

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `vital-program-456100-p3.modulabs_project.data`  
LIMIT 10
```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT *  
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과

결과 저장 다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Un
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	
11	536367	22745	POPPY'S PLAYHOUSE BEDROO...	6	2010-12-01 08:34:00 UTC	
12	536367	22748	POPPY'S PI AYHOUSE KITCHEN	6	2010-12-01 08:34:00 UTC	

페이지당 결과 수: 501 - 50 (전체 541909행)

데이터 수 세기

- COUNT 함수를 사용해서, 각 컬럼별 데이터 포인트의 수를 세어 보기

```
SELECT
COUNT(InvoiceNo) AS COUNT_InvoiceNo,
COUNT(StockCode) AS COUNT_StockCode,
COUNT(Description) AS COUNT_Description,
COUNT(Quantity) AS COUNT_Quantity,
COUNT(InvoiceDate) AS COUNT_InvoiceDate,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerID) AS COUNT_CustomerID,
COUNT(Country) AS COUNT_Country
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	IT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
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

```
SELECT
'InvoiceNo' AS column_name,
ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`

UNION ALL

SELECT
'StockCode' AS column_name,
ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`

UNION ALL

SELECT
'Description' AS column_name,
ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`

UNION ALL

SELECT
'Quantity' AS column_name,
ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`

UNION ALL

SELECT
'InvoiceDate' AS column_name,
ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`
```

UNION ALL

```
SELECT
  'UnitPrice' AS column_name,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`
```

UNION ALL

```
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`
```

UNION ALL

```
SELECT
  'Country' AS column_name,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과

작업 정보				결과	차트	JSON	실행 세부정보
행	column_name	missing_percentage					
1	CustomerID	24.93					
2	UnitPrice	0.0					
3	Description	0.27					
4	InvoiceNo	0.0					
5	Country	0.0					
6	InvoiceDate	0.0					
7	Quantity	0.0					
8	StockCode	0.0					

결측치 처리 전략

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

```
SELECT Description
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE StockCode = '85123A'
GROUP BY Description
```

쿼리 결과


작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	Description ▾				
1	WHITE HANGING HEART T-LIGHT HOLDER				
2	?				
3	wrongly marked carton 22804				
4	CREAM HANGING HEART T-LIGHT HOLDER				

결측치 처리

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

```
DELETE FROM `vital-program-456100-p3.modulabs_project.data`  
WHERE CustomerID IS NULL OR Description IS NULL
```

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
	 이 문으로 data의 행 135,080개가 삭제되었습니다.		

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
WITH frequency AS (  
  SELECT *,  
    COUNT(*) AS frequency  
  FROM `vital-program-456100-p3.modulabs_project.data`  
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country  
)  
SELECT *  
FROM frequency  
WHERE frequency.frequency > 1
```

쿼리 결과					
결과 저장					
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	555524	22698	PINK REGENCY TEACUP AND ...	1	2011-06-05 1
2	555524	22697	GREEN REGENCY TEACUP AN...	1	2011-06-05 1
3	572861	22775	PURPLE DRAWERKNOB ACRYL...	12	2011-10-26 1
4	572344	M	Manual	48	2011-10-24 1
5	538514	21756	BATH BUILDING BLOCK WORD	1	2010-12-12 1

페이지당 결과 수: 50 1 ~ 50 (전체 4837행)

중복값 처리

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

```
CREATE OR REPLACE TABLE vital-program-456100-p3.modulabs_project.data AS
SELECT DISTINCT *
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과					
작업 정보 결과 실행 세부정보 실행 그래프					
이 문으로 이름이 data인 테이블이 교체되었습니다.					

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT InvoiceNo, COUNT(InvoiceNo) AS unique_InvoiceNo
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY InvoiceNo
```

쿼리 결과

결과 저장 ▼

다음에

작업 정보 **결과** 차트 JSON 실행 세부정보

행	InvoiceNo ▼	unique_InvoiceNo ▼
1	541431	1
2	C541433	1
3	537626	31
4	542237	29
5	549222	24
6	556201	18
7	562032	22
8	573511	47
9	581180	11
10	539318	17
11	541998	6
12	548955	5

- 고유한 **InvoiceNo** 를 앞에서부터 100개를 출력하기

```
SELECT InvoiceNo
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY InvoiceNo
LIMIT 100
```

쿼리 결과

작업 정보	결과	차트	JSON
행	InvoiceNo		
1	541431		
2	C541433		
3	537626		
4	542237		
5	549222		
6	556201		
7	562032		
8	573511		
9	581180		
10	539318		
11	541998		
12	548955		

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

```
SELECT *
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE InvoiceNo LIKE "C%"
LIMIT 100
```

쿼리 결과								
작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프	결과 저장	다음에서 열기	
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	84050	PINK HEART SHAPE EGG FRYL...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
9	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
10	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
11	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
12	C549955	22666	RECIPE BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC	2.95	12359	Cyprus

페이지당 결과 수: 50 ▼ 1 - 50 (전체 100행) |< < > >

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(InvoiceNo) * 100, 1)
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과

[결과](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_				
1	2.2				

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
SELECT DISTINCT StockCode
FROM `vital-program-456100-p3.modulabs_project.data`
```

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM project_name.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

페이지당 결과 수: 50 1 - 10 (전체 10행)

- 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

```
SELECT DISTINCT StockCode, number_count
FROM (
```



```
SELECT StockCode,
       LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `vital-program-456100-p3.modulabs_project.data`
)
WHERE number_count <= 1
```

쿼리 결과

[결과 저장 ▼](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	StockCode ▼	number_count ▼			
1	POST	0			
2	M	0			
3	C2	1			
4	D	0			
5	BANK CHARGES	0			
6	PADS	0			
7	DOT	0			
8	CRUK	0			

- **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)

```
SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(StockCode) * 100, 2)
FROM (
  SELECT StockCode,
         LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `vital-program-456100-p3.modulabs_project.data`
)
```

쿼리 결과

[결과 저장 ▼](#) [다음에서 열기 ▼](#) [↕](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_ ▼				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.data`
  )
)
```

```
WHERE number_count <= 1
)
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 30
```

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG SPACEBOY DESIGN	1006

페이지당 결과 수:

50

1 - 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE FROM `vital-program-456100-p3.modulabs_project.data`
WHERE
  LENGTH(REGEXP_REPLACE(Description, r'[A-Z]', '')) - LENGTH(REGEXP_REPLACE(Description, r'[a-z]', '')) > 0
```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

실행 세부정보

실행 그래프

이 문으로 data의 행 83개가 삭제되었습니다.

테이블로 이동

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과	결과 저장	다음에서 열기	
작업 정보	결과	실행 세부정보	실행 그래프
<div>이 문으로 이름이 data인 테이블이 교체되었습니다.</div> <div>테이블로 이동</div>			

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	min_price	max_price	avg_price	
1	0.0	649.5	2.904956757406...	

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

```
SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS avg_quantity
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE UnitPrice = 0
```

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

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

- UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.data` AS
SELECT *
FROM `vital-program-456100-p3.modulabs_project.data`
WHERE UnitPrice != 0.0
```

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다. 테이블로 이동

11-7. RFM 스코어

Recency

- InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `vital-program-456100-p3.modulabs_project.data`
```

쿼리 결과 결과 저장 다음에서 열기

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
2	2011-01-18	C541433	23166	-74215	2011-01-18
3	2010-12-07	537626	21064	6	2010-12-07
4	2010-12-07	537626	22726	4	2010-12-07
5	2010-12-07	537626	85167B	30	2010-12-07
6	2010-12-07	537626	22772	12	2010-12-07
7	2010-12-07	537626	22805	12	2010-12-07
8	2010-12-07	537626	22728	4	2010-12-07
9	2010-12-07	537626	84997D	6	2010-12-07

페이지당 결과 수: 50 1 - 50 (전체 399573행) |< < > >|

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT MAX(DATE(InvoiceDate)) AS most_recent_date
FROM `vital-program-456100-p3.modulabs_project.data`
```

```
-- 예시에서
-- # [[YOUR QUERY]] AS InvoiceDay,
-- *
-- 이 두 줄을 어떻게 살려야 할 지 모르겠어요;
```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	most_recent_date			
1	2011-12-09			

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT CustomerID,
       MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
```

쿼리 결과

결과 저장 ▼



작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08

페이지당 결과 수: 50 ▼ 1 – 50 (전체 4362행)

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

```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
```

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
);
```

쿼리 결과 결과 저장

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	recency
1	12474	17
2	12540	19
3	12545	75
4	12719	5
5	12962	7
6	13067	82
7	13109	2
8	13137	10
9	13220	78
10	13316	37
11	13505	67
12	13710	31

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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

```
CREATE OR REPLACE TABLE `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.data`
  GROUP BY CustomerID
);
```

쿼리 결과 결과 저장 다음에서 열기

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다. 테이블로 이동

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
```

```
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
```

쿼리 결과 결과 저장

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
11	12357	1
12	12358	2

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID,
SUM(Quantity) AS item_cnt
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
```

쿼리 결과 결과 저장

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573
11	12357	2708
12	12358	242

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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

```
CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
```

```

SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID,
  SUM(Quantity) AS item_cnt
FROM `vital-program-456100-p3.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 `vital-program-456100-p3.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;

```

쿼리 결과	결과 저장	다음에서 열기
작업 정보	결과	실행 세부정보
이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.		테이블로 이동

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  ROUND(SUM(Quantity * Unitprice), 1) AS user_total
FROM `vital-program-456100-p3.modulabs_project.data`
GROUP BY CustomerID

```


쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 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

```

CREATE OR REPLACE TABLE `vital-program-456100-p3.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ut.user_total / rf.purchase_cnt AS user_average -- 여기 다시 해석해보기!
FROM `vital-program-456100-p3.modulabs_project.user_rf` AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * Unitprice), 1) AS user_total
  FROM `vital-program-456100-p3.modulabs_project.data`
  GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;

```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

실행 세부정보

실행 그래프

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

테이블로 이동

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *  
FROM `vital-program-456100-p3.modulabs_project.user_rfm`
```

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프			
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average		
1	12713	1	505	0	794.5	794.5		
2	13436	1	76	1	196.9	196.9		
3	15520	1	314	1	343.5	343.5		
4	13298	1	96	1	360.0	360.0		
5	14569	1	79	1	227.4	227.4		
6	15195	1	1404	2	3861.0	3861.0		
7	15471	1	256	2	454.5	454.5		
8	14204	1	72	2	150.6	150.6		
9	14578	1	240	3	168.6	168.6		
10	12650	1	250	3	242.4	242.4		
11	17914	1	457	3	329.4	329.4		
12	15318	1	642	3	312.6	312.6		

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

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 vital-program-456100-p3.modulabs_project.user_data AS  
WITH unique_products AS (  
  SELECT  
    CustomerID,  
    COUNT(DISTINCT StockCode) AS unique_products  
  FROM vital-program-456100-p3.modulabs_project.data  
  GROUP BY CustomerID  
)  
SELECT ur.*, up.* EXCEPT (CustomerID)  
FROM vital-program-456100-p3.modulabs_project.user_rfm AS ur  
JOIN unique_products AS up  
ON ur.CustomerID = up.CustomerID;
```

쿼리 결과			
작업 정보	결과	실행 세부정보	실행 그래프
<div> <div></div> <div>이 문으로 이름이 user_data인 테이블이 교체되었습니다.</div> </div>			

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE vital-program-456100-p3.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_interval
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS interval_
    FROM
      vital-program-456100-p3.modulabs_projectt.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM vital-program-456100-p3.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

<div> <div>←</div> <div>쿼리 결과</div> </div>			
이(으)로 돌아가기	결과	실행 세부정보	실행 그래프
<div> <div></div> <div>이 문으로 이름이 user_data인 테이블이 교체되었습니다.</div> </div>			

3. 구매 취소 경향성

- 고객의 취소 패턴 파악하기
 - 취소 빈도(`cancel_frequency`) : 고객 별로 취소한 거래의 총 횟수
 - 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)

```
CREATE OR REPLACE TABLE vital-program-456100-p3.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
  FROM vital-program-456100-p3.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(cancel_frequency / total_transactions * 100, 1) AS cancel_rate
FROM `vital-program-456100-p3.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

쿼리 결과	결과 저장	다음에서 열기
작업 정보	결과	실행 세부정보
이 문으로 이름이 user_data인 테이블이 교체되었습니다.		테이블로 이동

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

```
SELECT *
FROM vital-program-456100-p3.modulabs_project.user_data
```

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	omerid	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	total_transactions	cancel_frequency	cancel_rate	
42	15100	6	58	330	635.1	105.85000000000...	1	6	3	50.0	
43	16878	2	0	24	0.0	0.0	3	6	3	50.0	
44	17900	4	0	164	0.0	0.0	3	10	5	50.0	
45	13762	2	0	204	0.0	0.0	7	14	7	50.0	
46	13364	2	0	66	0.0	0.0	10	20	10	50.0	
47	12558	2	0	1	0.0	0.0	11	22	11	50.0	
48	18274	2	0	17	0.0	0.0	11	22	11	50.0	
49	12454	2	0	53	0.0	0.0	15	30	15	50.0	
50	14557	2	0	64	0.0	0.0	16	32	16	50.0	

페이지당 결과 수: 50

1 - 50 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : 손으로 직접 최대한 치면서 하려고 노력했다. 포기하지 않고 열심히 했다.

Problem : 가이드에서 몇 가지 해석이 어려운 것들이 아직 남아 있다. (문법을 몰라서 그런 거 말고 해석) 백업을 안 했는데 다행히 자료 날아간 일은 없지만 백업 빠뜨리지 않을 것!

Try : 분석 전략을 가질 때 너무 돌아가지 않도록 조금 심플하게 생각해보는 것도 필요!