CZ2007-Introduction To Databases

Lab 5 Report

DSS2 Group 1

Name	Individual Contribution to Submission 3 (Lab 5)	Percentage of Contribution	Signature
Clement	Lab 5 Deliverables	14.3	
Nihal	Lab 5 Deliverables	14.3	
Samarth	Lab 5 Deliverables	14.3	
Kelly Wong	Lab 5 Deliverables	14.3	
Cao Qingtian	Lab 5 Deliverables	14.3	
Huang Jingfang	Lab 5 Deliverables	14.3	
Lin Jacky	Lab 5 Deliverables	14.3	

Table Creation

Price History

```
CREATE TABLE PRICE_HISTORY(
    SName [NVARCHAR](50) NOT NULL REFERENCES SHOPS(SName),
    PName [NVARCHAR](50) NOT NULL REFERENCES PRODUCTS(PName),
    START_DATE DATETIME NOT NULL CHECK(START_DATE <= GETDATE()),
    END_DATE DATETIME DEFAULT GETDATE(),
    Price FLOAT NOT NULL CHECK (Price >=0)
    PRIMARY KEY(SName, PName, START_DATE, END_DATE)
);
```

Products_In_Orders

```
CREATE TABLE PRODUCTS_IN_ORDERS(
     PName [NVARCHAR](50) REFERENCES PRODUCTS (PName) ON DELETE
CASCADE ON UPDATE CASCADE,
     SName [NVARCHAR](50) REFERENCES SHOPS(SName) ON DELETE CASCADE ON
UPDATE CASCADE,
     OPID INT NOT NULL,
     OID INT NOT NULL REFERENCES ORDERS(OID) ON UPDATE CASCADE,
     OPrice FLOAT NOT NULL CHECK (OPrice >= 0),
     OQuantity INT NOT NULL CHECK(OQuantity >= 0),
     Delivery_date DATETIME DEFAULT GETDATE(),
PRIMARY KEY (PName, SName, OID),
UNIQUE (OID, OPID),
Status varchar(50) CHECK (Status = 'Delivered' OR Status = 'Being
Processed' OR Status = 'Shipped' OR Status = 'Returned')
);
```

Feedback

```
CREATE TABLE FEEDBACK(
     UID INT NOT NULL REFERENCES USERS (UID) ON DELETE CASCADE ON
UPDATE CASCADE,
     PName [NVARCHAR](50) REFERENCES PRODUCTS(PName) ON DELETE CASCADE
ON UPDATE CASCADE,
     SName [NVARCHAR](50) REFERENCES SHOPS(SName) ON DELETE CASCADE ON
UPDATE CASCADE,
```

```
      Rating INT NOT NULL CHECK(Rating >= 1 AND Rating <=5),
      Comment varchar(max) NOT NULL,
      DATE_TIME DATETIME NOT NULL CHECK (DATE_TIME <= GETDATE())
      PRIMARY KEY (UID, PName, SName)
);
```

Products_In_Shops

```
CREATE TABLE PRODUCTS_IN_SHOPS(
    PName [NVARCHAR](50)  NOT NULL REFERENCES PRODUCTS(PName) ON
DELETE CASCADE ON UPDATE CASCADE,
    SPID INT NOT NULL,
    SPrice FLOAT NOT NULL CHECK (SPrice >= 0),
    SQuantity INT NOT NULL CHECK (SQuantity >=0),
    SName [NVARCHAR](50)  NOT NULL REFERENCES SHOPS(SName) ON DELETE
CASCADE ON UPDATE CASCADE
    PRIMARY KEY (PName, SName),
    UNIQUE (SName, SPID)
);
```

Complaints

```
CREATE TABLE COMPLAINTS(
    CID INT NOT NULL PRIMARY KEY,
    Text varchar(max) NOT NULL,
    FILED_DATE_TIME DATETIME NOT NULL,
    Status varchar(50) CHECK (Status = 'Pending' OR Status = 'Being
Handled' OR Status = 'Addressed'),
    UID INT NOT NULL REFERENCES USERS(UID) ON DELETE CASCADE ON UPDATE
CASCADE,
     EmployeeID INT REFERENCES EMPLOYEES(EmployeeID) ON DELETE SET
NULL ON UPDATE CASCADE,
     HANDLED_DATE_TIME DATETIME DEFAULT NULL,
    UNIQUE (UID, FILED_DATE_TIME)
);
```

Complaints_On_Orders

```
CREATE TABLE COMPLAINTS_ON_ORDERS(
    CID INT NOT NULL PRIMARY KEY,
```

```
    OID INT NOT NULL REFERENCES ORDERS(OID),
    FOREIGN KEY (CID) REFERENCES COMPLAINTS(CID) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

Products

```
CREATE TABLE PRODUCTS(
      PName [NVARCHAR](50) NOT NULL PRIMARY KEY,
      Maker [NVARCHAR](50) NOT NULL,
      Category [NVARCHAR](50) NOT NULL
);
```

Orders

```
CREATE TABLE ORDERS(
    OID INT NOT NULL PRIMARY KEY,
    Date_time DATETIME NOT NULL DEFAULT GETDATE(),
    Shipping_address varchar(max) NOT NULL,
    UID INT REFERENCES USERS(UID) ON DELETE CASCADE ON UPDATE CASCADE
    UNIQUE (Date_time, UID)
);
```

Shops

```
CREATE TABLE SHOPS(
   SName [NVARCHAR](50) NOT NULL PRIMARY KEY
);
```

Users

```
CREATE TABLE USERS(
      UID INT NOT NULL PRIMARY KEY,
      UName [NVARCHAR](50) NOT NULL
);
```

Complaints_On_Shops

```
CREATE TABLE COMPLAINTS_ON_SHOPS(
    CID INT NOT NULL PRIMARY KEY,
```

```
     SName [NVARCHAR](50)  REFERENCES SHOPS(SName) ON DELETE SET NULL
ON UPDATE CASCADE,
     FOREIGN KEY (CID) REFERENCES COMPLAINTS(CID) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

Employees

```
CREATE TABLE EMPLOYEES
(
    EmployeeID INT NOT NULL PRIMARY KEY,
    Name [NVARCHAR](50) NOT NULL,
    Salary FLOAT NOT NULL DEFAULT 0.0 CHECK (Salary>=0)
)
```

TRIGGERS

Feedback

```
CREATE TRIGGER FeedbackCheck

ON FEEDBACK

AFTER INSERT

AS



IF (EXISTS(SELECT * FROM inserted, PRODUCTS_IN_ORDERS, ORDERS



WHERE

inserted.SName=PRODUCTS_IN_ORDERS.SName

AND inserted.PName=PRODUCTS_IN_ORDERS.PName

AND PRODUCTS_IN_ORDERS.OID=ORDERS.OID
```

```
AND PRODUCTS_IN_ORDERS.Status = 'Delivered'OR PRODUCTS_IN_ORDERS.Status =
'Returned'

AND inserted.UID = ORDERS.UID

))



ROLLBACK

    RAISERROR('You must purchase this product first before rating
it.',10,1);
```

Description: this disallows users to rate the products before they get it. When an insertion like this occurs, the table rejects it and rolls back to the previous version.

CONSTRAINTS

ALTER TABLE COMPLAINTS

ADD CONSTRAINT check_date

CHECK (FILED_DATE_TIME<= GETDATE())

ALTER TABLE PRICE_HISTORY

ADD CONSTRAINT checkStartnEnd

CHECK (START_DATE <= END_DATE)

Queries

Query 1

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

```
SELECT AVG(Price) AS AveragePrice

FROM PRICE_HISTORY

WHERE PName = 'iPhone Xs'

AND Start_date <= convert(DATETIME,'2021/08/31', 111)

AND End_date >= convert(DATETIME,'2021/08/01', 111)
```

Output:

	AveragePrice
1	3389.30

Description: We selected all products that were named “iPhone Xs” in our price_history table. The start_date and end_date were converted to the correct format such that a comparison between dates can be performed. Lastly, we used the aggregate function average to get the average price of the device, with the various conditions applied.

Video File:
https://drive.google.com/file/d/1ydA0kVskoYkNsqLrHlM_eLRh8Wur2FK8/view?usp=sharing

Query 2

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

```
-- temporary table which stores product name with

-- more than 100 ratings of "5"

SELECT PName INTO T1
```

```
FROM FEEDBACK

WHERE Rating = 5 AND MONTH(DATE_TIME) = 8  AND YEAR(DATE_TIME) = 2021

GROUP BY PName

HAVING COUNT(Rating) >= 100;

-- find average ratings for each product

SELECT PName,ROUND(AVG(CAST(Rating as float)),2) AS AvgRating

FROM FEEDBACK

WHERE PName IN (SELECT * FROM T1) AND MONTH(DATE_TIME) = 8 AND
YEAR(DATE_TIME) = 2021

GROUP BY PName

ORDER BY AVG(Rating) DESC;
```

Output:

	PName	AvgRating
1	Galaxy S	5
2	Galaxy A	4.98

Description: A temporary table T1 is created to store all the product names that have received more than 100 ratings of “5”. Then we use aggregation method again to calculate the average ratings for each product that is in T1.

Video File:
https://drive.google.com/file/d/1IbGcPyX8QO7_BHlcEiWKqZqhUWYT0WnH/view?usp=sharing

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.

```
SELECT AVG(ABS(DATEDIFF(hour,Date_Time,Delivery_date))) AS
Avg_Delivery_Time_In_Hours
```

```
FROM PRODUCTS p, ORDERS o, PRODUCTS_IN_ORDERS po

WHERE po.PName = p.PName

AND po.OID = o.OID

AND o.Date_time >='2021-06-01 00:00:01'

AND o.Date_time <= '2021-06-30 23:59:59'

AND (po.Status ='Delivered' OR po.Status='Returned');
```

Output:

	Avg_Delivery_Time_In_Hours
1	47

Description: We assume that the question wants to find the average delivery time, which we quantify in hours, for all the products bought in the month of June. We query the required result using the aggregate function AVG() and the DATEDIFF() function to find the average time between ordered date time and delivered timestamp for products ordered in June only.

Video File:
https://drive.google.com/file/d/1IGTLZ49WNrS05-RTN-vrXggVw7ATEv32/view?usp=sharing

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.

```
SELECT Minimum.minLatency, Average.EmployeeID
FROM(
    SELECT MIN(avg_duration) AS minLatency
            FROM (
                SELECT EmployeeID, AVG(DATEDIFF(MINUTE, FILED_DATE_TIME,
HANDLED_DATE_TIME)) AS avg_duration
                FROM COMPLAINTS
                WHERE [Status]='Addressed'
                GROUP BY EmployeeID
```

```
            )AS Average
)AS Minimum
JOIN  (SELECT EmployeeID, AVG(DATEDIFF(MINUTE, FILED_DATE_TIME,
HANDLED_DATE_TIME)) AS avg_duration
                FROM COMPLAINTS
                WHERE [Status]='Addressed'
                GROUP BY EmployeeID
                )AS Average
ON Minimum.minLatency = Average.avg_duration
```

Output:

minLatency	EmployeeID
1440	6

Description: Using subqueries and aggregation find MIN of avg_duration. Average is found of the date difference between Filled and Handled date time using DATEDIFF and the Aggregator: AVG from Complaints. We also group by employee ID and the join table to a table with the averages of the respective employees. The employee whose latency matches the min from the attribute avg_duration is projected.

Video File:
https://drive.google.com/file/d/1ISAErEZq2QcwGUt0oG0jrYMENe1k0xtn/view?usp=sharing

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.

```
-- (i) a list that contains all products made by Samsung
-- Selecting Products where Maker='Samsung'
SELECT s.PName
FROM (
   SELECT PName, Maker FROM PRODUCTS WHERE Maker='Samsung'
   ) AS s
-- (ii) for each of them, the number of shops on Shiokee that sell the
product
SELECT PName, numberShops FROM(
 SELECT PName, Count(*) AS numberShops
   FROM PRODUCTS_IN_SHOPS
   GROUP BY PName
```

```
 ) AS p
WHERE PName IN (
 SELECT s.PName
 FROM (
   SELECT PName, Maker FROM PRODUCTS WHERE Maker='Samsung'
 ) AS s)
```

Output:

(i)

	PName
1	Galaxy A
2	Galaxy Fit
3	Galaxy M
4	Galaxy Note
5	Galaxy S
6	Galaxy S22
7	Galaxy S7
8	Galaxy S9
9	Galaxy Tab A
10	Galaxy Tab S8
11	Galaxy Watch4
12	Galaxy Z
13	Samsung S21
14	Samsung S22

(ii)

	PName	numberShops
1	Galaxy A	1
2	Galaxy S22	3
3	Samsung S21	3

4	Samsung S22	4

Description: (i) Selecting Product names (PName) from the PRODUCTS table where Maker of the product is 'Samsung'.
(ii) For each product in PRODUCTS_IN_SHOPS, use groupby to get the number of shops with the product. Print the rows of PName and numberShops only If the name of product is in the list of product names (PName) which are made by Samsung.

Video File:
https://drive.google.com/file/d/1Ae9MuMn22Yq13iUKW-S8zgwolMBXhjAs/view?usp=sharing

Query 6

Find shops that made the most revenue in August 2021.

```
SELECT TOP 3 SName, SUM(OPrice*OQuantity) AS TotalSales FROM
PRODUCTS_IN_ORDERS
WHERE YEAR(Delivery_date) = 2021 AND MONTH(Delivery_date) = 8
GROUP BY SName
ORDER BY SUM(OPrice*OQuantity) DESC
```

Output:

MegaStore	7351
Shiokee 1	6057.75
Galaxy	3496.9

Description: The revenue for a shop was calculated by multiplying the price and quantity in the orders, and summing it up with the aggregate function SUM. The revenue was renamed as TotalSales and was grouped together by the name of the shops. It was then ordered in descending order, and the top 3 shops were the output for this query.

Video File:
https://drive.google.com/file/d/1dwPbUlgV3YQ-cd802z5ROyqJgfwreRNt/view?usp=sharing

Query 7

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

```
SELECT Totalprice.PName, Totalprice.OPrice, Totalprice.UID
FROM (
    SELECT MAX(OPrice) as maxPrice
        FROM(
            SELECT MAX(totalComplaints) as maxComplaints
                FROM (
                    SELECT UID, COUNT(FILED_DATE_TIME) as totalComplaints
                    FROM COMPLAINTS
                    GROUP BY UID
                )AS Total
        )AS Maximum
    JOIN  (SELECT UID, COUNT(FILED_DATE_TIME) as totalComplaints
                FROM COMPLAINTS
                GROUP BY UID
                )AS Total
    ON Total.totalComplaints =  Maximum.maxComplaints
    JOIN ORDERS o
    ON o.UID = Total.UID
    JOIN PRODUCTS_IN_ORDERS pod
    ON o.OID = pod.OID
) AS MaxPrice
JOIN (
    SELECT pod.Pname, pod.OPrice, Total.UID
        FROM(
            SELECT MAX(totalComplaints) as maxComplaints
                FROM (
                    SELECT UID, COUNT(FILED_DATE_TIME) as totalComplaints
                    FROM COMPLAINTS
                    GROUP BY UID
                )AS Total
            )AS Maximum
        JOIN  (SELECT UID, COUNT(FILED_DATE_TIME) as totalComplaints
                FROM COMPLAINTS
                GROUP BY UID
            )AS Total
    ON Total.totalComplaints =  Maximum.maxComplaints
```

```
    JOIN ORDERS o
    ON o.UID = Total.UID
    JOIN PRODUCTS_IN_ORDERS pod
    ON o.OID = pod.OID
) AS TotalPrice
ON MaxPrice.maxPrice = TotalPrice.OPrice
```

Output:

PName	OPrice	UID
iPhone 11	1129	2

Description: Join the count of the total complaints grouped by UID to the one with the max complaints, using the table COMPLAINTS on the condition that the total complaint from that UID == MAX of the COUNT of complaints from all the UID. Join orders from the table ORDERS where UID == that of the person who made the most complaints. Join the table with the orders made by the user who made the max complaints to PRODUCTS_IN_ORDERS to find the price. Finally, find the most expensive product (using the aggregation MAX) purchased by that UID, project their PName, OPrice and UID.

Video File:
https://drive.google.com/file/d/19p9la-U7E5W0TeWYcUKihNzLCyErXnv0/view?usp=sharing

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.

```
CREATE VIEW purchaseRecords AS
SELECT ORDERS.UID AS UID, PRODUCTS_IN_ORDERS.Pname AS Pname
FROM ORDERS,PRODUCTS_IN_ORDERS
WHERE PRODUCTS_IN_ORDERS.Pname = PRODUCTS_IN_ORDERS.Pname
AND PRODUCTS_IN_ORDERS.OID = ORDERS.OID
```

```
SELECT top 5 notpurchased.PName AS Product,
sum(PRODUCTS_IN_ORDERS.OQuantity) Total
FROM
```

```
     (
          SELECT DISTINCT PName
          FROM USERS, purchaseRecords
          WHERE
          -- there are users who have not brought this product
          EXISTS
               (SELECT USERS.UID FROM USERS WHERE USERS.UID NOT IN (SELECT
purchaseRecords.UID FROM purchaseRecords))
     )AS notpurchased,
     PRODUCTS_IN_ORDERS
-- Xref the products that not all have brought it with products in
PRODUCTS_IN_ORDERS
-- and rank based on sum of Quantity sold
WHERE notpurchased.PName=PRODUCTS_IN_ORDERS.PName
GROUP BY notpurchased.PName
ORDER BY Total DESC
```

Output:

	Product	Total
1	iPhone Xs	10
2	iPhone 7	10
3	Samsung S22	10
4	Samsung S21	9
5	iPhone 11	9

Description: Firstly, we created a view to collect all the purchased history, in the format of 'UID' and 'Pname', so this table contains the information of which user has purchased which product. Then in the subquery, we find out the products that were not purchased by some users. And lastly, rank these products using the 'top 5' method.

Video File:
https://drive.google.com/file/d/1aTKOD__cXAeFwA6vfC4vo5ltFiHqAg6a/view?usp=sharing

Query 9

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

```sql
-- for each products in PRODUCTS, find how many it sells for each month
CREATE VIEW MonthlyPurchase AS
SELECT PRODUCTS.PName AS PName, Sum(PRODUCTS_IN_ORDERS.OQuantity)
Quantity, CONVERT(DATETIME,CAST(YEAR(DATE_TIME) AS
varchar)+'.'+CAST(MONTH(DATE_TIME) AS varchar)+'.01', 102) AS YEAR_MONTH
FROM
PRODUCTS_IN_ORDERS, ORDERS, PRODUCTS
WHERE PRODUCTS.PName = PRODUCTS_IN_ORDERS.PName
AND ORDERS.OID= PRODUCTS_IN_ORDERS.OID
GROUP BY MONTH(DATE_TIME), YEAR(DATE_TIME),PRODUCTS.Pname

GO

SELECT DISTINCT p1.Pname AS Pname FROM
MonthlyPurchase AS p1, MonthlyPurchase AS p2, MonthlyPurchase AS p3
WHERE DATEDIFF(MONTH, p1.YEAR_MONTH,p2.YEAR_MONTH)=1
AND DATEDIFF(MONTH, p2.YEAR_MONTH,p3.YEAR_MONTH)=1
AND p1.YEAR_MONTH < p2.YEAR_MONTH
AND p2.YEAR_MONTH < p3.YEAR_MONTH
AND p1.Quantity < p2. Quantity
AND p2.Quantity < p3.Quantity
AND p1.PName=p2.PName
AND p2.PName=p3.PName
```

Output:

PName
iPhone 11
iPhone 7
Samsung S21

Description:
First, we create a view named MonthlyPurchase that records the quantity sold for each product in each month. It is achieved by summing the OQuantity in PRODUCTS_IN_ORDERS for all products in PRODUCTS for each month.

Then, we create 3 copies of MonthlyPurchase view, and find products that are sold with increasing quantity for consecutive 3 months.

Video File:
https://drive.google.com/file/d/1NzLs_oYk_UYHVq0BSHgTiDzuPP9gHRCF/view?usp=sharing

ALL VIDEO FILES IN DRIVE:
https://drive.google.com/drive/folders/1wDowRylkVSYN9WwI_4IskiO73DBLONs2?usp=sharing

Additional Queries

Q1

Frequent shoppers are shoppers who have purchased more than 2 items per shop for at least 5 shops in the last 30 days. Who are the top 3 frequent shoppers in terms of the total cost of the items they have purchased?

```
SELECT t5.UID, totalcost
FROM(
SELECT t4.UID, OID, SUM(cost) as totalcost, COUNT(DISTINCT SName) as shopNames
FROM(
SELECT t2.UID, t2.OID, t2.Date_time, t3.totalQuantity, t2.SName, (t2.cost)
FROM(
       SELECT *
   FROM(
   SELECT UID, Date_time, OrdersTable.OID, joined_products_order.SName,
joined_products_order.cost
   FROM(
       SELECT UID, OID, Date_time
       FROM ORDERS
   )AS OrdersTable
  JOIN(
      SELECT OID, PRODUCTS_IN_ORDERS.SName, (OPrice*OQuantity) as cost
      FROM PRODUCTS_IN_ORDERS
  )AS joined_products_order
  ON OrdersTable.OID = joined_products_order.OID
)AS t1
)AS t2
JOIN(
   SELECT *
FROM(
       SELECT *
   FROM(
   SELECT UID, Date_time, OrdersTable.OID, totalQuantity
```

```
   FROM(
       SELECT UID, OID, Date_time
       FROM ORDERS
   )AS OrdersTable
  JOIN(
      SELECT OID, SUM(OQuantity) AS totalQuantity
      FROM PRODUCTS_IN_ORDERS
      GROUP BY OID
  )AS joined_products_order
  ON OrdersTable.OID = joined_products_order.OID
)AS t1
)AS t2
)AS t3
ON t3.UID = t2.UID AND (NOT t3.OID = t2.OID OR t3.totalQuantity>5) AND DATEDIFF(DAY,
t2.Date_time, t3.Date_time)>=0 AND DATEDIFF(DAY, t2.Date_time, t3.Date_time)<30
)AS t4
WHERE t4.totalQuantity>=2
GROUP BY OID, UID
)AS t5
WHERE t5.shopNames>=5
ORDER BY  totalcost DESC
```

Result

	USERS	Expenditure
1	8	14236.8
2	7	11660.8
3	6	9484.9

Q2

Popular shops are shops which have sold more than 3 items in the last 30 days. Who are the top three shoppers in these popular shops in terms of the number of items they have purchased?

```sql
SELECT UID, SUM(OQuantity) AS TotalQuantity FROM PRODUCTS_IN_ORDERS
JOIN ORDERS ON ORDERS.OID = PRODUCTS_IN_ORDERS.OID
WHERE SName IN (
   SELECT SName FROM PRODUCTS_IN_ORDERS
   WHERE Delivery_date >= DATEADD(day,-30,GETDATE()) AND Delivery_date <=
getdate()
   GROUP BY SName
   HAVING SUM(OQuantity) > 3) AND Delivery_date >=
DATEADD(day,-30,GETDATE()) AND Delivery_date <= getdate()
GROUP BY UID
ORDER BY TotalQuantity DESC
```

Results

RESULTS

	UID	TotalQuantity
1	8	21
2	7	17
3	6	15

Table Records

Price_History

```
1   SELECT * FROM PRICE_HISTORY
```

Untitled-4

RESULTS CTRL+ALT+R

	SName	PName	START_DATE	END_DATE	Price
1	Galaxy	iPhone 11	2021-06-01 00:...	2021-07-01 00:...	1129
2	Galaxy	iPhone 11	2021-07-02 00:...	2021-08-02 00:...	1099
3	Galaxy	iPhone 11	2021-08-03 00:...	2021-09-03 00:...	1079
4	Galaxy	Samsung S21	2021-06-01 00:...	2021-07-01 00:...	875
5	Galaxy	Samsung S21	2021-07-02 00:...	2021-08-02 00:...	809.4
6	Galaxy	Samsung S21	2021-08-03 00:...	2021-09-03 00:...	799.9
7	Galaxy	Samsung S22	2021-06-01 00:...	2021-07-01 00:...	901
8	Galaxy	Samsung S22	2021-07-02 00:...	2021-08-02 00:...	900
9	Galaxy	Samsung S22	2021-08-03 00:...	2021-09-03 00:...	899
10	Megastore	iPhone 11	2021-06-01 00:...	2021-07-01 00:...	1099
11	Megastore	iPhone 11	2021-07-02 00:...	2021-08-02 00:...	1089
12	Megastore	iPhone 11	2021-08-03 00:...	2021-09-03 00:...	1079
13	Megastore	Samsung S21	2021-06-01 00:...	2021-07-01 00:...	899
14	Megastore	Samsung S21	2021-07-02 00:...	2021-08-02 00:...	888
15	Megastore	Samsung S21	2021-08-03 00:...	2021-09-03 00:...	877
16	Shiokee 1	iPhone 7	2021-06-01 00:...	2021-07-01 00:...	999.69
17	Shiokee 1	iPhone 7	2021-07-02 00:...	2021-08-02 00:...	420.69
18	Shiokee 1	iPhone 7	2021-08-03 00:...	2021-09-03 00:...	69.42
19	Shiokee 1	iPhone Xs	2021-06-01 00:...	2021-07-01 00:...	1051.3
20	Shiokee 1	iPhone Xs	2021-07-02 00:...	2021-08-02 00:...	1047.8
21	Shiokee 1	iPhone Xs	2021-08-03 00:...	2021-09-03 00:...	1024.3
22	Shiokee 1	Samsung S22	2021-06-01 00:...	2021-07-01 00:...	900
23	Shiokee 1	Samsung S22	2021-07-02 00:...	2021-08-02 00:...	900
24	Shiokee 1	Samsung S22	2021-08-03 00:...	2021-09-03 00:...	900
25	Shiokee 2	iPhone 11	2021-06-01 00:...	2021-07-01 00:...	1099
26	Shiokee 2	iPhone 11	2021-07-02 00:...	2021-08-02 00:...	1099

Products_In_Orders

```sql
SELECT * FROM PRODUCTS_IN_ORDERS
```

Untitled-5

RESULTS CTRL+ALT+R

	PName	SName	OPID	OID	OPrice	OQuantity	Delivery_date
1	iPhone 11	Galaxy	1	7	1129	1	2021-06-12 18
2	iPhone 11	MegaStore	3	6	1079	6	2021-08-22 10
3	iPhone 11	MegaStore	1	8	1089	2	2021-07-22 10
4	iPhone 7	Shiokee 1	2	1	999.69	1	2021-06-04 10
5	iPhone 7	Shiokee 1	1	3	420.69	4	2021-07-12 10
6	iPhone 7	Shiokee 1	1	4	999.69	5	2021-08-04 10
7	iPhone Xs	Shiokee 1	1	1	1051.3	1	2021-06-04 10
8	iPhone Xs	Shiokee 1	1	5	1059.3	1	2021-08-10 10
9	iPhone Xs	Shiokee 1	1	11	1051.3	6	2021-06-16 10
10	iPhone Xs	Shiokee 2	1	2	1040.8	2	2021-07-10 10
11	Samsung S21	Galaxy	2	6	799.9	1	2021-08-12 10
12	Samsung S21	MegaStore	3	1	899	2	2021-06-04 10
13	Samsung S21	MegaStore	2	8	888	3	2021-07-22 10
14	Samsung S21	MegaStore	1	9	877	1	2021-08-22 10
15	Samsung S21	Shiokee 3	2	9	800	2	2021-08-22 10
16	Samsung S22	Galaxy	3	9	899	3	2021-08-22 10
17	Samsung S22	Shiokee 1	1	10	900	4	2021-06-19 15
18	Samsung S22	Shiokee 2	2	10	919	2	2021-06-19 15
19	Samsung S22	Shiokee 3	1	6	899.89	1	2021-08-12 10

Feedback

```
1  SELECT * FROM FEEDBACK
```

Untitled-4

RESULTS CTRL+ALT+R

	UID	PName	SName	Rating	Comment	DATE_TIME
1	1	Galaxy A	Galaxy	3	Gd!	2021-08-20 16:...
2	1	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...
3	1	iPhone X	Apple	5	wonderful prod	2021-08-20 16:...
4	2	Galaxy A	Galaxy	5	Great!	2021-08-20 16:...
5	2	Galaxy Fit	Galaxy	3	can be betteer	2021-08-20 16:...
6	2	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...
7	2	iPhone 9	Apple	5	great!!	2021-08-20 16:...
8	3	Galaxy A	Galaxy	5	Great!	2021-08-20 16:...
9	3	Galaxy M	Galaxy	4	NICE!	2021-08-20 16:...
10	3	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...
11	3	iPhone 8	Apple	5	nice	2021-08-20 16:...
12	4	Galaxy A	Galaxy	5	Great!	2021-08-20 16:...
13	4	Galaxy Note	Galaxy	4	GREAT experie...	2021-08-20 16:...
14	4	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...
15	4	iPhone 7	Apple	5	cute!	2021-08-20 16:...
16	5	Galaxy A	Galaxy	5	wonderful!	2021-08-20 16:...
17	5	Galaxy S	Galaxy	5	will buy again	2021-08-20 16:...
18	6	Galaxy A	Galaxy	5	Great!	2021-08-20 16:...
19	6	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...
20	6	Galaxy S22	Galaxy	5	nice seller!	2021-08-20 16:...
21	7	Galaxy A	Galaxy	5	Great!	2021-08-20 16:...
22	7	Galaxy S	Galaxy	5	Great!	2021-08-20 16:...

MESSAGES CTRL+ALT+Y

Products_In_Shops

```
1   SELECT * FROM PRODUCTS_IN_SHOPS
```

Untitled-4

RESULTS

	PName	SPID	SPrice	SQuantity	SName
1	Galaxy A	17	877	18	MegaStore
2	Galaxy S22	14	899	10	Galaxy
3	Galaxy S22	15	900	14	Shiokee 1
4	Galaxy S22	16	899	15	Shiokee 2
5	iPhone 11	7	1079	12	Galaxy
6	iPhone 11	8	1079	14	MegaStore
7	iPhone 11	3	1049	20	Shiokee 2
8	iPhone 7	2	69.42	10	Shiokee 1
9	iPhone Xs	1	1024.3	10	Shiokee 1
10	iPhone Xs	4	1014.3	10	Shiokee 2
11	Samsung S21	6	799.9	19	Galaxy
12	Samsung S21	13	877	18	MegaStore
13	Samsung S21	10	800	11	Shiokee 3
14	Samsung S22	9	899	10	Galaxy
15	Samsung S22	11	900	14	Shiokee 1
16	Samsung S22	12	899	15	Shiokee 2
17	Samsung S22	5	899.89	12	Shiokee 3

Complaints

```
1  SELECT * FROM COMPLAINTS
```

Untitled-4

RESULTS CTRL+ALT+R

	CID	Text	FILED_DATE_TI...	Status	UID	EmployeeID	HANDLED_DAT...
1	1	Phone Broken	2022-01-08 23:...	Addressed	2	1	2022-01-09 23:...
2	2	Phone Screen B...	2022-02-10 23:...	Addressed	5	1	2022-02-12 21:...
3	3	Phone Screen B...	2022-01-08 23:...	Addressed	12	2	2022-01-09 23:...
4	4	Phone Broken	2022-02-26 23:...	Addressed	22	1	2022-03-02 23:...
5	5	Rude Store Ow...	2022-03-10 23:...	Addressed	34	2	2022-03-15 23:...
6	6	Phone Screen B...	2022-01-10 23:...	Addressed	56	3	2022-02-15 23:...
7	7	Unresponsive S...	2022-03-14 23:...	Addressed	66	4	2022-03-19 23:...
8	8	Phone Broken	2022-03-18 23:...	Addressed	67	5	2022-03-20 23:...
9	9	Phone Screen B...	2022-03-20 23:...	Addressed	82	6	2022-03-21 23:...

Complaints_On_Orders

```
1  SELECT * FROM COMPLAINTS_ON_ORDERS
```

Untitled-4

RESULTS

	CID	OID
1	1	2
2	2	4
3	3	7
4	4	5
5	6	6
6	8	1
7	9	3

Products

```
1   SELECT * FROM PRODUCTS
```

Untitled-4

RESULTS

	PName	Maker	Category
1	Galaxy A	Samsung	Phone
2	Galaxy Fit	Samsung	Watch
3	Galaxy M	Samsung	Phone
4	Galaxy Note	Samsung	Phone
5	Galaxy S	Samsung	Phone
6	Galaxy S22	Samsung	Phone
7	Galaxy S7	Samsung	Phone
8	Galaxy S9	Samsung	Phone
9	Galaxy Tab A	Samsung	Tablet
10	Galaxy Tab S8	Samsung	Tablet
11	Galaxy Watch4	Samsung	Watch
12	Galaxy Z	Samsung	Phone
13	iPhone 11	Apple	Phone
14	iPhone 7	Apple	Phone
15	iPhone 8	Apple	Phone
16	iPhone 9	Apple	Phone
17	iPhone X	Apple	Phone
18	iPhone Xs	Apple	Phone
19	Samsung S21	Samsung	Phone
20	Samsung S22	Samsung	Phone

Orders

```
1   SELECT * FROM ORDERS
```

Untitled-4

RESULTS

	OID	Date_time	Shipping_addr...	UID
1	1	2021-06-02 23:...	10 Upper Seran...	1
2	2	2021-07-08 23:...	10 Upper Seran...	1
3	3	2021-07-10 23:...	10 Upper Seran...	2
4	4	2021-08-02 23:...	10 Upper Seran...	2
5	5	2021-08-08 23:...	10 Upper Seran...	1
6	6	2021-08-10 23:...	10 Upper Seran...	1
7	7	2021-06-10 23:...	10 Upper Seran...	2
8	8	2021-07-22 23:...	10 Upper Seran...	3
9	9	2021-08-22 23:...	10 Upper Seran...	4
10	10	2021-06-15 23:...	10 Upper Seran...	5
11	11	2021-06-15 23:...	10 Upper Seran...	15

Shops

```
1   SELECT * FROM SHOPS
```

Untitled-5

RESULTS

	SName
1	Apple
2	EarthJoy
3	Eldorado
4	Galaxy
5	KidCity
6	KKBeauty
7	MegaStore
8	PrimeSpot
9	Shiokee 1
10	Shiokee 2
11	Shiokee 3
12	SpringFoods

Users

```
1   SELECT * FROM USERS
```

Untitled-5

RESULTS

	UID	UName
1	1	Andy
2	2	Bharat
3	3	Cindy
4	4	Phillips
5	5	Jon
6	6	Chris
7	7	Julia
8	8	Phoebe
9	9	Mary
10	10	Lucy
11	11	Amanda
12	12	Liam
13	13	Olivia
14	14	Noah
15	15	Emma
16	16	Oliver
17	17	Ava
18	18	Elijah
19	19	Charlotte
20	20	William
21	21	Sophia
22	22	James

Complaints_On_Shops

```
1   SELECT * FROM COMPLAINTS_ON_SHOPS
```

Untitled-5

RESULTS

	CID	SName
1	5	MegaStore
2	7	Galaxy

Employees

```
1   SELECT * FROM EMPLOYEES
```

Untitled-5

RESULTS

	EmployeeID	Name	Salary
1	1	Andrew	1200
2	2	Betty	2000
3	3	Charlie	1500
4	4	Derrick	1600
5	5	Elise	2100
6	6	Frank	1900
7	7	Grace	3400
8	8	Harry	2000