

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
-- 10개 행 출력
SELECT *
FROM uplifted-sphinx-439401-k5.modulabs_project.data
LIMIT 10;
```

Query results					
SAVE RESULTS EXPLORE DATA					
JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EX					
Row	InvoiceNo	StockCode	Description	Quantity	
1	536414	22139	null		
2	536544	22219	LOVEBIRD HANGING DECORAT...		
3	536544	85049H	URBAN BLACK RIBBONS		
4	536544	22300	COFFEE MUG DOG + BALL DES...		
5	536544	22361	GLASS JAR DAISY FRESH COT...		
6	536544	22515	CHILDS GARDEN SPADE PINK		
7	536544	90099	NECKLACE+BRACELET SET BL...		
8	536544	20854	BLUE PATCH PURSE PINK HEA...		
9	536544	20685	DOORMAT RED RETROSPOT		
10	536544	21486	PINK HEART DOTS HOT WATE...		

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
-- 전체 행의 개수 출력
SELECT count(*)
FROM uplifted-sphinx-439401-k5.modulabs_project.data;
```

Query results					
SAVE RESULTS EXPLORE DATA					
JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EX					
Row	fo_				
1	541909				

데이터 수 세기

- COUNT 함수를 사용해서, 각 컬럼 별 데이터 포인트의 수를 세어 보기

BigQuery에서는 각 컬럼별로 데이터 포인트의 수를 세는 것이 가능하지만, 모든 컬럼을 자동으로 한 번에 계산하는 기능은 없고, 각각을 수동으로 지정해야 한다.

```
-- 데이터 수 세기
SELECT count(InvoiceNo) AS COUNT_InvoiceNO,
       count(StockCode) AS COUNT_StockCode,
       count(Description) AS COUNT_Description, -- 결측치 존재
       count(Quantity) AS COUNT_Quantity,
       count(InvoiceDate) AS COUNT_InvoiceDate,
       count(UnitPrice) AS COUNT_UnitPrice,
       count(CustomerID) AS COUNT_CustomerID, -- 결측치 존재
```

```
count(Country) AS COUNT_CustomerID
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```

Query results									
JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS		EXECUTION GRAPH		
Row		COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_CustomerID
1		541909	541909	540455	541909	541909	541909	406829	541909

11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```
-- 컬럼 별 누락된 값의 비율 계산
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
  SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM `u
  plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM `u
  plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FRO
  M `uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM `upl
  ifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FRO
  M `uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM `u
  plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM
  `uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
  SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM `uplif
  ted-sphinx-439401-k5.modulabs_project.data`
) AS column_data
ORDER BY 2 DESC;
```

Query results			
JOB INFORMATION		RESULTS	CHART
Row	column_name	f0_	
1	CustomerID	24.93	
2	Description	0.27	
3	Country	0.0	
4	InvoiceDate	0.0	
5	UnitPrice	0.0	
6	Quantity	0.0	
7	InvoiceNo	0.0	
8	StockCode	0.0	

결측치 처리 전략

- `StockCode = '85123A'` 의 `Description` 을 추출하는 쿼리문을 작성하기

```
-- StockCode = '85123A'을 만족하는 Description 컬럼
SELECT DISTINCT description
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE StockCode = '85123A';
```

Query results	
< JOB INFORMATION RESULTS	
Row	description
1	WHITE HANGING HEART T-LIG...
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIG...

결측치 처리

- `DELETE` 구문을 사용하며, `WHERE` 절을 통해 데이터를 제거할 조건을 제시

```
-- 결측치 데이터 삭제
DELETE FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE description IS NULL
OR customerID IS NULL
```

Query results		SAVE RESULTS	EXPLORE DATA
JOB INFORMATION RESULTS EXECUTION DETAILS EXECUTION GRAPH			
This statement removed 135,080 rows from data.		GO TO TABLE	

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- 중복된 행의 수를 세어보기
 - 8개의 컬럼에 그룹 함수를 적용한 후, `COUNT`가 1보다 큰 데이터를 세어보기

```
-- 중복된 행의 수 계산
WITH CHECK_DUPLICATED_ROWS AS (
SELECT *, count(*) AS DUPLICATED_ROWS
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY InvoiceNo,
         StockCode,
         Description,
         Quantity,
         InvoiceDate,
         UnitPrice,
```

```

        CustomerID,
        Country
    HAVING DUPLICATED_ROWS > 1
)
SELECT count(DUPLICATED_ROWS) AS TOTAL_DUPLICATED
FROM CHECK_DUPLICATED_ROWS;

```

Query results		SAVE RESULTS
INFORMATION		RESULTS
Row	TOTAL_DUPLICATED	
1	4837	

중복값 처리

- 중복값을 제거하는 쿼리문 작성하기
 - CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(*)을 DISTINCT 한 데이터로 업데이트

```

-- 중복된 행들을 제거하여 테이블을 대체
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT DISTINCT InvoiceNo,
                StockCode,
                Description,
                Quantity,
                InvoiceDate,
                UnitPrice,
                CustomerID,
                Country
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;

```

Query results		SAVE RESULTS	EXPLORE DATA
JOB INFORMATION		RESULTS	EXECUTION DETAILS
<p>This statement replaced the table named data.</p>		GO TO TABLE	

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```

-- 고유(unique)한 InvoiceNo의 개수를 출력
SELECT count(DISTINCT InvoiceNo) AS num_of_InvoiceNo
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`

```

Query results			SAVE RESULTS
<	JOB INFORMATION	RESULTS	CHART
Row	num_of_InvoiceNo		
1	22190		

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
-- 고유한 InvoiceNo를 앞에서부터 100개를 출력
SELECT DISTINCT InvoiceNo
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
LIMIT 100;
```

Query results			SAVE RESULTS	EXPLAIN
<	JOB INFORMATION	RESULTS	CHART	JSON
Row	InvoiceNo			
1	574301			
2	C575531			
3	557305			
4	543008			
5	549735			
6	554032			
7	561387			
8	574868			
9	574827			
10	546015			

Results per page: 50 1 – 50 of 100

- InvoiceNo 가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)

```
-- InvoiceNo가 'C'로 시작하는 행을 필터링(100개)
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

Query results

[Save results](#)
[Explore data](#)

Job information									
	Results	Chart	JSON	Execution details		Execution graph			
Row	InvoiceNo	InvoiceCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
1	C575531	22060	JAMA MARRING SET WITH JAMS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain	
2	C550380	22040	ROUND CAKE TIN VINTAGE RED	-1	2011-09-26 11:35:00 UTC	7.95	15104	United Kingdom	
3	C550380	22047	BREAD BAKING SET VINTAGE IVORY	-1	2011-09-26 11:35:00 UTC	16.95	15104	United Kingdom	
4	C554983	47098	PINK HAPPY BIRTHDAY BUNTL	-20	2011-05-29 12:18:00 UTC	4.65	17162	United Kingdom	
5	C554983	47098A	BLUE HAPPY BIRTHDAY BUNTL	-20	2011-05-29 12:18:00 UTC	4.65	17162	United Kingdom	
6	C597029	84078	HANGING HEART JAW FLIGHT	-1	2010-12-21 12:30:00 UTC	1.25	18176	United Kingdom	
7	C597029	21480	RETROSPOT HEART HOT WAT	-1	2010-12-21 12:30:00 UTC	4.95	18176	United Kingdom	
8	C597029	22052	BROCCANTE SHELF WITH-HOOKS	-2	2010-12-21 12:30:00 UTC	10.75	18176	United Kingdom	
9	C543020	21217	RED RETROSPOT ROUND CAL	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom	
10	C543020	21024	DARRY BARD LARGE WICK JOG	-1	2011-02-17 14:24:00 UTC	4.95	14081	United Kingdom	

Results per page

50

1 – 50 of 100

<

>

- 구매 건 상태가 Canceled 인 데이터의 비율(%) - 소수점 첫번째 자리까지

```
-- 구매 건 상태가 Canceled 인 데이터의 비율(%)
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) *100, 1)
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```

Query results

SAVE RESULTS

EXECUTE

<

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTING

Row	f0_	
1	2.2	

Results per page: 50 1 - 1 of 1

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
-- 고유한 StockCode의 개수를 출력
SELECT count(DISTINCT StockCode)
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```

Query results

SAVE RESULTS

<

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTING

Row	f0_	
1	3684	

Results per page: 50 1 - 1 of 1

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
 - 상위 10개의 제품들을 출력하기

```
-- StockCode 별 등장 빈도를 출력(상위10개)
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

Query results

- StockCode 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고

- 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 출력하기

```
-- 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지를 확인
SELECT DISTINCT StockCode, number_count
FROM (
    SELECT StockCode,
           LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
)
WHERE number_count = 0
OR number_count = 1;
```

Query results [SAVE RESULTS](#) [EXPORT](#)

<	JOB INFORMATION	RESULTS	CHART	JSON	EXECUTING
Row	StockCode	number_count			
1	POST	0			
2	M	0			
3	PADS	0			
4	D	0			
5	BANK CHARGES	0			
6	DOT	0			
7	CRUK	0			
8	C2	1			

Results per page: 50 1 - 8 of 8

- StockCode**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
-- 해당 코드 값들을 가지고 있는 데이터 수가 전체 데이터 에서 차지하는 퍼센트
SELECT
    ROUND(SUM(CASE WHEN number_count = 0 OR number_count = 1 THEN 1 ELSE 0 END)/COUNT(*)*100, 2) AS
percentage
FROM (
    SELECT StockCode,
           LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
);
```

Query results [SAVE RESULTS](#) [EXPORT](#)

<	JOB INFORMATION	RESULTS	CHART	JSON	EXECUTING
Row	percentage				
1	0.48				

Results per page: 50 1 - 1 of 1

- 제품과 관련되지 않은 거래 기록을 제거하기

```
-- 제품과 관련되지 않은 거래 기록을 제거
DELETE FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE StockCode IN (
```

```
SELECT DISTINCT StockCode
FROM ( SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
      FROM `uplifted-sphinx-439401-k5.modulabs_project.data`)
WHERE number_count = 0 OR number_count = 1
);
```

Query results	SAVE RESULTS	EXPLORE DATA
JOB INFORMATION	RESULTS	EXECUTION DETAILS
This statement removed 1,915 rows from data.		
GO TO TABLE		

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
-- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력
SELECT Description, COUNT(*) AS description_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY Description
LIMIT 30;
```

Query results	SAVE RESULTS	EXPLORE DATA
JOB INFORMATION	RESULTS	CHART
JOB INFORMATION	RESULTS	CHART
Row	Description	description_cnt
1	TRADITIONAL CHRISTMAS RIB...	272
2	PAPER CHAIN KIT 50'S CHRIST...	1013
3	CHRISTMAS CRAFT LITTLE FRI...	433
4	SCANDINAVIAN REDS RIBBONS	343
5	EMBROIDERED RIBBON REEL R...	49
6	SET OF 4 KNICK KNACK TINS ...	328
7	ASSORTED COLOUR MINI CAS...	372
8	FELTCRAFT PRINCESS LOLA D...	383
9	EMBROIDERED RIBBON REEL E...	87
10	ASSORTED COLOUR BIRD ORN...	1405
11	JAM MAKING SET WITH JARS	966
12	6 RIBBONS RUSTIC CHARM	747

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
-- 서비스 관련 정보를 포함하는 행들을 제거
-- 'Next Day Carriage'나 'High Resolution Image'를 포함하는 행 제거
DELETE
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE description IN ('Next Day Carriage', 'High Resolution Image');
```

Query results	SAVE RESULTS	EXPLORE DATA
JOB INFORMATION	RESULTS	EXECUTION DETAILS
This statement removed 83 rows from data.		
GO TO TABLE		

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
-- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT * EXCEPT (Description),
       UPPER(Description) AS Description
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS EXECUTION DETAILS EXECUTION GRAPH

i This statement replaced the table named data. GO TO TABLE

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
-- UnitPrice의 최솟값, 최댓값, 평균
SELECT min(UnitPrice) AS min_price,
       max(UnitPrice) AS max_price,
       avg(UnitPrice) AS avg_price
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```

Query results SAVE RESULTS EXPLORE DATA

< JOB INFORMATION RESULTS CHART JSON EXECUTION

Row	min_price	max_price	avg_price
1	0.0	649.5	2.904956757405...

Results per page: 50 1 - 1 of 1

- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균 구하기

```
-- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균
SELECT count(Quantity) AS cnt_quantity,
       min(Quantity) AS min_quantity,
       max(Quantity) AS max_quantity,
       avg(Quantity) AS avg_quantity
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE UnitPrice = 0;
```

Query results SAVE RESULTS EXPLORE DATA

< JOB INFORMATION RESULTS CHART JSON EXECUTION

Row	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

Results per page: 50 1 - 1 of 1

- `UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기

```
-- UnitPrice = 0 제거
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE UnitPrice != 0;
```

Query results				SAVE RESULTS	EXPLORE DATA	
JOB INFORMATION RESULTS EXECUTION DETAILS EXECUTION GRAPH						
This statement replaced the table named data.						GO TO TABLE

11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
-- InvoiceDate 컬럼을 연월일 자료형으로 변경
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```

Query results											Save Results	Explore Data						
Job Information											Results		Chart	JSON	Execution Details		Execution Graph	
Row	InvoiceID	InvoiceID	InvoiceDate	Quantity	UnitPrice	CustomerID	Country	Description										
1	2011-11-03	574001	2011-11-03 16:15:00 UTC	12	1.25	12544	Spain	TRADITIONAL CHRISTMAS REL.										
2	2011-11-03	574001	2011-11-03 16:15:00 UTC	4	2.75	12544	Spain	FELT CRAFT PRINCESS LOLA D.										
3	2011-11-03	574001	2011-11-03 16:15:00 UTC	6	2.1	12544	Spain	CHRISTMAS CRAFT LITTLE FRL.										
4	2011-11-03	574001	2011-11-03 16:15:00 UTC	12	1.25	12544	Spain	PINK BILLS FELD CRAFT TONG.										
5	2011-11-03	574001	2011-11-03 16:15:00 UTC	6	4.35	12544	Spain	JAM MAKING SET WITH JARS										
6	2011-11-03	574001	2011-11-03 16:15:00 UTC	6	2.95	12544	Spain	PAPER CHAIN KIT SETS CHRETI.										
7	2011-11-03	574001	2011-11-03 16:15:00 UTC	12	1.65	12544	Spain	4 PINKIES PLATEC CASHM										
8	2011-11-03	574001	2011-11-03 16:15:00 UTC	6	2.98	12544	Spain	EMBROIDERED HEBRON REL. E.										

Results per page: 501 - 50 of 301253

[<](#) [>](#)

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
-- 가장 최근 구매 일자
SELECT max(DATE(InvoiceDate)) OVER () AS most_recent_date,
       DATE(InvoiceDate) AS InvoiceDay,
       *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```

Query results													SAVE RESULTS	EXPLORE DATA					
JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EXECUTION GRAPH																			
Row	most_recent_date	InvoiceDay	InvoiceID	Quantity	UnitPrice	CustomerID	Country	Description											
1	2011-12-09	2011-03-08	586015	7937	96	2011-03-08 16:48:00 UTC	14325	United Kingdom	BLUE FLICK GLASS CANDLEH.										
2	2011-12-09	2011-12-05	586072	22540	3	2011-12-05 14:29:00 UTC	17420	United Kingdom	MMN JIGSAW CIRCUS PARADE										
3	2011-12-09	2011-11-18	572967	20648	12	2011-11-18 12:48:00 UTC	14952	United Kingdom	DISCO BALL CHRISTMAS DEC.										
4	2011-12-09	2011-10-12	579602	475960	6	2011-10-12 12:03:00 UTC	646	United Kingdom	PINK HAPPY BIRTHDAY BIRTH.										
5	2011-12-09	2011-11-13	576066	79621	1	2011-11-13 15:44:00 UTC	16898	United Kingdom	CHILLI LIGHTS										
6	2011-12-09	2010-12-23	539020	22797	2	2010-12-23 11:06:00 UTC	16485	United Kingdom	CHEST OF DRAWERS CINCIAL.										
7	2011-12-09	2011-09-19	567285	23919	12	2011-09-19 10:44:00 UTC	249	United Kingdom	BOX OF 4 MMN JIGSAW C.										
8	2011-12-09	2011-09-04	526238	20020	1	2011-09-04 10:07:00 UTC	16123	United Kingdom	GLASS SONGERS STORAGE J.										

Results per page

10

20

50

100

250

500

1000

1 - 8 of 8 RESULTS

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
-- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장
SELECT CustomerID, max(DATE(InvoiceDate)) AS most_recent_date
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EX

Row	CustomerID	most_recent_date
1	12544	2011-11-10
2	13568	2011-06-19
3	13824	2011-11-07
4	14080	2011-11-07
5	14336	2011-11-23
6	14592	2011-11-04
7	15104	2011-06-26

Results per page: 50 1 - 50 of 4362

- 가장 최근 일자(**most_recent_date**)와 유저별 마지막 구매일(**InvoiceDay**)간의 차이를 계산하기

```
-- 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
);
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS CHART JSON EXECUTION DETAILS EX

Row	CustomerID	recency
1	14080	32
2	17409	184
3	13059	249
4	16902	116
5	17673	1
6	13324	32
7	14110	3

Results per page: 50 1 - 50 of 4362

- 최종 데이터 셋에 필요한 데이터들을 각각 정제해서 이어붙이고 지금까지의 결과를 **user_r** 이라는 이름의 테이블로 저장하기

```
-- 지금까지의 결과를 user_r이라는 이름의 테이블로 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_r` AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
);
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS EXECUTION DETAILS EXECUTION GRAPH

This statement replaced the table named user_r. [GO TO TABLE](#)

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
-- 고객마다 고유한 InvoiceNo의 수
SELECT
  CustomerID,
  count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

	JOB INFORMATION	RESULTS	CHART	JSON	EXECUTION DETAILS	EXI
Row	CustomerID	purchase_cnt				
1	12544	2				
2	13568	1				
3	13824	5				
4	14080	1				
5	14336	4				
6	14592	3				
7	15104	3				

Results per page: 50 1 - 50 of 4362

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
-- 각 고객 별로 구매한 아이템의 총 수량
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

	JOB INFORMATION	RESULTS	CHART	JSON	EXECUTION DETAILS	EXI
Row	CustomerID	item_cnt				
1	12544	130				
2	13568	66				
3	13824	768				
4	14080	48				
5	14336	1759				
6	14592	407				
7	15104	633				

Results per page: 50 1 - 50 of 4362

- 전체 거래 건수 계산과 구매한 아이템의 총 수량 계산의 결과를 합쳐서 `user_rf` 라는 이름의 테이블에 저장하기

```
-- 1,2를 합쳐 user_rf라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    count(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
```

```

item_cnt AS (
SELECT
    CustomerID,
    sum(quantity) AS item_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID
)
-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
    pc.CustomerID,
    pc.purchase_cnt,
    ic.item_cnt,
    ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
    ON pc.CustomerID = ic.CustomerID
JOIN `uplifted-sphinx-439401-k5.modulabs_project.user_r` AS ur
    ON pc.CustomerID = ur.CustomerID;

```

Query results		SAVE RESULTS	EXPLORE DATA
JOB INFORMATION	RESULTS	EXECUTION DETAILS	EXECUTION GRAPH
<p>i This statement replaced the table named user_rf.</p>		GO TO TABLE	

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

-- 고객별 총 지출액 계산
-- 소수점 첫째 자리에서 반올림
SELECT
    CustomerID,
    round(sum(Quantity*UnitPrice)) AS user_total
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;

```

Query results		SAVE RESULTS	EXPLORE DATA
<	JOB INFORMATION	RESULTS	CHART
Row	CustomerID	user_total	
1	12544	300.0	
2	13568	187.0	
3	13824	1699.0	
4	14080	46.0	
5	14336	1615.0	
6	14592	558.0	
7	15104	969.0	

- 고객별 평균 거래 금액 계산

- 고객별 평균 거래 금액을 구하기 위해 1) `data` 테이블을 `user_rf` 테이블과 조인(LEFT JOIN) 한 후, 2) `purchase_cnt` 로 나누어서 3) `user_rfm` 테이블로 저장하기

```

-- user_rfm 테이블로 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_rfm` AS
SELECT

```

```

rf.CustomerID AS CustomerID,
rf.purchase_cnt,
rf.item_cnt,
rf.recency,
ut.user_total,
ut.user_total / purchase_cnt AS user_average
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    round(sum(Quantity*UnitPrice)) AS user_total
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;

```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS EXECUTION DETAILS EXECUTION GRAPH

This statement replaced the table named user_rfm. [GO TO TABLE](#)

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

/*RFM 통합 테이블 출력*/
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rfm`

```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS		EXECUTION GRAPH	
Row	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average		
1	12719	1	905	0	795.0	795.0		
2	18010	1	40	256	175.0	175.0		
3	15083	1	38	256	88.0	88.0		
4	12792	1	215	256	345.0	345.0		
5	15520	1	214	1	244.0	244.0		
6	14569	1	79	1	227.0	227.0		
7	13436	1	76	1	197.0	197.0		
8	13298	1	96	1	360.0	360.0		

Results per page: 50 1 - 50 of 4362

11-8. 추가 Feature 추출

1. 구매하는 제품의 다양성

- 1) 고객 별로 구매한 상품들의 고유한 수를 계산하기
- 2) `user_rfm` 테이블과 결과를 합치기
- 3) `user_data` 라는 이름의 테이블에 저장하기

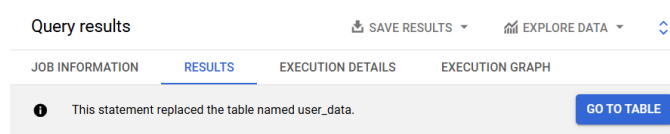
```

CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS

WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products

```

```
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

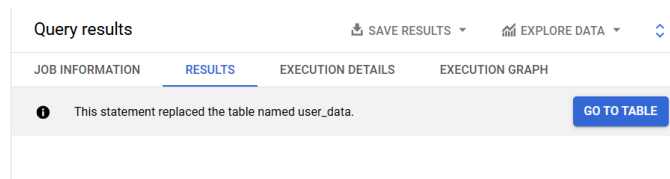


2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 평균 구매 소요 일수를 계산하고, 그 결과를 `user_data`에 통합

```
-- 평균 구매 소요 일수를 계산
-- 결과를 user_data에 통합
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_in
  terval
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DA
    Y) AS interval_
  FROM
    `uplifted-sphinx-439401-k5.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



3. 구매 취소 경향성

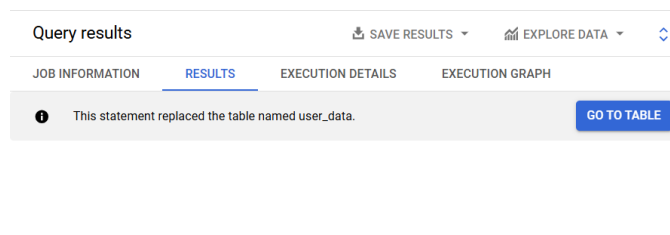
- 고객의 취소 패턴 파악하기

- 1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
- 2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
 - 취소 빈도와 취소 비율을 계산하고 그 결과를 **user_data**에 통합하기
(취소 비율은 소수점 두번째 자리)

```
-- 취소 빈도와 취소 비율을 계산
-- 결과를 user_data에 통합
-- 취소 비율은 소수점 두번째 자리까지 구하기
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(DISTINCT InvoiceNo) AS total_transactions,
    countif(DISTINCT InvoiceNo LIKE 'C%') AS cancel_frequency
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round(cancel_frequency / total_transactions* 100, 2) AS cancel_rate
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



- 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 **user_data**를 출력하기

```
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_data`
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS		EXECUTION GRAPH				
Row	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
46	15894	19	2513	0	3949.0	202.579473684	170	0.7	19	1	5.2%
47	15738	19	2967	18	4789.0	252.056315789	46	1.97	19	1	5.2%
48	16928	19	2071	80	2741.0	144.263157894	71	1.37	19	1	5.2%
49	15291	19	2073	25	4552.0	239.578947368	62	3.18	19	1	5.2%
50	12989	19	2915	3	6855.0	360.789473684	41	5.3	19	1	5.2%

Results per page: 501 - 50 of 4369

회고

중간중간에 데이터를 잘못 추출한 경우가 생겨 되돌려야할 때가 있었습니다. 코드를 정확히 작성하고 데이터를 백업해놓는 것이 중요함을 느꼈습니다.