

고객을 세그멘테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
select * from amiable-nova-447401-t6.modulabs_project.data limit 10;
```

[결과 이미지를 넣어주세요]

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

명	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	

페이지당 결과 수: 50 ▼

1 ~ 10 (전체 10행)

< > >>

페이지당 결과 수: 50 1 - 10 (전체 10행) < > > >

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
select count(*) from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	f0_
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
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 amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

5-4. 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

```
# [[YOUR QUERY]]
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.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 amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	column_name	missing_percentage
1	UnitPrice	0.0
2	Country	0.0
3	Quantity	0.0
4	InvoiceNo	0.0
5	InvoiceDate	0.0
6	Description	0.27
7	CustomerID	24.93
8	StockCode	0.0

```
//참고_ 위와 결과 같음
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
  SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM amiable-
  SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM amiable-
  SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM amia
  SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM amiable-nc
  SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM amia
  SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM amiable-
  SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM amiabl
  SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM amiable-nova
) AS column_data;
```

결측치 처리 전략

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

```
SELECT # [YOUR QUERY]
FROM project_name.modulabs_project.data
# [YOUR QUERY]; distinct(Description)
```

[결과 이미지를 넣어주세요]

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 project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]CustomerID is null or Description is null;
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *  
FROM project_name.modulabs_project.data  
# [[YOUR QUERY]]  
select * FROM amiable-nova-447401-t6.modulabs_project.data  
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country  
having count(*) > 1;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice
1	557305	22645	CERAMIC HEART FAIRY CAKE ...	4	2011-06-19 14:42:00 UTC	
2	569943	20972	PINK CREAM FELT CRAFT TRIN...	1	2011-10-06 18:08:00 UTC	
3	571241	22807	SET OF 6 T-LIGHTS TOADSTOO...	1	2011-10-14 14:58:00 UTC	
4	571241	72816	SET/3 CHRISTMAS DECOUPAG...	1	2011-10-14 14:58:00 UTC	
5	571241	22940	FELTCRAFT CHRISTMAS FAIRY	1	2011-10-14 14:58:00 UTC	
6	571241	22095	LADS ONLY TISSUE BOX	3	2011-10-14 14:58:00 UTC	
7	571241	22630	DOLLY GIRL LUNCH BOX	1	2011-10-14 14:58:00 UTC	
8	554917	22848	BREAD BIN DINER STYLE PINK	1	2011-05-27 12:29:00 UTC	
9	554917	22849	BREAD BIN DINER STYLE MINT	1	2011-05-27 12:29:00 UTC	

페이지당 결과 수: 50 1 ~ 50 (전체 4837명) |< < > >I

중복값 처리

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

```
# [[YOUR QUERY]];  
CREATE OR REPLACE TABLE amiable-nova-447401-t6.modulabs_project.data  
as select distinct * from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

행	f0_
1	401604

5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
# [[YOUR QUERY]]
select count(distinct InvoiceNo) from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	f0_
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
select distinct InvoiceNo from amiable-nova-447401-t6.modulabs_project.data limit 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868
9	574877

페이지당 결과 수: 50 1 - 50 (전체 100행)

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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]] InvoiceNo like 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPric
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	
3	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	
4	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	
5	C554983	47590B	PINK HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	
6	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	
7	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	
8	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	

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 project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보		결과
행	f0_	
1		2.2

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
# [[YOUR QUERY]]
select count(distinct StockCode) from amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보		결과	샤드
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 project_name.modulabs_project.data
# group by StockCode
ORDER BY sell_cnt DESC
# limit 10;
```

[결과 이미지를 넣어주세요]

- StockCode 의 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM project_name.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;
```

[결과 이미지를 넣어주세요]

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

- **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 amiable-nova-447401-t6.modulabs_project.data
)
WHERE # number_count <=1;
```

[결과 이미지를 넣어주세요]

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)

```
# [[YOUR QUERY]]
select ROUND(SUM(CASE WHEN number_count <=1 THEN 1 ELSE 0 END)/count(*)*100, 2)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM amiable-nova-447401-t6.modulabs_project.data
) ;
```

[결과 이미지를 넣어주세요]

행	f0_
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM amiable-nova-447401-t6.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 amiable-nova-447401-t6.modulabs_project.data) WHERE number_count <=1
  );
```


[결과 이미지를 넣어주세요]

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 project_name.modulabs_project.data
# group by Description limit 30;
```

[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	ASSORTED COLOUR BIRD ORN...	1405
2	SCANDINAVIAN REDS RIBBONS	343
3	PAPER CHAIN KIT 50'S CHRIST...	1013
4	PAPER CHAIN KIT VINTAGE CH...	718
5	EMBROIDERED RIBBON REEL S...	63
6	EMBROIDERED RIBBON REEL E...	87
7	TRADITIONAL KNITTING NANCY	523
8	ASSORTED COLOUR MINI CAS...	372
9	POSTAGE	1196
10	FELTCRAFT PRINCESS OLIVIA ...	254

페이지당 결과 수: 50 1 - 30 (전체 30행)

- 대소문자가 혼합된 Description이 있는지 확인하기

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	Description ▼				
1	BAG 125g SWIRLY MARBLES				
2	3 TRADITIONAL BISCUIT CUTTE...				
3	BAG 250g SWIRLY MARBLES				
4	ESSENTIAL BALM 3.5g TIN IN ...				
5	FOLK ART GREETING CARD,pa...				
6	BAG 500g SWIRLY MARBLES				
7	POLYESTER FILLER PAD 45x45...				
8	POLYESTER FILLER PAD 40x40...				
9	Next Day Carriage				

페이지당 결과 수: 50 ▼ 1 - 19 (전체 19행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM project_name.modulabs_project.data
WHERE
# WHERE Description = "Next Day Carriage"
OR description = "High Resolution Image";
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE amiable-nova-447401-t6.modulabs_project.data AS
SELECT
* EXCEPT (Description),
# upper(Description) AS Description
FROM amiable-nova-447401-t6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	Description ▼
1	ASSORTED COLOUR MINI CAS...
2	EMBROIDERED RIBBON REEL R...
3	6 RIBBONS RUSTIC CHARM
4	ASSORTED COLOUR BIRD ORN...
5	FELTCRAFT PRINCESS LOLA D...
6	EMBROIDERED RIBBON REEL E...
7	JAM MAKING SET WITH JARS
8	PAPER CHAIN KIT 50'S CHRIST...
9	TRADITIONAL CHRISTMAS RIB...

UnitPrice 살펴보기

- **UnitPrice** 의 최솟값, 최댓값, 평균을 구하기

```
SELECT # min(UnitPrice) AS min_price, max(UnitPrice) AS max_price, avg(UnitPrice) AS avg_price
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	min_price	max_price	avg_price		
1	0.0	649.5	2.904956757405...		

- 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, a
FROM project_name.modulabs_project.data
WHERE # UnitPrice= 0;
```

[결과 이미지를 넣어주세요]

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 project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]

```
6 select count(*) from amiable-nova-447401-t6.modulabs_project.data2 where UnitPrice=0;
```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	count				
1	0				

5-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT # Date(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice
1	2011-11-03	574301	22910	6	2011-11-03 16:15:00 UTC	2.9
2	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC	3.7
3	2011-11-03	574301	23512	6	2011-11-03 16:15:00 UTC	2.0
4	2011-11-03	574301	84879	8	2011-11-03 16:15:00 UTC	1.6
5	2011-11-03	574301	23511	6	2011-11-03 16:15:00 UTC	2.0
6	2011-11-03	574301	22734	6	2011-11-03 16:15:00 UTC	2.8
7	2011-11-03	574301	22750	4	2011-11-03 16:15:00 UTC	3.7
8	2011-11-03	574301	22621	12	2011-11-03 16:15:00 UTC	1.6
9	2011-11-03	574301	22077	12	2011-11-03 16:15:00 UTC	1.9
10	2011-11-03	574301	22960	6	2011-11-03 16:15:00 UTC	4.2

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  # MAX(Date(InvoiceDate)) AS most_recent_date,
  # Date(InvoiceDate) AS InvoiceDay,
  *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
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
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		

- 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(InvoiceDate) AS InvoiceDay
```

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

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON
행	CustomerID	recency	
1	15360	39	
2	13314	1	
3	13061	72	
4	15373	8	
5	13328	316	
6	12822	70	
7	17942	7	
8	12847	22	
9	16689	75	

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

```
CREATE OR REPLACE TABLE project_name.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 amiable-nova-447401-t6.modulabs_project.data2
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON
행	CustomerID	recency	
1	15344	0	
2	12680	0	
3	14051	0	
4	17389	0	
5	17001	0	
6	17754	0	
7	16446	0	
8	12985	0	

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  # count(InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
#group by CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON
행	CustomerID	purchase_cnt	
1	12544	19	
2	13568	43	
3	13824	46	
4	14080	4	
5	14336	90	
6	14592	150	

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  # count(Quantity) AS item_cnt
FROM project_name.modulabs_project.data
# group by CustomerID ;
```

[결과 이미지를 넣어주세요]

행	CustomerID	item_cnt	
10	16128	89	
11	16384	33	
12	17152	16	
13	17408	2	
14	17664	43	
15	17920	664	
16	18176	33	
17	12545	48	
18	13313	78	

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS

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

```

    CustomerID,
    count(InvoiceNo) AS purchase_cnt
FROM amiable-nova-447401-t6.modulabs_project.data2
group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    #
    SELECT
    CustomerID,
    count(Quantity) AS item_cnt
FROM amiable-nova-447401-t6.modulabs_project.data2
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 project_name.modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;

```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▼	purchase_cnt ▼	item_cnt ▼	recency ▼		
1	15694	78	78	0		
2	15910	266	266	0		
3	12985	78	78	0		
4	14051	214	214	0		
5	17364	409	409	0		
6	14441	59	59	0		
7	12423	118	118	0		
8	13777	217	217	0		
9	16626	184	184	0		
10	12662	222	222	0		

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
    CustomerID,
    # Round(sum(UnitPrice)) AS user_total
FROM project_name.modulabs_project.data
# group by CustomerID;

```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12544	54.0
2	13568	131.0
3	13824	127.0
4	14080	4.0
5	14336	145.0
6	14592	324.0
7	15104	365.0
8	15360	31.0

- 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 project_name.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  # ut.user_total / rf.purchase_cnt AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    SELECT
      CustomerID,
      Round(sum(UnitPrice)) AS user_total
    FROM amiable-nova-447401-t6.modulabs_project.data2
    GROUP BY CustomerID
  ) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 `user_rfm`인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

```
# select * from amiable-nova-447401-t6.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	15694	78	78	0	357.0	4.576923076923...
2	14422	222	222	0	624.0	2.810810810810...
3	12985	78	78	0	146.0	1.871794871794...
4	12423	118	118	0	245.0	2.076271186440...
5	12526	68	68	0	184.0	2.705882352941...
6	17581	451	451	0	1599.0	3.545454545454...
7	16705	284	284	0	1284.0	4.521126760563...
8	17364	409	409	0	1115.0	2.726161369193...

5-8. 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 project_name.modulabs_project.user_data AS
WITH unique_products AS (
    SELECT
        CustomerID,
        COUNT(DISTINCT StockCode) AS unique_products
    FROM project_name.modulabs_project.data
    GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 `user_data`인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12428	292	292	25	863.0	2.955479452054...	256
2	14432	377	377	9	553.0	1.466843501326...	256
3	13268	439	439	17	691.0	1.574031890660...	256
4	16428	1	1	81	3.0	3.0	1
5	15668	1	1	217	1.0	1.0	1
6	13366	1	1	50	0.0	0.0	1
7	16953	1	1	30	2.0	2.0	1
8	16737	1	1	53	1.0	1.0	1
9	13829	1	1	359	9.0	9.0	1
10	13135	1	1	196	1.0	1.0	1

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE project_name.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
        project_name.modulabs_project.data
    WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프		
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	12428	292	292	25	863.0	2.955479452054...	256	0.8
2	14432	377	377	9	553.0	1.466843501326...	256	0.
3	13268	439	439	17	691.0	1.574031890660...	256	0.5
4	16138	1	1	368	8.0	8.0	1	0.
5	16323	1	1	196	4.0	4.0	1	0.
6	16061	1	1	269	30.0	30.0	1	0.
7	17948	1	1	147	2.0	2.0	1	0.
8	15562	1	1	351	3.0	3.0	1	0.
9	13703	1	1	318	10.0	10.0	1	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 project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
    SELECT
        CustomerID,
        # count(InvoiceNo) AS total_transactions,
        # SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
    FROM project_name.modulabs_project.data
    #group by CustomerID

    SELECT u.*, t.* EXCEPT(CustomerID), # round(cancel_frequency/total_transactions,3) AS cancel_rate
    FROM `project_name.modulabs_project.user_data` AS u
    LEFT JOIN TransactionInfo AS t
    ON # t.CustomerID = u.CustomerID;
```

[결과 이미지를 넣어주세요]

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

```
# select * from amiable-nova-447401-t6.modulabs_project.user_data;;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interv
1	14424	1	1	17	7.0	7.0	1	
2	13391	1	1	203	15.0	15.0	1	
3	13270	1	1	366	3.0	3.0	1	
4	15668	1	1	217	1.0	1.0	1	
5	15389	1	1	172	1.0	1.0	1	
6	13366	1	1	50	0.0	0.0	1	
7	15195	1	1	2	3.0	3.0	1	
8	16737	1	1	53	1.0	1.0	1	
9	15313	1	1	110	2.0	2.0	1	

페이지당 결과 수: 50 1 - 50 (전체 4362행) |< > >|