

# MoneySmart Coding Challenge

Lim Hyelim, Data Analyst

# Objective

To share how I approached and solved the questions of MoneySmart coding challenge.

# Question1

Write an SQL statement to find the **total number of user sessions each page has each day**. (A user session is defined as continuous activity on a site where each activity is within 10 mins of each other.)

ID	User_ID	Page_ID	Visit_Date	Visit_Time
1	1	54	2018-01-01	11:54:34
2	1	55	2018-01-01	11:55:10
3	1	56	2018-01-02	13:11:12
4	1	55	2018-01-02	17:10:08
5	1	56	2018-01-02	17:12:45

```
SELECT
```

```
    Page_ID,
```

```
    Visit_Date,
```

```
    count(ID) AS Total_User_Sessions
```

```
FROM `moneysmart`.`samplepageviews`
```

```
GROUP BY Page_ID, Visit_Date
```

```
ORDER BY Page_ID
```

Step2: Count user sessions by using ID

Step1: Group the table by Page\_ID and Visit\_Date

# Actual result based on given table for Question 1

ID	Page_ID	Visit_Date
1	54	2018-01-01
2	55	2018-01-01
3	56	2018-01-02
4	55	2018-01-02
5	56	2018-01-02
6	55	2018-01-01
7	55	2018-01-01
8	55	2018-01-01
9	54	2018-01-02
10	56	2018-01-02
11	55	2018-01-02
12	56	2018-01-02



Page_ID	Visit_Date	Total_User_Sessions
54	2018-01-01	1
54	2018-01-02	1
55	2018-01-01	4
55	2018-01-02	2
56	2018-01-02	4

# Actual result based on given table for Question 1

ID	Page_ID	Visit_Date
1	54	2018-01-01
2	55	2018-01-01
3	56	2018-01-02
4	55	2018-01-02
5	56	2018-01-02
6	55	2018-01-01
7	55	2018-01-01
8	55	2018-01-01
9	54	2018-01-02
10	56	2018-01-02
11	55	2018-01-02
12	56	2018-01-02



Page_ID	Visit_Date	Total_User_Sessions
54	2018-01-01	1
54	2018-01-02	1
55	2018-01-01	4
55	2018-01-02	2
56	2018-01-02	4

# Question2

Write an SQL statement to find out which **items** are the **most frequently** being **purchased together** by **the same user**. (Could be in different Order)

RowID	OrderID	OrderDate	CustomerID	ProductID	ProductName	Sales	Quantity
1	CA-2016-152156	2008-11-16	CG-12520	FUR-BO-10001798	Bush Somerset Collection Bookcase	261.96	2
2	CA-2016-152156	2008-11-16	CG-12520	FUR-CH-10000454	Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back	731.94	3
3	CA-2016-138688	2012-06-16	DV-13045	OFF-LA-10000240	Self-Adhesive Address Labels for Typewriters by Universal	14.62	2
4	US-2015-108966	2011-10-15	SO-20335	FUR-TA-10000577	Bretford CR4500 Series Slim Rectangular Table	957.5775	5
5	US-2015-108966	2011-10-15	SO-20335	OFF-ST-10000760	Eldon Fold 'N Roll Cart System	22.368	2

Step1: Create a table with the **combinations** of two different items by joining tables of CustomerID and ProductID

t1.CustomerID	t1.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t2.CustomerID	t2.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t1.CustomerID	t1.ProductID	t2.CustomerID	t2.ProductID
Krish	A	Krish	A
Krish	B	Krish	A
Krish	C	Krish	A
Krish	A	Krish	B
Krish	B	Krish	B
Krish	C	Krish	B
Krish	A	Krish	C
Krish	B	Krish	C
Krish	C	Krish	C
Albert	A	Albert	A
Albert	B	Albert	A
Albert	A	Albert	B
Albert	B	Albert	B
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A

Joining two tables by CustomerID



Step1: Create a table with the **combinations** of two different items by joining tables of CustomerID and ProductID

t1.CustomerID	t1.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t2.CustomerID	t2.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



1 row

t1.CustomerID	t1.ProductID	t2.CustomerID	t2.ProductID
Krish	A	Krish	A
Krish	B	Krish	A
Krish	C	Krish	A
Krish	A	Krish	B
Krish	B	Krish	B
Krish	C	Krish	B
Krish	A	Krish	C
Krish	B	Krish	C
Krish	C	Krish	C
Albert	A	Albert	A
Albert	B	Albert	A
Albert	A	Albert	B
Albert	B	Albert	B
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A

duplicated row

1. Remove the duplicated rows

2. Remove non-cross-sell cases

3. Remove the duplicated combinations of two different items

Step1: Create a table with the **combinations** of two different items by joining tables of CustomerID and ProductID

t1.CustomerID	t1.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t2.CustomerID	t2.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t1.CustomerID	t1.ProductID	t2.CustomerID	t2.ProductID
<del>Krish</del>	<del>A</del>	<del>Krish</del>	<del>A</del>
Krish	B	Krish	A
Krish	C	Krish	A
Krish	A	Krish	B
<del>Krish</del>	<del>B</del>	<del>Krish</del>	<del>B</del>
Krish	C	Krish	B
Krish	A	Krish	C
Krish	B	Krish	C
<del>Krish</del>	<del>C</del>	<del>Krish</del>	<del>C</del>
<del>Albert</del>	<del>A</del>	<del>Albert</del>	<del>A</del>
Albert	B	Albert	A
Albert	A	Albert	B
<del>Albert</del>	<del>B</del>	<del>Albert</del>	<del>B</del>
<del>Hyelim</del>	<del>A</del>	<del>Hyelim</del>	<del>A</del>
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A

1. Remove the duplicated rows

2. Remove non-cross-sell cases  
(t1.ProductID = t2.ProductID)

3. Remove the duplicated combinations of two different items

Step1: Create a table with the combinations of two different items by joining tables of CustomerID and ProductID

t1.CustomerID	t1.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t2.CustomerID	t2.ProductID
Krish	A
Krish	B
Krish	C
Albert	A
Albert	B
Hyelim	A
Hyelim	A



t1.CustomerID	t1.ProductID	t2.CustomerID	t2.ProductID
Krish	A	Krish	A
<del>Krish</del>	<del>B</del>	<del>Krish</del>	<del>A</del>
<del>Krish</del>	<del>C</del>	<del>Krish</del>	<del>A</del>
Krish	A	Krish	B
Krish	B	Krish	B
<del>Krish</del>	<del>C</del>	<del>Krish</del>	<del>B</del>
Krish	A	Krish	C
Krish	B	Krish	C
Krish	C	Krish	C
Albert	A	Albert	A
<del>Albert</del>	<del>B</del>	<del>Albert</del>	<del>A</del>
Albert	A	Albert	B
Albert	B	Albert	B
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A
Hyelim	A	Hyelim	A

1. Remove the duplicated rows

2. Remove non-cross-sell cases  
(t1.ProductID = t2.ProductID)

3. Remove the duplicated combinations of  
two different items  
(t1.ProductID > t2.ProductID)

Step2: Calculate the frequency of each combination of items purchased together

t1.CustomerID	t1.ProductID	t2.CustomerID	t2.ProductID
Krish	A	Krish	B
Krish	A	Krish	C
Krish	B	Krish	C
Albert	A	Albert	B



t1.ProductID	t2.ProductID	t1.CustomerID
A	B	2
A	C	1
B	C	1

```
SELECT
    Product_A,
    Product_B,
    COUNT(CustomerID) AS Unique_User_Count
```

```
FROM (
```

```
SELECT
    DISTINCT
    t1.CustomerID,
    t1.ProductID AS Product_A,
    t2.ProductID AS Product_B
FROM `moneysmart`.`sampleorders` AS t1
    INNER JOIN `moneysmart`.`sampleorders` AS t2 ON t1.CustomerID = t2.CustomerID

WHERE t1.ProductID <> t2.ProductID
    and t1.ProductID < t2.ProductID
```

```
) a
```

```
GROUP by product_A,Product_B
ORDER by Unique_User_Count DESC;
```

Step1: Create a table with the combinations of two different items by joining tables of CustomerID and ProductID

Step2: Calculate the frequency of each combination of items purchased together

# Actual result based on given table for Question 2

Product_A (=t1.ProductID)	Product_B (=t2.ProductID)	Unique_User_Count (=t1.CustomerID)
OFF-LA-10004093	TEC-AC-10003095	4
FUR-CH-10003379	TEC-PH-10003885	3
FUR-CH-10002880	FUR-TA-10001520	3
FUR-CH-10001146	OFF-ST-10002583	3
OFF-PA-10001144	TEC-PH-10002890	3

The most popular items purchased together are 'OFF-LA-10004093' and 'TEC\_AC\_10003095'

# Question3

Write an SQL statement to find out the most frequent **date interval** between **orders** from the **same user**.

RowID	OrderID	OrderDate	CustomerID	ProductID	ProductName	Sales	Quantity
1	CA-2016-152156	2008-11-16	CG-12520	FUR-BO-10001798	Bush Somerset Collection Bookcase	261.96	2
2	CA-2016-152156	2008-11-16	CG-12520	FUR-CH-10000454	Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back	731.94	3
3	CA-2016-138688	2012-06-16	DV-13045	OFF-LA-10000240	Self-Adhesive Address Labels for Typewriters by Universal	14.62	2
4	US-2015-108966	2011-10-15	SO-20335	FUR-TA-10000577	Bretford CR4500 Series Slim Rectangular Table	957.5775	5
5	US-2015-108966	2011-10-15	SO-20335	OFF-ST-10000760	Eldon Fold 'N Roll Cart System	22.368	2

# Step1: Create a table with unique rows of CustomerID, orderID and OrderDate

OrderID	CustomerID	OrderDate
1	Krish	2018-03-21
2	Krish	2018-03-21
2	Krish	2018-03-21
3	Albert	2018-04-20
4	Hyelim	2018-04-30
5	Albert	2018-05-01
6	Albert	2018-05-12



OrderID	CustomerID	OrderDate
1	Krish	2018-03-21
2	Krish	2018-03-21
<del>2</del>	<del>Krish</del>	<del>2018-03-21</del>
3	Albert	2018-04-20
4	Hyelim	2018-04-30
5	Albert	2018-05-01
6	Albert	2018-05-12

Delete duplicates



OrderID	CustomerID	OrderDate
3	Albert	2018-04-20
5	Albert	2018-05-01
6	Albert	2018-05-12
4	Hyelim	2018-04-30
1	Krish	2018-03-21
2	Krish	2018-03-21

Rearrange the table according to CustomerID, OrderDate and then OrderID



Step2: Create a loop to calculate the date interval between the current and the previous purchase of each customer

OrderID	CustomerID	OrderDate
3	Albert	2018-04-20
5	Albert	2018-05-01
6	Albert	2018-05-12
4	Hyelim	2018-04-30
1	Krish	2018-03-21
2	Krish	2018-03-21



OrderID	CustomerID	OrderDate	Date_Interval_Since_Prev_Order
3	Albert	2018-04-20	
5	Albert	2018-05-01	<b>11 days</b>
6	Albert	2018-05-12	<b>11 days</b>
4	Hyelim	2018-04-30	
1	Krish	2018-03-21	
2	Krish	2018-03-21	<b>0 days</b>

\*A date interval will be calculated only if the CustomerID of the current row is the same as previous.

## Step3: Calculate the frequency of the date intervals

OrderID	CustomerID	OrderDate	Date_Interval_Since_Prev_Order
3	Albert	2018-04-20	
5	Albert	2018-05-01	11 days
6	Albert	2018-05-12	11 days
4	Hyelim	2018-04-30	
1	Krish	2018-03-21	
2	Krish	2018-03-21	0 days



Date_Interval_Since_Prev_Order	Order_Count
11 days	2
0 days	1

```

USE `moneysmart`;
DROP PROCEDURE IF EXISTS proc_order;
DELIMITER $$
CREATE PROCEDURE proc_order()
BEGIN
    /* Declare variables */
    DECLARE v_customerid VARCHAR(10); -- CustomerID from the current row
    DECLARE v_orderdate DATE; -- OrderDate from the current row
    DECLARE v_orderid VARCHAR(14); -- OrderID from the current row
    DECLARE old_customerid VARCHAR(10) DEFAULT ''; -- customerID from the previous row
    DECLARE old_orderdate DATE DEFAULT NULL; -- OrderDate from the previous row
    DECLARE v_datediffer INT DEFAULT NULL; -- OrdDate difference between previous and current row
    DECLARE v_finished INT DEFAULT 0; -- to escape the loop

```

Step1: Create a table with unique rows of CustomerID, orderID and OrderDate

```

/* Declare cursor */
DECLARE c1 CURSOR FOR SELECT DISTINCT CustomerID, OrderID, OrderDate FROM `moneysmart`.`sampleorders` ORDER BY CustomerID, OrderDate, OrderID;
/* Declare the following handler: When there is no more row below, assign 1 to v_finished */
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished=1;

/* Create a temporary table to store the output from the loop below*/
DROP TEMPORARY TABLE IF EXISTS `moneysmart`.`temp_output`;
CREATE TEMPORARY TABLE `moneysmart`.`temp_output` (
    `customerid` VARCHAR(10) NOT NULL,
    `datediff` INT NULL);

```

```
OPEN c1;
```

```
/* Create loop */
```

```
get_date: LOOP
```

```
/* Fetch the first row */
```

```
FETCH c1 INTO v_customerid, v_orderid, v_orderdate;
```

```
/* Escape the loop when it is the last row */
```

```
IF v_finished=1 THEN
```

```
    LEAVE get_date;
```

```
END IF;
```

```
/* Calculate date interval */
```

```
IF(v_customerid = old_customerid) THEN
```

```
    SET v_datediffer = DATEDIFF(v_orderdate,old_orderdate);
```

```
ELSE
```

```
    SET v_datediffer = NULL;
```

```
END IF;
```

```
/*Insert values into the output table*/
```

```
SET old_customerid = v_customerid;
```

```
SET old_orderdate = v_orderdate;
```

```
INSERT INTO `moneysmart`.`temp_output`
```

```
    VALUES (v_customerid,v_datediffer);
```

```
END LOOP get_date;
```

```
CLOSE c1;
```

Step2: Create a loop to calculate the date interval between the current and the previous purchase of each customer

Condition: Calculate the date interval only if the CustomerID of the current row is the same as previous

```
/* Return the answer of the question */  
SELECT  
    concat(datediff, ' ', 'days') AS Date_Interval_Since_Prev_Order,  
    COUNT(*) AS Order_Count  
FROM `moneysmart`.`temp_output`  
WHERE datediff IS NOT NULL  
GROUP BY datediff  
ORDER BY Order_Count DESC  
LIMIT 5 -- show top 5 of the most date intervals only  
;
```

```
END $$
```

```
/* Execute the procedure */
```

```
CALL proc_order() $$
```

```
/* Change delimiter from $$ back to ';' */
```

```
DELIMITER ;
```

Step3: Calculate the frequency of date intervals

# Actual result based on given table for Question 3

Date_Interval_Since _Prev_Order	Order_Count
7 days	<b>34</b>
4 days	31
11 days	31
13 days	31
34 days	30

A customer has the highest chance of returning 7 days after a purchase.