Safeway Project

Wanxin Xia – MSBA

Table of Contents

1. Business Scenario Description 3

2. List of Table and Table Creation Script 3

3. SQL Queries – Without JOIN & JOIN 5

4. Advanced SQL & VIEW & TRIGGER & PROCEDURE 11

5. Tableau Report 17

6. Conclusion 17

1. Business Scenario Description

Vincy walks into a Safeway nearby her home at 10:10 AM. As Vincy has a lot of stuff to purchase, she grabs a shopping basket before starting to shop. Vincy walks to diary area for Mozzarella cheese and lactose-free milk. Vincy finds a kind of family-sized Mozzarella and picks up one and she puts 1 box of milk into the basket as well. Then Vincy heads to ingredient area and picks up 1 box of basil leaves, 1 box of thyme leaves and 1 bottle of corn oil. Vincy suddenly remembers to buy a comb but she cannot find where the comb is. She asks Tom, a Safeway employee, and Tom asks Vincy to find it in personal care area. Vincy goes there and picks up 1 comb. Finally, Vincy walks to food area and grabs 1 bag of bread, 1 bag of tomatoes, 1 bag of onions and 2 boxes of Tofu. After finishing shopping, Vincy heads to checkout counter. Lisa, another Safeway employee, scans items and asks Vincy whether she has a Safeway member card. Vincy shows her member card to Lisa and Lisa scans it as well. Lisa tells that the total purchase is $46.98 with tax. Vincy makes her payment by using her debit card. Vincy takes the receipt from Lisa and leaves Safeway at 10:55 AM.

2. List of Table and Table Creation Script

There are four normalized tables:

Customer table

```
CREATE TABLE Customer
(
FirstName   VARCHAR(50) NOT NULL,
LastName    VARCHAR(50),
CustomerId  INT NOT NULL,
MemberID    INT,
CONSTRAINT PKCustomerID PRIMARY KEY (CustomerID)
);
```

Orders table

```
CREATE TABLE Orders
(
OrderID     INT NOT NULL,
OrderDate   DATETIME NOT NULL,
CustomerID  INT NOT NULL,
CONSTRAINT PKOrderID PRIMARY KEY (OrderID),
CONSTRAINT FK1 FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);
```

Product table

```
CREATE TABLE Product
(
ProductID       INT NOT NULL,
Descriptions    VARCHAR(100),
UnitPrice       DECIMAL(5,2) NOT NULL,
CONSTRAINT PKProductID PRIMARY KEY (ProductID)
);
```

Orderline table

```
CREATE TABLE OrderLine
(
OrderID     INT NOT NULL,
ProductID   INT NOT NULL,
Volume      DECIMAL(5,2) NOT NULL,
CONSTRAINT FK2 FOREIGN KEY (OrderID) REFERENCES Orders (OrderID),
CONSTRAINT FK3 FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
CONSTRAINT PKBuys PRIMARY KEY (OrderID, ProductID)
);
```

And one denormalized table:

```
CREATE TABLE Denormalized
(
FirstName       VARCHAR(50),
LastName        VARCHAR(50),
CustomerId      INT,
MemberID        INT,
OrderID         INT,
OrderDate       DATETIME,
ProductID       INT,
Descriptions    VARCHAR(100),
UnitPrice       DECIMAL(5,2),
Volume          DECIMAL(5,2)
);
```

3. SQL Queries – Without JOIN & JOIN

Five queries without JOIN:

a. How many customers have membership?

```
SELECT COUNT(*) AS 'Count'
FROM customer
WHERE MemberID > 0;
```

Result:

Count
6

b. How many different orders did occur?

```
SELECT COUNT(*)
FROM orders;
```

Result:

COUNT(*)
13

c. How many days do these orders expand?

```
SELECT DATEDIFF(MAX(OrderDate),MIN(OrderDate)) AS 'date expand'
FROM orders;
```

Result:

date expand
8

d. What's the average unit price of all the products?

```
SELECT AVG(UnitPrice) AS 'avg unit price'
FROM product;
```

Result:

avg unit price
4.903555555555555

e. What's the total volume does OrderId 13 purchased?

```
SELECT SUM(Volume) AS 'total volume'
FROM orderline
GROUP BY OrderID
HAVING OrderID = 13;
```

Result:

total volume
13.48

Ten questions with JOIN:

a. What are customer names for each order?

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name', OrderID, OrderDate
FROM orders o
LEFT JOIN customer c ON o.CustomerID = c.CustomerID;
```

Result:

Customer Name	OrderID	OrderDate
Amanda Mackenzie	1	2020-02-11
April Baker	2	2020-02-11
Burton Tracy	3	2020-02-11
Amanda Mackenzie	4	2020-02-13
Charles Owen	5	2020-02-12
Darren Smith	6	2020-02-13
David Farrell	7	2020-02-13
Darren Smith	8	2020-02-16
Ponder Stibbons	9	2020-02-13
Janice Joplette	10	2020-02-14
Amanda Mackenzie	11	2020-02-19
Jemima Farrell	12	2020-02-18
Vincy Smith	13	2020-02-11

b. Which customer has the most orders?

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name', COUNT(OrderID) AS 'Count'
FROM orders o
LEFT JOIN customer c ON o.CustomerID = c.CustomerID
GROUP BY o.CustomerID
ORDER BY 2 DESC
LIMIT 1;
```

Result:

Customer Name	Count
Amanda Mackenzie	3

c. What's the daily revenue?

```
SELECT o.OrderDate, SUM((ol.Volume * p.UnitPrice)) AS 'Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
GROUP BY 1
ORDER BY 1;
```

Result:

OrderDate	Revenue
2020-02-11	96.82020000000001
2020-02-12	37.93
2020-02-13	63.89000000000001
2020-02-14	12.170000000000002
2020-02-16	20.96
2020-02-18	6.68
2020-02-19	14.260000000000002

d. Which customer spent the most?

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name', SUM((ol.Volume * p.UnitPrice)) AS 'Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
LEFT JOIN customer c ON c.CustomeriD = o.CustomerID
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

Result:

Customer Name	Revenue
Vincy Smith	47.330200000000005

e. Which product is the most popular?

```
SELECT Descriptions, COUNT(ol.OrderID) AS 'Count'
FROM orderline ol
INNER JOIN product p ON ol.ProductID = p.ProductID
GROUP BY p.ProductID
ORDER BY 2 DESC
LIMIT 1;
```

Result:

Descriptions	Count
Milk	5

f. Which customers purchased 'Tofu'?

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
LEFT JOIN customer c ON c.CustomeriD = o.CustomerID
WHERE p.ProductID = 10;
```

Result:

Customer Name
Amanda Mackenzie
David Farrell
Darren Smith
Vincy Smith

g. Which product contributes the most to revenue?

```
SELECT Descriptions, SUM((ol.Volume * p.UnitPrice)) AS 'Revenue'
FROM orderline ol
INNER JOIN product p ON ol.ProductID = p.ProductID
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

Result:

Descriptions	Revenue
Milk	26.940000000000005

h. What product did each member purchase?

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name', Descriptions
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
LEFT JOIN customer c ON c.CustomeriD = o.CustomerID
WHERE MemberID > 0;
```

Result:

Customer Name	Descriptions
April Baker	Milk
April Baker	WhiteAmericanCheese
April Baker	LadyGreyTea
April Baker	VanillaSyrup
Charles Owen	Milk
Charles Owen	CleansingFoam
Charles Owen	MenstrualPainRelief
Charles Owen	EnergyCereal
Darren Smith	MoisturizingCreamSensitive
Darren Smith	RedBeans
Darren Smith	OvenCleaner
Darren Smith	Tofu
Darren Smith	Beer
Darren Smith	OrangesSyrup
Ponder Stibbons	Milk
Ponder Stibbons	MortonSalt
Ponder Stibbons	GreenOlives
Jemima Farrell	LeanSpaghetti
Jemima Farrell	CinnamonSquares
Vincy Smith	BasilLeaves
Vincy Smith	ThymeLeaves
Vincy Smith	CornOlil
Vincy Smith	Mozarrella
Vincy Smith	Milk
Vincy Smith	Comb
Vincy Smith	Bread
Vincy Smith	Onion
Vincy Smith	Tomato
Vincy Smith	Tofu

i. How much money did MemberID 1 spend?

```
SELECT SUM((ol.Volume * p.UnitPrice)) AS 'Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
LEFT JOIN customer c ON c.CustomeriD = o.CustomerID
WHERE MemberID = 1;
```

Result:

Revenue
24.04

j. What is the average payment of all the orders?

```
SELECT AVG(ol.Volume * p.UnitPrice) AS 'Average Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID;
```

Result:

Average Revenue
5.615782222222224

4. Advanced SQL & VIEW & TRIGGER & PROCEDURE

Four advanced SQL:

a. UNION - return max and min revenue for each customer.

```
WITH t1 AS (
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer_Name', SUM((ol.Volume * p.UnitPrice)) AS 'Revenue'
    FROM orderline ol, product p, orders o, customer c
    WHERE ol.ProductID = p.ProductID
        AND o.OrderID = ol.OrderID
        AND c.CustomeriD = o.CustomerID
GROUP BY 1
)
SELECT t1.Customer_Name, Revenue
FROM t1
WHERE t1.Revenue = (SELECT MAX(Revenue) FROM t1)

UNION

SELECT t1.Customer_Name, Revenue
FROM t1
WHERE t1.Revenue = (SELECT MIN(Revenue) FROM t1);
```

Result:

Customer_Name	Revenue
Vincy Smith	47.330200000000005
Jemima Farrell	6.68

b. Date function - How many orders in each month?

```
SELECT MONTHNAME(OrderDate) AS 'Month', COUNT(*)
FROM orders
GROUP BY 1;
```

Result:

Month	COUNT(*)
February	13

c. RANK - rank each customer's revenue

```
SELECT CONCAT(FirstName,' ',LastName) AS 'Customer Name', SUM(ol.Volume * p.UnitPrice) AS 'Revenue',
RANK () OVER(ORDER BY SUM(ol.Volume * p.UnitPrice) DESC) amount_rank
FROM orderline ol, product p, orders o, customer c
WHERE ol.ProductID = p.ProductID
AND o.OrderID = ol.OrderID
AND c.CustomeriD = o.CustomerID
GROUP BY 1;
```

Result:

Customer Name	Revenue	amount_rank
Vincy Smith	47.330200000000005	1
Darren Smith	44.82	2
Amanda Mackenzie	44.42	3
Charles Owen	37.93	4
April Baker	24.04	5
Ponder Stibbons	18.37	6
Janice Joplette	12.170000000000002	7
David Farrell	9.48	8
Burton Tracy	7.470000000000001	9
Jemima Farrell	6.68	10

d. CASE - classify revenue

```
SELECT Descriptions, SUM(o.Volume * p.UnitPrice) AS 'Revenue',
    CASE
        WHEN SUM(o.Volume * p.UnitPrice) > 15 THEN 'high revenue'
    ELSE 'low revenue'
END AS revenue_class
FROM orderline o
LEFT JOIN product p ON o.ProductID = p.ProductID
GROUP BY 1
ORDER BY 2 DESC;
```

Result:

Descriptions	Revenue	revenue_class
Milk	26.940000000000005	high revenue
Beer	19.950000000000003	high revenue
Tofu	17.5	high revenue
EnergyCereal	15.78	high revenue
Jazmine Rice	14.99	low revenue
GreenOlives	12.89	low revenue
MoisturizingCreamSensitive	11.99	low revenue
MenstrualPainRelief	11.97	low revenue
Tomato	9.935100000000002	low revenue
Mozarrella	8.99	low revenue
WhiteAmericanCheese	8.99	low revenue
VanillaSyrup	8.97	low revenue
Guava	7.98	low revenue
RedBeans	5.98	low revenue
OvenCleaner	5.89	low revenue
PizzaSauce	5.69	low revenue
CleansingFoam	5.69	low revenue
LeanSpaghetti	4.29	low revenue
TomatoKetchup	4.19	low revenue
OrangesSyrup	3.99	low revenue
BasilLeaves	3.99	low revenue
ThymeLeaves	3.49	low revenue
CornOlil	3.49	low revenue
RiceVinegar	3.29	low revenue
Yogurt	2.99	low revenue
Bread	2.99	low revenue
CombSet	2.99	low revenue
Onion	2.9651	low revenue
CinnamonSquares	2.39	low revenue
GluePen	1.99	low revenue
RedTea	1.99	low revenue
Comb	1.99	low revenue
LadyGreyTea	1.59	low revenue
PeachWhiteTea	1.49	low revenue
Peppercorn	1.49	low revenue
MortonSalt	0.99	low revenue

Two VIEW:

a. Create a view of revenue

```
CREATE VIEW revenue AS
SELECT c.CustomerID, o.OrderID, p.ProductID, (ol.Volume * p.UnitPrice) AS 'Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
LEFT JOIN customer c ON c.CustomeriD = o.CustomerID;

SELECT * FROM revenue;
```

Result:

CustomerID	OrderID	ProductID	Revenue
1	1	11	14.99
1	1	12	2.99
2	2	5	4.49
2	2	13	8.99
2	2	15	1.59
2	2	38	8.97
3	3	16	1.49
3	3	17	3.99
3	3	39	1.99
1	4	10	5
1	4	19	1.49
1	4	40	5.69
4	5	5	4.49
4	5	20	5.69
4	5	21	11.97
4	5	41	15.78
5	6	23	11.99
5	6	24	5.98
5	6	42	5.89
6	7	10	2.5
6	7	17	3.99
6	7	25	2.99
5	8	10	5
5	8	17	11.97
5	8	27	3.99
7	9	5	4.49
7	9	30	0.99
7	9	45	12.89
8	10	31	7.98
8	10	32	4.19
1	11	5	8.98
1	11	33	1.99
1	11	34	3.29
9	12	36	4.29
9	12	37	2.39
10	13	1	3.99
10	13	2	3.49
10	13	3	3.49
10	13	4	8.99
10	13	5	4.49
10	13	6	1.99
10	13	7	2.99
10	13	8	2.9651
10	13	9	9.9351...
10	13	10	5

b. Create a view of daily revenue

```
CREATE VIEW time_revenue AS
SELECT OrderDate, SUM(ol.Volume * p.UnitPrice) AS 'Revenue'
FROM orderline ol
LEFT JOIN product p ON ol.ProductID = p.ProductID
LEFT JOIN orders o ON o.OrderID = ol.OrderID
GROUP BY 1
ORDER BY 1;

SELECT * FROM time_revenue;
```

Result:

OrderDate	Revenue
2020-02-11	96.82020000000001
2020-02-12	37.93
2020-02-13	63.89000000000001
2020-02-14	12.170000000000002
2020-02-16	20.96
2020-02-18	6.68
2020-02-19	14.260000000000002

One TRIGGER – When update product information, automatically update trigger table

```
CREATE TABLE ProductUpdates
(
ProductID       INT,
Descriptions    VARCHAR(50),
UnitPrice       DECIMAL(5,2)
);

CREATE TRIGGER ProductUpdateTrigger
AFTER UPDATE ON product
FOR EACH ROW
    INSERT INTO ProductUpdates
        SELECT ProductID, Descriptions, UnitPrice FROM product;

SET SQL_SAFE_UPDATES = 0;
UPDATE product
SET UnitPrice = 3.99
WHERE ProductID = 1;

SELECT * FROM ProductUpdates;
SELECT * FROM product;
```

Result:

ProductID	Descriptions	UnitPrice
1	BasilLeaves	3.99

(ProductUpdate table)

ProductID	Descriptions	UnitPrice
1	BasilLeaves	3.99

(Product table)

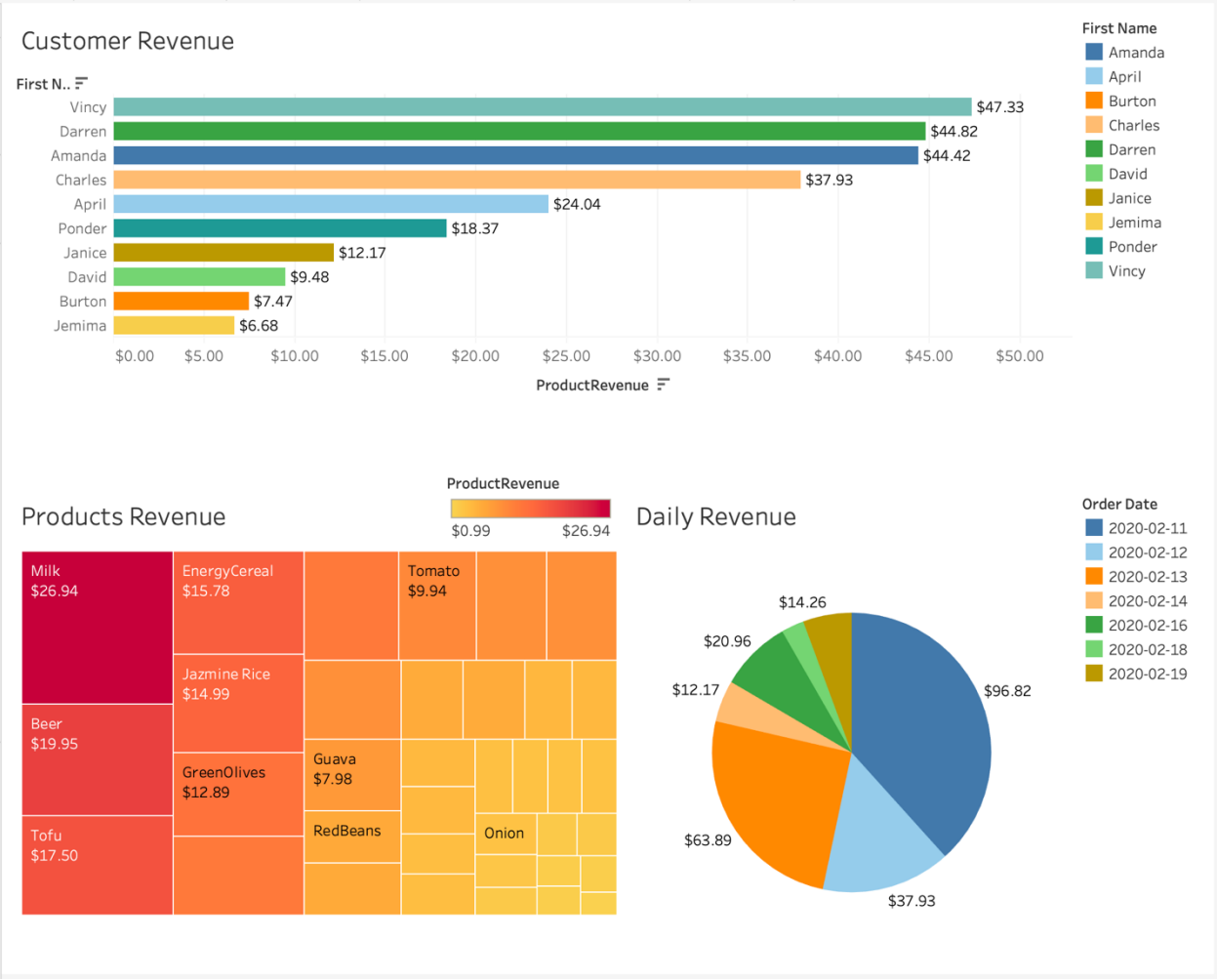
One PROCEDURE – Update customer information

```
DELIMITER //
CREATE PROCEDURE UpdateCust(IN id INT, IN f_name CHAR(20))
BEGIN
UPDATE customer
SET FirstName = f_name
WHERE CustomerID = id;
END //

CALL UpdateCust(1,'Amanda');
CALL UpdateCust(2,'April');

SELECT * FROM customer;
```

Result:

FirstName	LastName	CustomerID	MemberID
Amanda	Mackenzie	1	
April	Baker	2	1

5. Tableau Report



6. Conclusion

- Among all the customers, Vincy purchased the most. Given this, Safeway can send more advertisement to Vincy to boost her purchasing.
- From the Products Revenue chart, we find that milk is the most popular product. Safeway can introduce more brands of milk to further boost sales.
- It seems that the revenue on weekend is lower than that on weekday. It is possible for Safeway to take actions to promote sales on weekend.