

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM lunar-planet-425501-t3.modulabs_project.data  
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

[결과 저장](#) [테이블](#)

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프		
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	null	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) AS row_count  
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	row_count				
1	541909				

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 lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

 결과 저장 ▾

 데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프			
행	COUNT_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 InvoiceNo,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'StockCode' AS StockCode,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'Description' AS Description,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'Quantity' AS Quantity,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'InvoiceDate' AS InvoiceDate,
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
```

```

SELECT
  'UnitPrice' AS UnitPrice,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'CustomerID' AS CustomerID,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data
UNION ALL
SELECT
  'Country' AS Country,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM lunar-planet-425501-t3.modulabs_project.data;

```

[결과 이미지를 넣어주세요]

쿼리 결과

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

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE StockCode = '85123A';

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	Description ▼			
1	?			
2	wrongly marked carton 22804			
3	CREAM HANGING HEART T-LIG...			
4	WHITE HANGING HEART T-LIG...			


결측치 처리

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

```
DELETE FROM lunar-planet-425501-t3.modulabs_project.data
WHERE Description IS NULL
OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]

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 duplicates AS (
  SELECT (CASE WHEN COUNT(*) > 1 THEN 1 ELSE 0 END) as duplicate
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
)
SELECT SUM(duplicate) as dup_count
FROM duplicates;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트
행	dup_count ▼	
1	4837	

중복값 처리

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

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.data AS
SELECT DISTINCT *
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 **InvoiceNo** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) as Unique_InvoiceNo
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보		결과	차트
행		Unique_InvoiceNo	
1		22190	

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

```
SELECT DISTINCT InvoiceNo
FROM lunar-planet-425501-t3.modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과					
작업 정보		결과	차트	JSON	실행 세부정보
행		InvoiceNo			
1		574301			
2		C575531			
3		557305			
4		543008			
5		549735			
6		554032			
7		561387			
8		574868			
9		574827			

페이지당 결과 수: 50 1 - 50 (전체 100행)

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

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE InvoiceNo LIKE "C%"
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탭

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
7	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
8	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
9	C543620	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom

페이지당 결과 수:

50

1 - 50 (전체 100행)

<

>

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE "C%" THEN 1 ELSE 0 END)/ COUNT(InvoiceNo) * 100, 1) AS Pct_Car
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

차

행	Pct_Cancelled	
1	2.2	

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS Unique_StockCode
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

행	Unique_StockCode
1	3684

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

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
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		

- 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 lunar-planet-425501-t3.modulabs_project.data
)
WHERE number_count = 1
OR number_count = 0;
```


[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	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		

- **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)

```
WITH StockCodes AS (  
  SELECT StockCode, number_count  
  FROM (  
    SELECT StockCode,  
           LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count  
    FROM lunar-planet-425501-t3.modulabs_project.data  
  )  
)  
SELECT ROUND(SUM(CASE WHEN number_count < 5 THEN 1 ELSE 0 END) / COUNT(StockCode) * 100, 2) AS StockCode_  
FROM StockCodes;  
  
""  
OR  
  
SELECT ROUND(SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 5 THEN  
FROM lunar-planet-425501-t3.modulabs_project.data;  
""
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트
행	StockCode_OutlierRatio	
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM lunar-planet-425501-t3.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 lunar-planet-425501-t3.modulabs_project.data
  )
  WHERE number_count < 5
);

...
OR

DELETE FROM lunar-planet-425501-t3.modulabs_project.data
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) < 5
...
```

[결과 이미지를 넣어주세요]

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 lunar-planet-425501-t3.modulabs_project.data
GROUP BY Description
LIMIT 30;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	Description	description_cnt			
1	SET OF 6 RIBBONS VINTAGE C...	343			
2	EMBROIDERED RIBBON REEL R...	49			
3	PAPER CHAIN KIT 50'S CHRIST...	1013			
4	SCANDINAVIAN REDS RIBBONS	343			
5	EMBROIDERED RIBBON REEL S...	63			
6	ASSORTED COLOUR BIRD ORN...	1405			
7	ASSORTED COLOUR MINI CAS...	372			
8	6 RIBBONS RUSTIC CHARM	747			
9	SET OF 4 KNICK KNACK TINS ...	328			
10	PAPER CHAIN KIT VINTAGE C...	718			
11	FELTCRAFT PRINCESS LOLA D...	383			
12	PINK BLUE FELT CRAFT TRINK...	465			
13	FELTCRAFT PRINCESS OLIVIA ...	254			
14	ADDITIONAL CUDIETAAAC DIB	272			

페이지당 결과 수: 50 1 ~ 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE Description = 'Next Day Carriage'
OR Description = 'High Resolution Image';
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 data의 행 83개가 삭제되었습니다.</p>			

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
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 lunar-planet-425501-t3.modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

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 = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.modulabs_project.data AS
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.data
WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceDay	InvoiceNo	StockCode	Quantity	UnitPrice	CustomerID	Country	Description
1	2011-11-03	574301	23511	6	2.08	12544	Spain	EMBROIDERED RIBBON REEL E
2	2011-11-03	574301	22734	6	2.89	12544	Spain	SET OF 6 RIBBONS VINTAGE C
3	2011-11-03	574301	23512	6	2.08	12544	Spain	EMBROIDERED RIBBON REEL R
4	2011-11-03	574301	20749	4	7.95	12544	Spain	ASSORTED COLOUR MINI CAS.
5	2011-11-03	574301	22086	6	2.95	12544	Spain	PAPER CHAIN KIT 50'S CHRIST
6	2011-11-03	574301	85049A	12	1.25	12544	Spain	TRADITIONAL CHRISTMAS RIB
7	2011-11-03	574301	85049E	12	1.25	12544	Spain	SCANDINAVIAN REDS RIBBONS
8	2011-11-03	574301	22621	12	1.65	12544	Spain	TRADITIONAL KNITTING NANC
9	2011-11-03	574301	22750	4	3.75	12544	Spain	FELTCRAFT PRINCESS LOLA D.
10	2011-11-03	574301	22960	6	4.25	12544	Spain	JAM MAKING SET WITH JARS

페이지당 결과 수: 50

1 ~ 50 (전체 399573행)

<<

<

>

>>

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
    MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
    DATE(InvoiceDate) AS InvoiceDay,
    *
FROM lunar-planet-425501-t3.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09	2010-12-01	536412	21034	1	2010-12-01 11:49:00 UTC	0.95	17920	United Kingdom
2	2011-12-09	2011-12-01	580119	23010	1	2011-12-01 16:37:00 UTC	16.95	17920	United Kingdom
3	2011-12-09	2011-07-14	560104	23093	2	2011-07-14 18:56:00 UTC	2.49	17921	United Kingdom
4	2011-12-09	2011-10-12	570832	475908	6	2011-10-12 13:28:00 UTC	5.45	13058	United Kingdom
5	2011-12-09	2011-11-13	576066	23194	1	2011-11-13 15:44:00 UTC	2.25	16898	United Kingdom
6	2011-12-09	2011-11-02	574049	23085	2	2011-11-02 13:40:00 UTC	10.4	13316	United Kingdom
7	2011-12-09	2011-05-18	553661	23073	2	2011-05-18 11:59:00 UTC	12.5	13572	United Kingdom
8	2011-12-09	2011-08-09	562788	23203	100	2011-08-09 13:48:00 UTC	1.79	17669	United Kingdom
9	2011-12-09	2011-11-23	578304	82483	8	2011-11-23 15:41:00 UTC	6.95	13831	United Kingdom

페이지당 결과 수: 50

1 ~ 50 (전체 399573행)

이전

다음

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	InvoiceDay			
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			
8	15360	2011-10-31			
9	15872	2011-11-25			

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 project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	17153	24			
2	16386	28			
3	17415	79			
4	13584	306			
5	15120	129			
6	18192	71			
7	16401	1			
8	17682	10			
9	16921	61			
10	16156	8			

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 lunar-planet-425501-t3.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 lunar-planet-425501-t3.modulabs_project.data
GROUP BY CustomerID
)
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt			
1	12544	2			
2	13568	1			
3	13824	5			
4	14080	1			
5	14336	4			
6	14592	3			
7	15104	3			
8	15360	1			
9	15872	2			
10	16128	5			
11	16384	2			

페이지당 결과 수: 50 ▼ 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  COUNT(StockCode) AS item_cnt
```



```
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과

[결과 저장](#) ▼

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▼	item_cnt ▼			
1	12544	19			
2	13568	43			
3	13824	46			
4	14080	4			
5	14336	90			
6	14592	158			
7	15104	69			
8	15360	13			
9	15872	108			
10	16128	89			
11	16384	33			

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 lunar-planet-425501-t3.modulabs_project.user_rf AS
```

```
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY 1
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    COUNT(StockCode) AS item_cnt
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY 1
)
```

```
-- 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 lunar-planet-425501-t3.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice),1) AS user_total
FROM lunar-planet-425501-t3.modulabs_project.data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

쿼리 결과

[결과 저장](#)

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	user_total			
1	12544	299.7			
2	13568	187.1			
3	13824	1698.9			
4	14080	45.6			
5	14336	1614.9			
6	14592	557.9			
7	15104	968.6			
8	15360	427.9			
9	15872	316.2			

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 lunar-planet-425501-t3.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(ut.user_total/rf.purchase_cnt) AS user_average
FROM lunar-planet-425501-t3.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY 1
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

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	14090	1	1	324	76.0	76.0
2	16148	1	1	296	76.0	76.0
3	18141	1	1	360	-35.0	-35.0
4	12791	1	1	373	178.0	178.0
5	12943	1	1	301	-4.0	-4.0
6	17291	1	1	308	551.0	551.0
7	15562	1	1	351	135.0	135.0
8	13703	1	1	318	100.0	100.0

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 lunar-planet-425501-t3.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM lunar-planet-425501-t3.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

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

```
CREATE OR REPLACE TABLE lunar-planet-425501-t3.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
    FROM
      lunar-planet-425501-t3.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM lunar-planet-425501-t3.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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 lunar-planet-425501-t3.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
  FROM lunar-planet-425501-t3.modulabs_project.data
  GROUP BY 1
)
SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND((t.cancel_frequency / t.total_transactions * 100), 2) AS cancel_rate
FROM `lunar-planet-425501-t3.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```
SELECT *
FROM lunar-planet-425501-t3.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보		결과	차트		JSON	실행 세부정보		실행 그래프			
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13120	1	1	238	31.0	31.0	1	0.0	1	0	0.0
2	17060	1	5	266	234.0	234.0	5	0.0	1	0	0.0
3	17844	1	5	106	52.0	52.0	5	0.0	1	0	0.0
4	12831	1	9	262	215.0	215.0	9	0.0	1	0	0.0
5	17110	1	12	165	163.0	163.0	12	0.0	1	0	0.0
6	13436	1	12	1	197.0	197.0	12	0.0	1	0	0.0
7	14689	1	13	208	113.0	113.0	13	0.0	1	0	0.0
8	14799	1	13	51	158.0	158.0	13	0.0	1	0	0.0
9	12812	1	15	44	230.0	230.0	15	0.0	1	0	0.0
10	18105	1	18	37	113.0	113.0	18	0.0	1	0	0.0
11	15016	1	20	172	170.0	170.0	20	0.0	1	0	0.0
12	17508	1	22	280	387.0	387.0	22	0.0	1	0	0.0
13	12447	1	25	243	431.0	431.0	25	0.0	1	0	0.0

페이지당 결과 수: 50 1 - 50 (전체 4362행) < >

[회고]

- LMS에서 설명된 내용을 따라하면서 지금까지 배웠던 리눅스, 깃헙, SQL을 사용하면서 복습할 수 있었습니다.
- 지난 노트에서 배운 RFM의 주 축인 recency, frequency, monetary외에도 구매하는 제품군의 다양성, 평균 구매 주기, 구매 취소 경향성을 통해 고객군을 더 세분화해서 나눌 수 있음을 배웠습니다.
- 앞에서 좀 더 간단하게 SQL을 사용할 수 있었는데, 노트 상에서 # [[YOUR QUERY]]가 있는 부분을 수정하는 것이라보니 몇 개는 복잡한 버전과 좀 더 간단한 버전 두가지로 적어봤습니다.
- 내일 새로 배울 노트에서 어떻게 파이썬을 사용해 데이터를 가시화할 수 있을지 기대됩니다.