

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM `atomic-snow-425501-c4.modulabs_project.data`
LIMIT 10;
```

[결과 이미지를 넣어주세요]

```
1
2 SELECT *
3 FROM `atomic-snow-425501-c4.modulabs_project.data`
4 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	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 ttl_rows
FROM `atomic-snow-425501-c4.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

```
6 -- 전체 행 갯수 확인
7 SELECT COUNT(*) AS ttl_rows
8 FROM `atomic-snow-425501-c4.modulabs_project.data`
```

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보

행	ttl_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_UnitPirce,
  COUNT(CustomerID) AS COUNT_CustomerID,
  COUNT(Country) AS COUNT_Country,
FROM `atomic-snow-425501-c4.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

```

10 -- 컬럼별 데이터 포인트 수 확인
11 |
12 SELECT
13   COUNT(InvoiceNo) AS COUNT_InvoiceNo,
14   COUNT(StockCode) AS COUNT_StockCode,
15   COUNT(Description) AS COUNT_Description,
16   COUNT(Quantity) AS COUNT_Quantity,
17   COUNT(InvoiceDate) AS COUNT_InvoiceDate,
18   COUNT(UnitPrice) AS COUNT_UnitPirce,
19   COUNT(CustomerID) AS COUNT_CustomerID,
20   COUNT(Country) AS COUNT_Country,
21 FROM `atomic-snow-425501-c4.modulabs_project.data`

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프				
행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPirce	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

```

-- 컬럼별 결측치 비율 확인
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM atomic-snow-425501-c4.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 atomic-snow-425501-c4.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_percentag
FROM atomic-snow-425501-c4.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 atomic-snow-425501-c4.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 atomic-snow-425501-c4.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 atomic-snow-425501-c4.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 atomic-snow-425501-c4.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 atomic-snow-425501-c4.modulabs_project.data

```

[결과 이미지를 넣어주세요]

쿼리 결과

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

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM atomic-snow-425501-c4.modulabs_project.data
WHERE StockCode = '85123A'

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트
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 atomic-snow-425501-c4.modulabs_project.data
WHERE Description IS NULL
OR CustomerID IS NULL
```

[결과 이미지를 넣어주세요]

143	-- 결측치 제거
144	DELETE FROM atomic-snow-425501-c4.modulabs_project.data
145	WHERE Description IS NULL
146	OR CustomerID IS NULL

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 data의 행 135,080개가 삭제되었습니다.</p>			

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(*)
FROM (
  SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  FROM atomic-snow-425501-c4.modulabs_project.data
  GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
)
```

[결과 이미지를 넣어주세요]

쿼리 결과	
작업 정보	결과
행	f0_ ▼
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 modulabs_project.distinct_data AS
SELECT DISTINCT *
FROM atomic-snow-425501-c4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

```
159 -- 중복행 제거
160 CREATE OR REPLACE TABLE modulabs_project.data AS
161 SELECT DISTINCT *
162 FROM atomic-snow-425501-c4.modulabs_project.data
```

쿼리 결과

결과 저장

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 atomic-snow-425501-c4.modulabs_project.distinct_data
```

[결과 이미지를 넣어주세요]

```
164
165 -- 고유한 invoice 개수 출력
166 SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
167 FROM atomic-snow-425501-c4.modulabs_project.data
168
169
```

쿼리 결과

작업 정보

결과

차트

JSON

실행 세부정보

실행

행	unique_invoice_coun
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo,
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
```

LIMIT 100

[결과 이미지를 넣어주세요]

```
169 -- 고유한 Invoice 100개 리밋 출력
170 SELECT DISTINCT InvoiceNo,
171 FROM atomic-snow-425501-c4.modulabs_project.data
172 LIMIT 100
```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	574301				
2	C575531				
3	557305				
4	543008				
5	549735				
6	554032				
7	561387				

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

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

```
SELECT *
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

176 SELECT *

177 FROM atomic-snow-425501-c4.modulabs_project.data

178 WHERE InvoiceNo LIKE 'C%'

179 LIMIT 100;

180

181

접근성 옵션을 보려면 Alt+F1 키를 누르세요

쿼리 결과

결과 저장

데이터 탐색

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	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
3	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
4	C554983	475908	PINK HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-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	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom

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(*)*100, 1)
FROM atomic-snow-425501-c4.modulabs_project.distinct_data;
```

[결과 이미지를 넣어주세요]

```

237 -- 구매건 상태가 cancelled인 데이터 비율
238 SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*)*100, 1)
239 FROM atomic-snow-425501-c4.modulabs_project.data;

```

접근성 옵션을 보려면 Alt+F1 키를 누.

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	f0_
1	2.2

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```

SELECT COUNT(DISTINCT StockCode)
FROM atomic-snow-425501-c4.modulabs_project.distinct_data;

```

[결과 이미지를 넣어주세요]

```

243 -- 고유한 StockCode 개수 출력
244 SELECT COUNT(DISTINCT StockCode)
245 FROM atomic-snow-425501-c4.modulabs_project.data;
246

```

쿼리 결과

결과 저장

작업 정보 결과 차트 JSON 실행 세부정보

행	f0_
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 `atomic-snow-425501-c4.modulabs_project.distinct_data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;

```

[결과 이미지를 넣어주세요]

```

250 SELECT
251     StockCode,
252     COUNT(*) AS sell_cnt
253 FROM `atomic-snow-425501-c4.modulabs_project.data`
254 GROUP BY StockCode
255 ORDER BY sell_cnt DESC
256 LIMIT 10;

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	StockCode	sell_cnt		
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

```

-- StockCode의 문자열 내 숫자의 길이 구하기
WITH UniqueStockCodes AS (
    SELECT DISTINCT StockCode
    FROM `atomic-snow-425501-c4.modulabs_project.data`
)
SELECT
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
    COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;

-- 숫자가 0~1개인 값에는 어떤 코드들이 들어가 있는지 확인
SELECT DISTINCT StockCode, number_count
FROM (
    SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `atomic-snow-425501-c4.modulabs_project.distinct_data`
)
WHERE number_count IN (0, 1);

```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보	결과	차트
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)

```
SELECT
  ROUND
    (SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS percentage
FROM (
  SELECT LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `atomic-snow-425501-c4.modulabs_project.distinct_data`
) AS subquery;
```

[결과 이미지를 넣어주세요]

쿼리 결과	
작업 정보 결과	
행	percentage
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `atomic-snow-425501-c4.modulabs_project.distinct_data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM `atomic-snow-425501-c4.modulabs_project.data`
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

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 `atomic-snow-425501-c4.modulabs_project.distinct_data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

[결과 이미지를 넣어주세요]

```

320
326 -- -- DESCRIPTION 출현 빈도 계산 & 상위 30개 출력
327 SELECT Description, COUNT(*) AS description_cnt
328 FROM atomic-snow-425501-c4.modulabs_project.data
329 GROUP BY Description
330 ORDER BY description_cnt DESC
331 LIMIT 30

```

접근성 옵션

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			

페이지당 결과 수: 50 1 - 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```

DELETE
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');

```

[결과 이미지를 넣어주세요]

```

343
344 DELETE
345 FROM atomic-snow-425501-c4.modulabs_project.data
346 WHERE
347 Description IN ('Next Day Carriage', 'High Resolution Image');
348
349

```

접근성 옵션

쿼리 결과 결과 저장

작업 정보	결과	실행 세부정보	실행 그래프
-------	----	---------	--------

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```

CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.distinct_data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM atomic-snow-425501-c4.modulabs_project.distinct_data;

```

[결과 이미지를 넣어주세요]

```

352 -- 대소문자 혼합 데이터를 대문자로 표준화
353 CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.data AS
354 SELECT
355     **EXCEPT(Description),
356     UPPER(Description) AS Description
357 FROM atomic-snow-425501-c4.modulabs_project.data
358
359

```

접근성 옵션을 보

쿼리 결과

결과 저장

데

작업 정보

결과

실행 세부정보

실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최소값, 최대값, 평균을 구하기

```

SELECT
    MIN(UnitPrice) AS min_price,
    MAX(UnitPrice) AS max_price,
    AVG(UnitPrice) AS avg_price
FROM atomic-snow-425501-c4.modulabs_project.distinct_data;

```

[결과 이미지를 넣어주세요]

```

362
363 -- 단위가격 (UnitPrice)
364 SELECT
365     MIN(UnitPrice) AS min_price,
366     MAX(UnitPrice) AS max_price,
367     AVG(UnitPrice) AS avg_price
368 FROM atomic-snow-425501-c4.modulabs_project.data

```

쿼리 결과

결과 저장

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 최소값, 최대값, 평균 구하기

```

SELECT
    COUNT(*) AS cnt_quantity,
    MIN(Quantity) AS min_quantity,
    MAX(Quantity) AS max_quantity,
    AVG(Quantity) AS avg_quantity
FROM
    atomic-snow-425501-c4.modulabs_project.distinct_data
WHERE
    UnitPrice = 0;

```

[결과 이미지를 넣어주세요]

```

-- UnitPrice = 0인 데이터의 개수과 UnitPrice(Quantity)의 최솟값, 최댓값, 평균
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM
  atomic-snow-425501-c4.modulabs_project.data
WHERE
  UnitPrice = 0;

```

결과

cnt_quantity	min_quantity	max_quantity	avg_quantity
33	1	12540	420.5151515151...

- UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기

```

CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.distinct_data AS
SELECT *
FROM atomic-snow-425501-c4.modulabs_project.data
WHERE UnitPrice != 0;

```

[결과 이미지를 넣어주세요]

```

388 CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.data AS
389 SELECT *
390 FROM atomic-snow-425501-c4.modulabs_project.data
391 WHERE UnitPrice != 0;
392

```

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 atomic-snow-425501-c4.modulabs_project.distinct_data;

```

[결과 이미지를 넣어주세요]

```

396
397 SELECT DATE(InvoiceDate) AS InvoiceDay, *
398 FROM atomic-snow-425501-c4.modulabs_project.data

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프				
번호	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
1	2010-12-21	539722	22423	10	2010-12-21 13:45:00 UTC	0.0	14911	EIRE	
2	2011-08-11	562973	23157	240	2011-08-11 11:42:00 UTC	0.0	14911	EIRE	
3	2011-07-28	561669	22960	11	2011-07-28 17:09:00 UTC	0.0	12507	Spain	
4	2011-11-18	577314	23407	2	2011-11-18 13:23:00 UTC	0.0	12444	Norway	
5	2010-12-05	537197	22841	1	2010-12-05 14:02:00 UTC	0.0	12647	Germany	
6	2011-11-04	574469	22385	12	2011-11-04 11:55:00 UTC	0.0	12431	Australia	
7	2011-05-20	554037	22619	80	2011-05-20 14:13:00 UTC	0.0	12415	Australia	

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```

SELECT
    MAX(DATE(InvoiceDate)) AS most_recent_date,
FROM atomic-snow-425501-c4.modulabs_project.distinct_data;

```

[결과 이미지를 넣어주세요]

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```

SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
GROUP BY CustomerID;

```

[결과 이미지를 넣어주세요]

- 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 atomic-snow-425501-c4.modulabs_project.distinct_data
    GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

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

```

CREATE OR REPLACE TABLE atomic-snow-425501-c4.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 atomic-snow-425501-c4.modulabs_project.distinct_data
    GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
GROUP BY CustomerID;

```

[결과 이미지를 넣어주세요]

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```

SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
GROUP BY CustomerID;

```

[결과 이미지를 넣어주세요]

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

```

CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT InvoiceNo) AS purchase_cnt
    FROM atomic-snow-425501-c4.modulabs_project.distinct_data
    GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    SELECT
        CustomerID,
        SUM(Quantity) AS item_cnt
    FROM atomic-snow-425501-c4.modulabs_project.distinct_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 atomic-snow-425501-c4.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM atomic-snow-425501-c4.modulabs_project.distinct_data
GROUP BY 1;
```

[결과 이미지를 넣어주세요]

- 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 atomic-snow-425501-c4.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 atomic-snow-425501-c4.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM atomic-snow-425501-c4.modulabs_project.distinct_data
  GROUP BY 1
) ut
  ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM atomic-snow-425501-c4.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

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 atomic-snow-425501-c4.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM atomic-snow-425501-c4.modulabs_project.distinct_data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM atomic-snow-425501-c4.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE atomic-snow-425501-c4.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
      atomic-snow-425501-c4.modulabs_project.distinct_data
    WHERE CustomerID IS NOT NULL
```



```

    )
    GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM atomic-snow-425501-c4.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

[결과 이미지를 넣어주세요]

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)

```

CREATE OR REPLACE TABLE atomic-snow-425501-c4.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
  FROM atomic-snow-425501-c4.modulabs_project.distinct_data
  GROUP BY 1
)

SELECT u.*, t.* EXCEPT(CustomerID),
       ROUND((t.cancel_frequency / t.total_transactions * 100),2) AS cancel_rate
FROM `atomic-snow-425501-c4.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

[결과 이미지를 넣어주세요]

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

```

SELECT *
FROM atomic-snow-425501-c4.modulabs_project.user_data;

```

[결과 이미지를 넣어주세요]