

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM `spry-cortex-466606-f7.modulabs_project.data`  
  
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	1785
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	1785
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	1785
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	1785
5	536365	84406B	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	1785

페이지당 결과 수: 50

1 - 10 (전체 10행)

<

>

>

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT *  
FROM `spry-cortex-466606-f7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	1785
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	1785
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	1785
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	1785
5	536365	84406B	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	1785

페이지당 결과 수: 50

1 ~ 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_  
COUNT(InvoiceNo) AS InvoiceNo_count,  
COUNT(StockCode) AS StockCode_count,  
COUNT(Description) AS Description_count,  
COUNT(Quantity) AS Quantity_count,  
COUNT(InvoiceDate) AS InvoiceDate_count,  
COUNT(UnitPrice) AS UnitPrice_count,  
COUNT(CustomerID) AS CustomerID_count,  
COUNT(Country) AS Country_count  
FROM `spry-cortex-466606-f7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

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 column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `project_name.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 `project_name.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 `project_name.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 `project_name.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 `project_name.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 `project_name.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 `project_name.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 `project_name.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

페이지당 결과 수: 50 1 - 8 (전체 8행)

결측치 처리 전략

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

```
SELECT Description
FROM `spry-cortex-466606-f7.modulabs_project.data`
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	Description				
1	WHITE HANGING HEART T-LIG...				
2	WHITE HANGING HEART T-LIG...				
3	WHITE HANGING HEART T-LIG...				
4	WHITE HANGING HEART T-LIG...				
5	WHITE HANGING HEART T-LIG...				
6	WHITE HANGING HEART T-LIG...				
7	WHITE HANGING HEART T-LIG...				
8	WHITE HANGING HEART T-LIG...				

페이지당 결과 수: 50 1 - 8 (전체 2313행)

결측치 처리

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

```
DELETE FROM `spry-cortex-466606-f7.modulabs_project.data`
WHERE CustomerID IS NULL OR Description IS NULL ;
```

[결과를 실수로 2번 눌러버리는 바람에 이미 삭제되어 개수가 0...ㅠㅠ]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 data의 행 0개가 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(*) AS D_CNT
FROM (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*)
  FROM `spry-cortex-466606-f7.modulabs_project.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  HAVING COUNT(*) > 1
);
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보	결과	차트
JSON	실행 세부정보	실행 그래프
행	D_CNT	
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 `spry-cortex-466606-f7.modulabs_project.data` AS
SELECT DISTINCT *
FROM `spry-cortex-466606-f7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

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 InvoiceNo의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM `spry-cortex-466606-f7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과		
작업 정보 결과 차트		
행	f0_	
1	22190	

- 고유한 InvoiceNo를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM `spry-cortex-466606-f7.modulabs_project.data`
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과			결과 저장	
작업 정보 결과 차트 JSON 실행 세부정보				
행	InvoiceNo			
1	541431			
2	C541433			
3	537626			
4	542237			
5	549222			
6	556201			
-	-----			

페이지당 결과 수: 50 1 - 50 (전체 100행)

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

```
SELECT *
FROM `spry-cortex-466606-f7.modulabs_project.data`
WHERE InvoiceNo Like 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541439	23166	MEDIUM CERAMIC TOP STORA...	-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	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
7	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway

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)*100/ COUNT(*), 1)
FROM `spry-cortex-466606-f7.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트

행	f0_
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM `spry-cortex-466606-f7.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차

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 `spry-cortex-466606-f7.modulabs_project.data`
GROUP BY StockCode
```

```
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과			
결과 저장 ▼			
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	
-	-----	-----	
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 spry-cortex-466606-f7.modulabs_project.data
)
WHERE number_count IN(0,1);
```

[결과 이미지를 넣어주세요]

쿼리 결과			
결과 저장 ▼			
작업 정보 결과 차트 JSON 실행 세부정보			
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	
페이지당 결과 수: 50 ▼ 1 - 8 (전체 8행)			

- **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 IN (0, 1) THEN 1 ELSE 0 END)*100/ COUNT(*), 2) AS Percent
FROM(
SELECT StockCode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM `spry-cortex-466606-f7.modulabs_project.data`) AS subQ;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차
행	Percent ▼	
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM spry-cortex-466606-f7.modulabs_project.data
WHERE StockCode IN (
SELECT StockCode
FROM (
SELECT Stockcode,
LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
FROM spry-cortex-466606-f7.modulabs_project.data)
WHERE number_count IN (0, 1)
);
```

[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 spry-cortex-466606-f7.modulabs_project.data_copy1
GROUP BY Description
ORDER BY description_cnt
Limit 30;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	Description	description_cnt			
1	INCENSE BAZAAR PEACH	1			
2	WOOLLY HAT SOCK GLOVE ADVENT STRING	1			
3	CREAM SWEETHEART TRAYS	1			
4	SWEETHEART KEY CABINET	1			
5	NEW BAROQUE B'FLY NECKLACE GREEN	1			
6	AMBER GLASS/SHELL/PEARL NECKLACE	1			
7	5 STRAND GLASS NECKLACE AMETHYST	1			
8	HOT WATER BOTTLE BABUSHKA LARGE	1			
9	FLAMINGO LIGHTS	1			

페이지당 결과 수: 200 1 - 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
WHERE REGEXP_CONTAINS(Description, 'Next Day Carriage|High Resolution Image');
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 data_copy1의 행 83개가 삭제되었습니다.			

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `spry-cortex-466606-f7.modulabs_project.data_copy1` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data_copy1인 테이블이 교체되었습니다.			

UnitPrice 살펴보기

- UnitPrice 의 최소값, 최대값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM spry-cortex-466606-f7.modulabs_project.data_copy1
```

[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 최소값, 최대값, 평균 구하기

```
SELECT
COUNT(Quantity) AS total_quantity,
SUM(CASE WHEN UnitPrice = 0 THEN 1 ELSE 0 END) AS zero_cnt,
MIN(Quantity) AS min_quantity,
MAX(Quantity) AS max_quantity,
AVG(Quantity) AS avg_quantity
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	total_quantity	zero_cnt	min_quantity	max_quantity	avg_quantity
1	399606	33	-80995	80995	12.23171824246...

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE spry-cortex-466606-f7.modulabs_project.data_copy1 AS
SELECT *
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
WHERE Unitprice != 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data_copy1인 테이블이 교체되었습니다.			

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay, *
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프				
행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM
2	2011-01-18	C541433	23166	74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM
3	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE D
4	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	RED TO
5	2010-12-07	537626	84640	4	2010-12-07 14:57:00 UTC	1.19	12347	Iceland	BLUE D

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

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  (SELECT MAX(InvoiceDate) FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

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 12:50:00 UTC	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	2011-12-09 12:50:00 UTC	2011-01-18	C541433	23166	74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	2011-12-09 12:50:00 UTC	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
4	2011-12-09 12:50:00 UTC	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland
5	2011-12-09 12:50:00 UTC	2010-12-07	537626	404649	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland

페이지당 결과 수: 50

1 - 50 (전체 399573행)

<

>

>>

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS most_recent_purchase_date
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	most_recent_purchase_date			
1	12346	2011-01-18			
2	12347	2011-12-07			
3	12348	2011-09-25			
4	12349	2011-11-21			

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 spry-cortex-466606-f7.modulabs_project.data_copy1
  GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
번호	CustomerID	recency			
1	12560	5			
2	12571	39			
3	12821	214			
4	12883	24			

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 spry-cortex-466606-f7.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 spry-cortex-466606-f7.modulabs_project.data_copy1
  GROUP BY CustomerID
);

```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

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

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt

```

```
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt			
1	12346	2			
2	12347	182			
3	12348	27			
4	12349	72			

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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			

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 spry-cortex-466606-f7.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS purchase_cnt
  FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
```

```

SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
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 spry-cortex-466606-f7.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;

```

[결과 이미지를 넣어주세요]

쿼리 결과



작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 total_spending
FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
GROUP BY CustomerID;

```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 ▾



작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▾	total_spending ▾			
1	12346	0.0			
2	12347	4310.0			
3	12348	1437.2			
4	12349	1457.6			

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 `spry-cortex-466606-f7.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
FROM `spry-cortex-466606-f7.modulabs_project.user_rf` AS rf
LEFT JOIN (

  SELECT
    CustomerID,
    SUM(Quantity * UnitPrice) AS user_total
  FROM `spry-cortex-466606-f7.modulabs_project.data_copy1`
  GROUP BY CustomerID
) AS 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 `spry-cortex-466606-f7.modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]

user_rfm 쿼리 다음에서 열기

스키마 세부정보 **미리보기** 테이블 탐색기 프리뷰 통계 계보 데이터

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12433	420	11071	0	13375.86999...	31.8
2	14441	59	430	0	1545.140000...	26.2
3	13069	469	5454	0	3713.139999...	7.9
4	14446	276	856	0	1005.550000...	3.6
5	12518	119	1306	0	1840.889999...	15.5
6	17428	343	9435	0	17078.45000...	49.8
7	14397	95	1852	0	2556.680000...	26.9
8	17364	409	2671	0	4437.229999...	10.8
9	13113	278	2596	0	10523.65000...	37.9
10	12423	118	1312	0	1624.110000...	13.8
11	15804	273	2513	0	3848.549999...	14.1
12	16626	184	2670	0	4379.650000...	23.8
13	17001	169	2164	0	3989.570000...	23.6
14	17315	482	3805	0	6153.340000...	12.8

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 spry-cortex-466606-f7.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM spry-cortex-466606-f7.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM spry-cortex-466606-f7.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

user_data								
<div> <div>스키마 세부정보</div> <div> <div>마리보기</div> <div>타이틀 탐색기</div> <div>표리본</div> <div>통계</div> <div>개보</div> <div>데이터 프로입</div> <div>데이터 품질</div> </div> </div>								
명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	
1	13135	1	4300	196	3096.0	3096.0	1	
2	14705	1	100	198	179.0	179.0	1	
3	16990	1	100	218	179.0	179.0	1	
4	16738	1	3	297	3.75	3.8	1	
5	16061	1	-1	269	-29.95	-29.9	1	
6	13120	1	12	238	30.59999999...	30.6	1	
7	17752	1	192	359	80.64	80.6	1	
8	17763	1	12	263	15.0	15.0	1	
9	13017	1	48	7	204.0	204.0	1	
10	13366	1	144	50	56.16000000...	56.2	1	
11	16093	1	20	106	17.0	17.0	1	
12	15510	1	2	330	250.0	250.0	1	
13	16995	1	-1	372	-1.25	-1.3	1	
14	13841	1	100	252	85.0	85.0	1	
15	15940	1	4	311	35.8	35.8	1	
16	15195	1	1,604	5	3861.0	3861.0	1	

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

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 spry-cortex-466606-f7.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
      spry-cortex-466606-f7.modulabs_project.data
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM spry-cortex-466606-f7.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

[결과 이미지를 넣어주세요]

user_data								
<div> <div>스키마 세부정보</div> <div> <div>마리보기</div> <div>타이틀 탐색기</div> <div>표리본</div> <div>통계</div> <div>개보</div> <div>데이터 프로입</div> <div>데이터 품질</div> </div> </div>								
명	CustomerID	recency	purchase_cnt	item_cnt	user_total	user_average	avg_purchase_interva...	
1	17914	3	75	457	329.39999999...	4.4	0.0	
2	17936	5	72	257	380.12999999...	5.3	0.0	
3	15471	2	73	256	454.48	6.2	0.0	
4	18139	17	159	5557	8438.399999...	53.1	0.0	
5	16222	318	122	325	773.55000000...	6.3	0.0	
6	13336	77	51	559	795.12	15.6	0.0	
7	15000	47	77	218	490.51999999...	6.4	0.0	
8	16274	373	63	151	331.54999999...	5.3	0.0	
9	13259	61	81	126	283.17999999...	3.5	0.0	
10	14381	52	56	146	202.68000000...	3.6	0.0	
11	12743	134	131	315	540.12999999...	4.1	0.0	
12	16956	25	56	171	261.90000000...	4.7	0.0	
13	16056	45	66	345	650.75000000...	9.9	0.0	
14	16800	11	156	524	1162.65000000...	7.5	0.0	
15	17100	18	65	800	971.74	14.9	0.0	
16	17945	18	75	458	1,457.50000000...	50.5	0.0	

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

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 spry-cortex-466606-f7.modulabs_project.user_data AS
```

```
WITH TransactionInfo AS (  
  SELECT  
    CustomerID,  
    COUNT(*) AS total_transactions,  
    COUNT(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 END) AS cancel_frequency  
  FROM spry-cortex-466606-f7.modulabs_project.data  
  GROUP BY CustomerID  
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(SAFE_DIVIDE(cancel_frequency, total_transactions) * 100, 2) AS cancel_rate  
FROM `spry-cortex-466606-f7.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

i 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 spry-cortex-466606-f7.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

user_data

스키마

세부정보

비율정보

데이터 탐색기

통계

계보

데이터 프로파일

데이터 품질

user_total

user_average

avg_purchases

average_inter...

cancel_count

transaction...

cancel_rate...

average_inter...

ten_total

ten_cancel

total_transac...

cancel_freque...

cancel_rate

24

30.0

30.0

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

100

179.0

179.0

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

72

76.32000000...

76.3

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

144

79.2

79.2

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

2

25.5

25.5

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

4

440.0

440.0

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

10

20.8

20.8

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

4300

3096.0

3096.0

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

16

175.2

175.2

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

4

59.8

59.8

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

10

99.5

99.5

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

72

550.8000000...

550.8

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

1440

244.8

244.8

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

10

20.8

20.8

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

6

101.6999999...

101.7

0.0

0.0

0

1

0.0

0.0

1

0

1

0

0.0

페이지당 결과 수

50

1 ~ 50 (전체 4362개)

페이지당 결과 수: 50 1 ~ 50 (전체 4362행) | < >

회고

[회고 내용을 작성해주세요]

Keep : SQL에 대한 이해도가 낮아서 계속해서 검색해서 답을 찾고 답을 이해하는 것조차 버거운 상태

Problem : 마지막 평균구매주기, 구매취소 경향성을 구할 때 여러번 쿼리문을 반복할 때마다 오류가 발생

Try : 검색해서 찾은 오류의 해결책들을 계속 시도하곤 했지만 사실상 이해도, 해결도 어려웠음