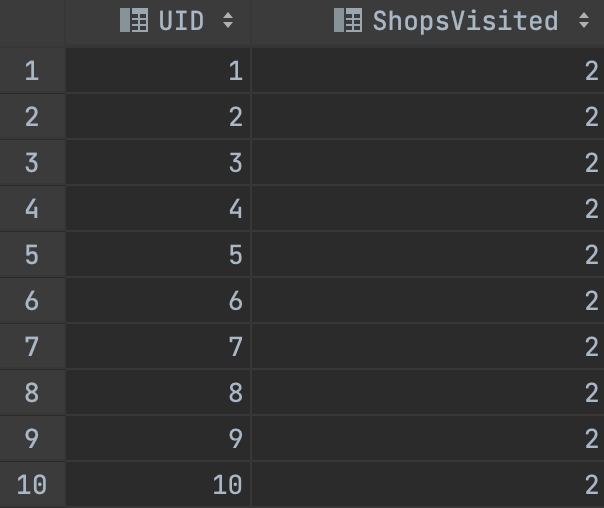
Table A: (only the first 10 records are shown)



**WHERE DATEDIFF**(**day**, ODate\_time, **GETDATE**()) < 730 gives us the queries where the difference between ODate\_time and current date is less than 730.

The PRODUCTS\_IN\_ORDERS table is joined with the ORDERS table to get the UID of users who have purchased at least 2 items in the same shop in the last 730 days.

Table B: (only the first 10 records are shown)



Next, we select the users from table A that have bought items from at least 2 shops.

Lastly, we find the users who meet the previous requirement and the total cost of items they have purchased is the top 3 in the users.

## 

## Additional Query 2:

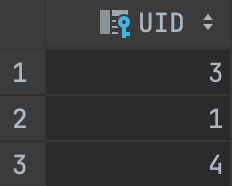
## Popular shops are shops which have sold more than 3 items in the last 365 days. Who are the top three shoppers in these popular shops in terms of the number of items they have purchased?

Note: Originally “shops which have sold more than 3 items in the last 30 days“

**SQL Query:**

| WITH A AS (  **SELECT** SName  **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID  **WHERE** **DATEDIFF**(**day**, ODate\_time, **GETDATE**()) < 365  **GROUP** **BY** SName  **HAVING** **SUM**(OQuantity) > 3 )  **SELECT** TOP 3 UID **FROM** PRODUCTS\_IN\_ORDERS **JOIN** ORDERS **ON** PRODUCTS\_IN\_ORDERS.OID = ORDERS.OID **WHERE** SName **IN** (**SELECT** SName **FROM** A) **GROUP** **BY** UID **ORDER** **BY** **SUM**(OQuantity) **DESC** |
| --- |

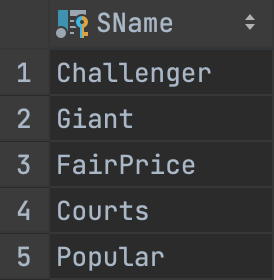
**Output:**



**Explanation:**

PRODUCTS\_IN\_ORDERS is first joined with ORDERS table based on the OID and we filter them based on the date whose orders were made in the last 730 days. Then we group by SName and selecting those with COUNT(OQuantity) > 3 will allow us to find popular shops. This will yield table A.

Table A:

****

Then we join the tables PRODUCTS\_IN\_ORDERS with ORDERS again, and we filter the shops in the joined table using the shops in table A with the IN statement. This gives us shoppers who shopped in popular shops. Lastly, we group by UID and order by the quantity the user purchased and select the top 3 users based on the quantity of items purchased.

# 

# **Printout of table records:**

# 

# 

# **APPENDIX C: INDIVIDUAL CONTRIBUTION FORM**

| Name | Individual Contribution to Submission 3 (Lab 5) | Percentage of Contribution | Signature |
| --- | --- | --- | --- |
| Eugene Poh Yang Quan | * Data generation * Table creation * Queries | 16.6 |  |
| Wong Yi Pun | * Data generation * Table creation * Queries | 16.6 |  |
| Roy Lau Run-Xuan | * Data generation * Table creation * Queries | 16.6 |  |
| Chua Zi Jian | * Data generation * Table creation * Queries | 16.6 |  |
| Ryan | * Data generation * Table creation * Queries | 16.6 |  |
| Koh Jun Kai | * Data generation * Table creation * Queries | 16.6 |  |