

# 고객을 세그멘테이션하자 [프로젝트] 정선아

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM proven-serenity-439401-f2.modulabs_project.data  
LIMIT 10
```

쿼리 결과

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	20958	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 total_rows  
FROM proven-serenity-439401-f2.modulabs_project.data
```

### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	total_rows				
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  
  proven-serenity-439401-f2.modulabs_project.data
```

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

```
-- InvoiceNo에 대한 결측치 비율 계산
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- StockCode에 대한 결측치 비율 계산
SELECT
  'StockCode' AS column_name,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- Description에 대한 결측치 비율 계산
SELECT
  'Description' AS column_name,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- Quantity에 대한 결측치 비율 계산
SELECT
  'Quantity' AS column_name,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- InvoiceDate에 대한 결측치 비율 계산
SELECT
  'InvoiceDate' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- UnitPrice에 대한 결측치 비율 계산
SELECT
  'UnitPrice' AS column_name,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- CustomerID에 대한 결측치 비율 계산
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
```

```

FROM
    proven-serenity-439401-f2.modulabs_project.data
UNION ALL
-- Country에 대한 결측치 비율 계산
SELECT
    'Country' AS column_name,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
    proven-serenity-439401-f2.modulabs_project.data;

```

## 쿼리 결과

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

## 결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM proven-serenity-439401-f2.modulabs_project.data
WHERE StockCode = '85123A';

```

## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
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 proven-serenity-439401-f2.modulabs_project.data
WHERE Description IS NULL OR CustomerID IS NULL;
```

## 쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프

**i** 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

```
SELECT COUNT(*) AS duplicate_count
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS count_per_row
  FROM
    proven-serenity-439401-f2.modulabs_project.data
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1 -- 중복된 행 필터링
) AS duplicate_rows;
```

## 쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	duplicate_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 proven-serenity-439401-f2.modulabs_project.data AS
SELECT DISTINCT *
FROM proven-serenity-439401-f2.modulabs_project.data ;
```

### 쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 data인 테이블이 교체되었습니다.

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 **InvoiceNo**의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM proven-serenity-439401-f2.modulabs_project.data
```

### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	unique_invoice_coun				
1	22190				

- 고유한 **InvoiceNo**를 앞에서부터 100개를 출력하기

```
SELECT InvoiceNo AS unique_invoice
FROM proven-serenity-439401-f2.modulabs_project.data
LIMIT 100;
```

## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	unique_invoice ▼				
88	549735				
89	554032				
90	561387				
91	561387				
92	561387				
93	561387				
94	561387				
95	561387				
96	561387				
97	561387				
98	561387				
99	561387				
100	574868				

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

```
SELECT *
FROM proven-serenity-439401-f2.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

쿼리 결과

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	47590A	BLUE HAPPY BIRTHDAY BUNTL	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590B	PINK 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
10	C546858	21534	DAIRY MAID LARGE MILK JUG	-1	2011-03-17 14:24:00 UTC	4.95	14081	United Kingdom
11	C546858	22839	3 TIER CAKE TIN GREEN AND ..	-1	2011-03-17 14:24:00 UTC	14.95	14081	United Kingdom
12	C542263	22699	ROSES REGENCY TEACUP AND..	-1	2011-01-26 17:16:00 UTC	2.95	14849	United Kingdom

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/COUNT(*) * 100, 1) AS canceled_percent
FROM proven-serenity-439401-f2.modulabs_project.data
```

## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	canceled_percentage ▼				
1	2.2				

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM proven-serenity-439401-f2.modulabs_project.data
```

### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	unique_stockcode_count ▼				
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 proven-serenity-439401-f2.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

### 쿼리 결과

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			

- 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 proven-serenity-439401-f2.modulabs_project.data
) AS stock_with_number_count
WHERE number_count <= 1;
```

### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
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 StockCodesWithFewNumbers AS (
  SELECT StockCode
  FROM `proven-serenity-439401-f2.modulabs_project.data`
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1 -- 숫자가 0개 또는 1개인
)
SELECT
  ROUND(COUNT(*) / (SELECT COUNT(*) FROM `proven-serenity-439401-f2.modulabs_project.data`) * 100, 2) AS
FROM
  `proven-serenity-439401-f2.modulabs_project.data`
WHERE
  StockCode IN (SELECT StockCode FROM StockCodesWithFewNumbers);
```

### 쿼리 결과

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

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM proven-serenity-439401-f2.modulabs_project.data_re
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM proven-serenity-439401-f2.modulabs_project.data_re
```



```
WHERE StockCode IN ('POST', 'BANK CHARGES', 'PADS', 'DOT', 'CARRIAGE', 'D', 'M', 'CRUK') -- 제품과 관련이 없는 코드
) AS non_product_codes
);
```

## 쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

**i** 이 문으로 data의 행 1,977개가 삭제되었습니다.

처음부터 데이터셋 새로 올려서 다시 수행했는데로 아래와 같이 나옴

## 쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

**i** 이 문으로 data\_re의 행 1,781개가 삭제되었습니다.

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

## 쿼리 결과

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

더보기

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM proven-serenity-439401-f2.modulabs_project.data_re
WHERE
    Description LIKE '%Next Day Carriage%' OR
    Description LIKE '%High Resolution Image%';
```

— 캡처 못해서, 다시 실행 시 삭제 행이 '0'으로 나옴

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE proven-serenity-439401-f2.modulabs_project.data_re AS
SELECT
    * EXCEPT (Description),
    UPPER(Description) AS Description -- Description을 대문자로 변환
FROM proven-serenity-439401-f2.modulabs_project.data_re;
```

```
198 -- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화
199 CREATE OR REPLACE TABLE proven-serenity-439401-f2.modulabs_project.data_re AS
200 SELECT
201     * EXCEPT (Description),
202     UPPER(Description) AS Description -- Description을 대문자로 변환
203 FROM proven-serenity-439401-f2.modulabs_project.data_re;
204
```

쿼리 결과

결과

작업 정보   **결과**   실행 세부정보   실행 그래프

이 문으로 이름이 data\_re인 테이블이 교체되었습니다.

## UnitPrice 살펴보기

- UnitPrice 의 최소값, 최댓값, 평균을 구하기

```
SELECT
    MIN(UnitPrice) AS min_price,
    MAX(UnitPrice) AS max_price,
    AVG(UnitPrice) AS avg_price
FROM proven-serenity-439401-f2.modulabs_project.data_re;
```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	min_price ▼	max_price ▼	avg_price ▼	
1	0.0	649.5	2.920788887777...	

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

```
SELECT
  COUNT(*) AS cnt_quantity,      -- 단가가 0원인 거래의 개수
  MIN(Quantity) AS min_quantity, -- 구매 수량의 최솟값
  MAX(Quantity) AS max_quantity, -- 구매 수량의 최댓값
  AVG(Quantity) AS avg_quantity  -- 구매 수량의 평균값
FROM proven-serenity-439401-f2.modulabs_project.data_re
WHERE UnitPrice = 0;            -- 단가가 0원인 거래 필터링
```

#### 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	cnt_quantity	min_quantity	max_quantity	avg_quantity	
1	33	1	12540	420.5151515151515	

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE proven-serenity-439401-f2.modulabs_project.data_re AS
SELECT *
FROM proven-serenity-439401-f2.modulabs_project.data_re
WHERE UnitPrice != 0;
```

#### 쿼리 결과

작업 정보      **결과**      실행 세부정보      실행 그래프

**i** 이 문으로 이름이 data\_re인 테이블이 교체되었습니다.

스키마	세부정보	미리보기	데이터를 탐색기	미리보기	통계	계보	데이터 프로파일	데이터 품질
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate			
1	574301	85049E	12	132033690...	1970-01-01 00:00:01.250 UTC			
2	574301	85049A	12	132033690...	1970-01-01 00:00:01.250 UTC			
3	574301	22910	6	132033690...	1970-01-01 00:00:02.950 UTC			
4	574301	23514	6	132033690...	1970-01-01 00:00:02.080 UTC			
5	574301	22086	6	132033690...	1970-01-01 00:00:02.950 UTC			
6	574301	20749	4	132033690...	1970-01-01 00:00:07.950 UTC			
7	574301	23240	6	132033690...	1970-01-01 00:00:04.150 UTC			
8	574301	20971	12	132033690...	1970-01-01 00:00:01.250 UTC			
9	574301	22751	4	132033690...	1970-01-01 00:00:03.750 UTC			
10	574301	22144	6	132033690...	1970-01-01 00:00:02.100 UTC			

## 11-7. RFM 스코어

## Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT
    DATE(InvoiceDate) AS InvoiceDay, -- InvoiceDate에서 연월일만 추출
    *
FROM proven-serenity-439401-f2.modulabs_project.data_re;
```

쿼리 결과

[결과 저장](#) [데이터 탐색](#)

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice
1	2011-11-03	574301	85049E	12	2011-11-03 16:15:00 UTC	1.25
2	2011-11-03	574301	85049A	12	2011-11-03 16:15:00 UTC	1.25
3	2011-11-03	574301	22910	6	2011-11-03 16:15:00 UTC	2.95
4	2011-11-03	574301	23514	6	2011-11-03 16:15:00 UTC	2.08
5	2011-11-03	574301	22086	6	2011-11-03 16:15:00 UTC	2.95
6	2011-11-03	574301	20749	4	2011-11-03 16:15:00 UTC	7.95
7	2011-11-03	574301	23240	6	2011-11-03 16:15:00 UTC	4.15
8	2011-11-03	574301	20971	12	2011-11-03 16:15:00 UTC	1.25
9	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC	3.75
10	2011-11-03	574301	22144	6	2011-11-03 16:15:00 UTC	2.1
11	2011-11-03	574301	23511	6	2011-11-03 16:15:00 UTC	2.08
12	2011-11-03	574301	22077	12	2011-11-03 16:15:00 UTC	1.95

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
    MAX(DATE(InvoiceDate)) OVER() AS most_recent_date, -- 가장 최근 구매 일자를 윈도우 함수로 계산
    DATE(InvoiceDate) AS InvoiceDay, -- 연월일로 변환한 구매 일자
    *
FROM proven-serenity-439401-f2.modulabs_project.data_re;
```

쿼리 결과

[결과 저장](#) [데이터 탐색](#)

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-12-09	2011-10-06	569943	23177	1	2011-10-06 18:08:00 UTC
2	2011-12-09	2011-07-01	558700	21034	1	2011-07-01 12:33:00 UTC
3	2011-12-09	2011-02-28	545158	22656	1	2011-02-28 13:28:00 UTC
4	2011-12-09	2011-07-31	561873	22568	24	2011-07-31 11:48:00 UTC
5	2011-12-09	2011-09-12	566431	23375	10	2011-09-12 14:39:00 UTC
6	2011-12-09	2011-04-14	550125	22059	12	2011-04-14 12:42:00 UTC
7	2011-12-09	2011-05-25	554682	23253	1	2011-05-25 16:08:00 UTC
8	2011-12-09	2011-04-21	550823	22803	1	2011-04-21 09:36:00 UTC
9	2011-12-09	2010-12-09	537889	22837	8	2010-12-09 10:04:00 UTC
10	2011-12-09	2011-12-07	581125	16011	25	2011-12-07 12:41:00 UTC
11	2011-12-09	2011-01-21	541789	22760	22	2011-01-21 13:07:00 UTC
12	2011-12-09	2011-11-23	578305	23406	12	2011-11-23 15:44:00 UTC
13	2011-12-09	2011-08-01	561917	21056	1	2011-08-01 11:57:00 UTC
14	2011-12-09	2011-10-04	569470	51008	17	2011-10-04 12:25:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS most_recent_date -- 유저별 가장 최근 구매일을 most_recent_date로 저장
```

```
FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY CustomerID; -- 유저별로 그룹화
```

## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	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			
8	15360	2011-10-31			
9	15872	2011-11-25			
10	16128	2011-11-22			
11	16384	2011-09-11			
12	17152	2011-05-29			

- 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 -- 전체에서 가장 최근 일자와 유저별 구매일
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay -- 유저별 마지막 구매일
  FROM proven-serenity-439401-f2.modulabs_project.data_re
  GROUP BY CustomerID
);
```

## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▼	recency ▼			
1	13824	32			
2	12551	357			
3	14345	38			
4	17418	42			
5	13579	14			
6	15380	8			
7	16150	38			
8	16919	156			
9	14364	108			
10	15667	39			
11	17736	9			

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

```
CREATE OR REPLACE TABLE proven-serenity-439401-f2.modulabs_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency -- 가장 최근 일자와 유저별 구매일 차이 계산
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay -- 유저별 마지막 구매일
  FROM proven-serenity-439401-f2.modulabs_project.data_re
  GROUP BY CustomerID
);
```

## 쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p><b>i</b> 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.</p>			



user\_r



쿼리 ▾



공유

스키마

세부정보

미리보기

행	CustomerID	recency
1	12985	0
2	18102	0
3	14051	0
4	17754	0
5	14446	0
6	13426	0
7	16446	0
8	17001	0
9	15804	0
10	14422	0
11	14441	0
12	16954	0
13	13777	0
14	12518	0
15	12433	0

## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt  -- 고객별 고유한 InvoiceNo 수(거래 건수)
FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY CustomerID;
```

## 쿼리 결과

작업 정보	결과	차트	JSON
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	
12	17152	4	
13	17408	1	

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt  -- 고객별로 구매한 아이템의 총 수량 합계
FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY CustomerID;
```



## 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▼	item_cnt ▼			
1 //	12544 //	130 //			
2	13568	66			
3	13824	768			
4	14080	48			
5	14336	1759			
6	14592	407			
7	15104	633			
8	15360	223			
9	15872	187			
10	16128	988			
11	16384	260			
12	17152	477			
13	17408	3			

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

```
CREATE OR REPLACE TABLE `proven-serenity-439401-f2.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt -- 고객별 고유한 InvoiceNo 수(거래 건수)
  FROM `proven-serenity-439401-f2.modulabs_project.data_re`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt -- 고객별로 구매한 아이템의 총 수량 합계
  FROM `proven-serenity-439401-f2.modulabs_project.data_re`
  GROUP BY CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.reccency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN `proven-serenity-439401-f2.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

## 쿼리 결과

[작업 정보](#)[결과](#)[실행 세부정보](#)[실행 그래프](#)

이 문으로 이름이 user\_rf인 새 테이블이 생성되었습니다.



user\_rf



쿼리 ▾



공유



복사



스냅샷

[스키마](#)[세부정보](#)[미리보기](#)[테이블 탐색기](#)[미리보기](#)

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	12792	1	215	256
3	18010	1	60	256
4	15083	1	38	256
5	15520	1	314	1
6	14569	1	79	1
7	13298	1	96	1
8	13436	1	76	1
9	14476	1	110	257
10	13357	1	321	257
11	14204	1	72	2
12	15195	1	1404	2
13	15471	1	256	2
14	12442	1	181	3
15	12650	1	250	3

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total -- 고객별 총 지출액, 소수점 첫째 자리에서 반올림
FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY CustomerID;
```

## 쿼리 결과

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			
10	16128	1880.2			

### 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 proven-serenity-439401-f2.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, 1) AS user_average -- 평균 거래 금액 계산
FROM proven-serenity-439401-f2.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 1) AS user_total -- 고객별 총 지출액 계산, 소수점 첫째 자리 반올림
  FROM proven-serenity-439401-f2.modulabs_project.data_re
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

## 쿼리 결과

결과 저장

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.			

user_rfm		쿼리	공유	복사	스냅샷	삭제	내보내기
스키마	세부정보	미리보기	테이블 탐색기	미리보기	통계	계보	데이터 프로필
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	
1	12713	1	505	0	794.6	794.6	
2	18010	1	60	256	174.8	174.8	
3	12792	1	215	256	344.5	344.5	
4	15083	1	38	256	88.2	88.2	
5	13436	1	76	1	196.9	196.9	
6	13298	1	96	1	360.0	360.0	
7	15520	1	314	1	343.5	343.5	
8	14569	1	79	1	227.4	227.4	
9	13357	1	321	257	609.4	609.4	
10	14476	1	110	257	193.0	193.0	

## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```
SELECT *
FROM proven-serenity-439401-f2.modulabs_project.user_rfm
LIMIT 100;
```

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	794.6	794.6
2	18010	1	60	256	174.8	174.8
3	12792	1	215	256	344.5	344.5
4	15083	1	38	256	88.2	88.2
5	13436	1	76	1	196.9	196.9
6	13298	1	96	1	360.0	360.0
7	15520	1	314	1	343.5	343.5
8	14569	1	79	1	227.4	227.4
9	13357	1	321	257	609.4	609.4
10	14476	1	110	257	193.0	193.0

## 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 proven-serenity-439401-f2.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products -- 고유한 제품 수 계산
```

```

FROM proven-serenity-439401-f2.modulabs_project.data_re
GROUP BY CustomerID
)
SELECT
  ur.*,
  up.unique_products -- unique_products 값을 추가
FROM proven-serenity-439401-f2.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

**i** 이 문으로 이름이 user\_data인 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 proven-serenity-439401-f2.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
      proven-serenity-439401-f2.modulabs_project.data_re
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM proven-serenity-439401-f2.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

### 쿼리 결과

작업 정보   **결과**   실행 세부정보   실행 그래프

**i** 이 문으로 이름이 user\_data인 테이블이 교체되었습니다.

user_data									
<div> <span>쿼리</span> <span>공유</span> <span>복사</span> <span>스냅샷</span> <span>삭제</span> <span>내보내기</span> <span>새로고침</span> </div>									
스키마	세부정보	미리보기	데이터 탐색기	미리보기	통계	계보	데이터 프로파일	데이터 품질	
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	
1	14432	6	2013	9	2248.5	374.8	256	0.2	
2	12428	11	3477	25	6366.0	578.7	256	0.87	
3	13268	14	3525	17	3105.7	221.8	256	0.56	
4	14911	243	76906	1	132893.2	546.9	1792	0.05	
5	15510	1	2	330	250.0	250.0	1	0.0	
6	15562	1	39	351	134.6	134.6	1	0.0	
7	12603	1	56	21	613.2	613.2	1	0.0	
8	16765	1	4	294	34.0	34.0	1	0.0	
9	15070	1	36	372	106.2	106.2	1	0.0	
10	18184	1	60	15	49.8	49.8	1	0.0	

### 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 proven-serenity-439401-f2.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions, -- 총 거래 횟수
    SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency -- 취소된 거래 횟수
  FROM proven-serenity-439401-f2.modulabs_project.data_re
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate -- 취소 비율 계산 (소수점 두번째 자리)
FROM proven-serenity-439401-f2.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

user_data									
<div> <span>쿼리</span> <span>공유</span> <span>복사</span> <span>스냅샷</span> <span>삭제</span> <span>내보내기</span> <span>새로고침</span> </div>									
스키마	세부정보	미리보기	데이터 탐색기	미리보기	통계	계보	데이터 프로파일	데이터 품질	
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_t
1	14432	6	2013	9	2248.5	374.8	256	0.2	
2	12428	11	3477	25	6366.0	578.7	256	0.87	
3	13268	14	3525	17	3105.7	221.8	256	0.56	
4	14911	243	76906	1	132893.2	546.9	1792	0.05	
5	16990	1	100	218	179.0	179.0	1	0.0	
6	15657	1	24	22	30.0	30.0	1	0.0	
7	16995	1	-1	372	-1.3	-1.3	1	0.0	
8	18113	1	72	368	76.3	76.3	1	0.0	
9	18184	1	60	15	49.8	49.8	1	0.0	

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

```
SELECT *
FROM proven-serenity-439401-f2.modulabs_project.user_data
LIMIT 100;
```

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	
1	14432	6	2013	9	2248.5	374.8	25	
2	12428	11	3477	25	6366.0	578.7	25	
3	13268	14	3525	17	3105.7	221.8	25	
4	14911	243	76906	1	132893.2	546.9	179	
5	16990	1	100	218	179.0	179.0		
6	15657	1	24	22	30.0	30.0		
7	16995	1	-1	372	-1.3	-1.3		
8	18113	1	72	368	76.3	76.3		
9	18184	1	60	15	49.8	49.8		

## 회고

이번 프로젝트를 통해 고객 세그먼테이션을 위한 **RFM 분석**과 추가적인 **고객 구매 패턴** 분석을 진행했습니다. RFM 분석 외에도 **구매 제품의 다양성**, **평균 구매 주기**, **취소 패턴** 등의 추가 특성을 추출함으로써 고객을 더 깊이 이해할 수 있었습니다. 각 특성을 바탕으로 **클러스터링 알고리즘**을 적용해 고객을 그룹화할 수 있는 기반을 마련했고, SQL을 활용한 데이터 처리와 분석이 실제로 어떻게 마케팅 전략에 적용될 수 있는지 실습할 수 있었습니다.