

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *  
from genial-shuttle-439400-e6.modulabs_project.data  
limit 10
```

[결과 이미지를 넣어주세요]

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	536414	22139		56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134		1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145		1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509		1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A		1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044		1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom

페이지당 결과 수:

50

1 - 10 (전체 10행)

<

>

>>

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*)  
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	col. ▾				
1	541909				

페이지당 결과 수: 50 ▾

1 - 1 (전체 1행)

<

>

>|

데이터 수 세기

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

```
select count(invoiceno) COUNT_InvoiceNo,  
count(stockcode) COUNT_StockCode,  
count(description) COUNT_Description,  
count(quantity) COUNT_Quantity,  
count(invoicedate) COUNT_InvoiceDate,  
count(unitprice) COUNT_UnitPrice,  
count(customerid) COUNT_CustomerID,  
count(country) COUNT_Country  
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프			
행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
1	541909	541909	540455	541909	541909	541909	406829	541909

페이지당 결과 수: 50

1 - 1 (전체 1행)

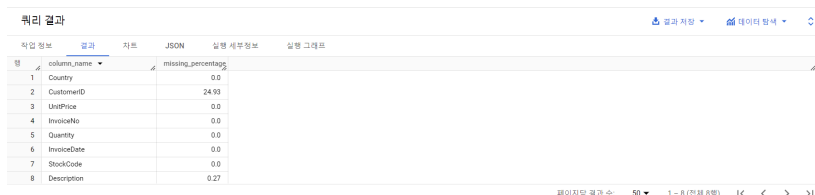
11-4. 데이터 전처리 방법(1): 결측치 제거

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_percent
FROM genial-shuttle-439400-e6.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_percent
FROM genial-shuttle-439400-e6.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_perce
FROM genial-shuttle-439400-e6.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_percenta
FROM genial-shuttle-439400-e6.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_perce
FROM genial-shuttle-439400-e6.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_percent
FROM genial-shuttle-439400-e6.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_perce
FROM genial-shuttle-439400-e6.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_percentag
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]



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

결측치 처리 전략

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

```
select distinct description
from genial-shuttle-439400-e6.modulabs_project.data
where StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 자르 JSON 실행 세부정보 실행 그래프

행	description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART FLAG...
4	WHITE HANGING HEART FLAG...

페이지당 결과 수: 50 1 - 4 (전체 4행) [K] < > >|

결측치 처리

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

```
DELETE FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE Description is null or customerid is null;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 data의 행 135,080개가 삭제되었습니다.	
----------------------------------	--

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
select *
from genial-shuttle-439400-e6.modulabs_project.data
group by 1, 2, 3, 4, 5, 6, 7, 8
having count(*) > 1
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 자르 JSON 실행 세부정보 실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	557905	22645	CERAMIC HEART FAIRY CAKE ...	4	2011-06-19 14:42:00 UTC	1.45	13568	United Kingdom
2	569943	20972	PINK CREAM FELT CRAFT TRI...	1	2011-10-06 18:08:00 UTC	1.25	14592	United Kingdom
3	571241	22807	SET OF 6 TLIGHTS TOASTTOO...	1	2011-10-14 14:58:00 UTC	2.95	14592	United Kingdom
4	571241	22095	LADS ONLY TISSUE BOX	3	2011-10-14 14:58:00 UTC	0.39	14592	United Kingdom
5	571241	72816	SET/3 CHRISTMAS DECOPAG...	1	2011-10-14 14:58:00 UTC	0.95	14592	United Kingdom
6	571241	22630	DOLLY GIRL LUNCH BOX	1	2011-10-14 14:58:00 UTC	1.95	14592	United Kingdom
7	571241	22940	FELTCRAFT CHRISTMAS FAIRY	1	2011-10-14 14:58:00 UTC	4.25	14592	United Kingdom

페이지당 결과 수: 50 1 - 50 (전체 4837행) [K] < > >|

count가 1보다 큰 데이터 개수 : 전체 행 개수

중복값 처리

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

```
create or replace table genial-shuttle-439400-e6.modulabs_project.data as
select distinct *
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

쿼리 결과									
작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프									
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
1	574301	22751	FELTCRAFT PRINCESS OLIVIA ..	4	2011-11-03 16:15:00 UTC	3.75	12544	Spain	
2	574301	22910	PAPER CHAIN KIT VINTAGE CH..	6	2011-11-03 16:15:00 UTC	2.95	12544	Spain	
3	574301	22144	CHRISTMAS CRAFT LITTLE PRL..	6	2011-11-03 16:15:00 UTC	2.1	12544	Spain	
4	574301	22754	SET OF 6 RIBBONS VINTAGE C..	6	2011-11-03 16:15:00 UTC	2.89	12544	Spain	
5	574301	23512	EMBROIDERED RIBBON REEL RL..	6	2011-11-03 16:15:00 UTC	2.08	12544	Spain	
6	574301	85049E	SCANDINAVIAN REDS RIBBONS	12	2011-11-03 16:15:00 UTC	1.25	12544	Spain	
7	574301	85049A	TRADITIONAL CHRISTMAS RIBL..	12	2011-11-03 16:15:00 UTC	1.25	12544	Spain	

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
select count(distinct invoiceno)
from genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과									
작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프									
행	f0								
1	22190								

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
select distinct invoiceno from genial-shuttle-439400-e6.modulabs_project.data
limit 100
```

[결과 이미지를 넣어주세요]

쿼리 결과									
작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프									
행	invoiceno								
1	574301								
2	C975531								
3	597595								
4	540309								
5	549735								
6	554032								
7	561387								
8	574868								

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

```
SELECT *
FROM genial-shuttle-439400-e6.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	C875331	22940	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 CASE 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	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(invoiceno) * 100, 1)
FROM genial-shuttle-439400-e6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

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 genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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 genial-shuttle-439400-e6.modulabs_project.data
group by stockcode
ORDER BY sell_cnt DESC
limit 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	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	P05T	1196			

페이지당 결과 수: 50 1 ~ 10 (현재 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 genial-shuttle-439400-e6.modulabs_project.data
)
WHERE number_count < 5;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	StockCode	number_count			
1	P05T	0			
2	M	0			
3	PADIS	0			
4	D	0			
5	BANK CHARGES	0			
6	DOT	0			
7	CRUK	0			
8	C2	1			

페이지당 결과 수: 50 1 ~ 8 (현재 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 stockcode in (SELECT DISTINCT StockCode
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM genial-shuttle-439400-e6.modulabs_project.data
)
WHERE number_count < 5)
THEN 1 ELSE 0 END)/count(stockcode) * 100, 2)
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.data
)
WHERE number_count < 5
)
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

i 이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM genial-shuttle-439400-e6.modulabs_project.data
group by description
order by description_cnt desc
limit 30;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
-------	----	----	------	---------	--------

행	Description	description_cnt
1	WHITE HANGING HEART FLAG...	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

페이지당 결과 수: 50 1 - 30 (전체 30행) < > >|

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE
description = 'Next Day Carriage'
or description = 'High Resolution Image'
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  upper(Description) as Description
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT min(unitprice) AS min_price, max(unitprice) AS max_price, avg(unitprice) AS avg_price
FROM genial-shuttle-439400-e6.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

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

```
SELECT count(unitprice) AS cnt_quantity, min(quantity) AS min_quantity, max(quantity) AS max_quantity
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE unitprice = 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

- UnitPrice = 0를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE genial-shuttle-439400-e6.modulabs_project.data AS
SELECT *
```



```
FROM genial-shuttle-439400-e6.modulabs_project.data
WHERE unitprice != 0;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT date(invoicedate) AS InvoiceDay, *
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터 탐색

작업 정보

결과

자르

JSON

실행 세부정보

실행 그래프

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID
1	2011-11-03	574301	84879	8	2011-11-03 16:15:00 UTC	1.69	12544
2	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC	3.75	12544
3	2011-11-03	574301	22750	4	2011-11-03 16:15:00 UTC	3.75	12544
4	2011-11-03	574301	23514	6	2011-11-03 16:15:00 UTC	2.08	12544
5	2011-11-03	574301	22077	12	2011-11-03 16:15:00 UTC	1.95	12544
6	2011-11-03	574301	22144	6	2011-11-03 16:15:00 UTC	2.1	12544
7	2011-11-03	574301	20749	4	2011-11-03 16:15:00 UTC	7.95	12544

페이지당 결과 수: 50

1 - 50 (전체 399573행)

<

<

>

>

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT max(invoicedate) AS most_recent_date
FROM genial-shuttle-439400-e6.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

데이터

작업 정보

결과

자르

JSON

실행 세부정보

실행 그래프

행	most_recent_date
1	2011-12-09 12:50:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    max(date(invoicedate)) AS InvoiceDay
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	InvoiceDay			
1	12544	2011-11-10			
2	13568	2011-06-19			
3	13824	2011-11-07			
4	14080	2011-11-07			
5	14336	2011-11-23			
6	14592	2011-11-04			
7	15104	2011-06-26			
8	15360	2011-10-31			

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

- 가장 최근 일자(**most_recent_date**)와 유저별 마지막 구매일(**InvoiceDay**)간의 차이를 계산하기

```
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	15361	361			
2	13223	3			
3	14111	14			
4	14371	295			
5	12594	37			
6	13874	358			
7	12865	26			
8	15427	33			

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 genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

i 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	16446	0			
2	13777	0			
3	15311	0			
4	15910	0			
5	17389	0			
6	12433	0			
7	17581	0			
8	12518	0			

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

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
select
  customerid,
  count(distinct invoiceno) as purchase_cnt
from genial-shuttle-439400-e6.modulabs_project.data
group by customerid
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	customerid	purchase_cnt			
1	12544	2			
2	13568	1			
3	13824	5			
4	14080	1			
5	14336	4			
6	14592	3			
7	15104	3			
8	15360	1			

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

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerID
```

[결과 이미지를 넣어주세요]

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			

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 genial-shuttle-439400-e6.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  select
    customerid,
    count(distinct invoiceno) as purchase_cnt
  from genial-shuttle-439400-e6.modulabs_project.data
  group by customerid
),
```

```
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    sum(quantity) AS item_cnt
  FROM genial-shuttle-439400-e6.modulabs_project.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 genial-shuttle-439400-e6.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.

쿼리 결과

결과 저장 데이터 탐색

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt	item_cnt	recency	
1	12713	1	505	0	
2	18010	1	60	256	
3	12792	1	215	256	
4	15083	1	98	256	
5	14569	1	79	1	
6	13298	1	96	1	
7	13436	1	76	1	
8	15520	1	314	1	

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

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  round(sum(unitprice * quantity), 0) AS user_total
FROM genial-shuttle-439400-e6.modulabs_project.data
group by customerid
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	user_total
1	12544	300.0
2	13568	187.0
3	13824	1699.0
4	14080	46.0
5	14336	1615.0
6	14592	558.0
7	15104	969.0
8	15360	428.0

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 genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    round(sum(unitprice * quantity), 0) AS user_total
  FROM genial-shuttle-439400-e6.modulabs_project.data
  group by customerid
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

쿼리 결과

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	795.0	795.0
2	12792	1	215	256	345.0	345.0
3	18010	1	60	256	175.0	175.0
4	15083	1	38	256	88.0	88.0
5	13436	1	76	1	197.0	197.0
6	14569	1	79	1	227.0	227.0
7	15520	1	314	1	344.0	344.0
8	13298	1	96	1	360.0	360.0

페이지당 결과 수: 50 1 - 50 (전체 4362행)

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
select * from genial-shuttle-439400-e6.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

결과

실행 세부정보

실행 그래프

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 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]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

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	18184	1	60	15	50.0	50.0	1
2	16093	1	20	106	17.0	17.0	1
3	15668	1	72	217	76.0	76.0	1
4	16738	1	3	297	4.0	4.0	1
5	18068	1	6	289	102.0	102.0	1
6	17382	1	24	65	50.0	50.0	1
7	15562	1	39	351	135.0	135.0	1
8	14576	1	12	372	35.0	35.0	1

페이지당 결과 수: 50

1 - 50 (전체 4362행)

<

>

>>

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)

- 군 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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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]

쿼리 결과

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

쿼리 결과

결과 저장 데이터 탐색

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프			
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	
1	14432	6	2013	9	2248.0	374.666666666...	256	0.2	
2	12428	11	3477	25	6366.0	578.72727272...	256	0.87	
3	13268	14	3525	17	3106.0	221.8571428571...	256	0.56	
4	17948	1	144	147	359.0	359.0	1	0.0	
5	16257	1	1	176	22.0	22.0	1	0.0	
6	15488	1	72	92	76.0	76.0	1	0.0	
7	14705	1	100	198	179.0	179.0	1	0.0	
8	17102	1	2	261	26.0	26.0	1	0.0	

페이지당 결과 수: 50

1 ~ 50 (전체 4362행)

<

>

>>

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

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 genial-shuttle-439400-e6.modulabs_project.user_data AS

WITH TransactionInfo AS (
  with sup as (
    select distinct invoiceno, customerid
    from genial-shuttle-439400-e6.modulabs_project.data
  )
)
```

```

SELECT
    CustomerID,
    count(distinct invoiceno) AS total_transactions,
    sum(case when invoiceno like 'C%' then 1 else 0 end) AS cancel_frequency
FROM sup
group by customerid
)

SELECT u.*, t.* EXCEPT(CustomerID), round(t.cancel_frequency / t.total_transactions,2) AS cancel_rate
FROM genial-shuttle-439400-e6.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 genial-shuttle-439400-e6.modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과

실행 세부정보

실행 그래프

작업 정보	결과	작업	JSON	실행 세부정보	실행 그래프						
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	15070	1	36	372	106.0	106.0	1	0.0	1	0	0.0
2	16754	1	4289	372	2002.0	2002.0	2	0.0	1	0	0.0
3	16545	1	78	164	339.0	339.0	3	0.0	1	0	0.0
4	16617	1	91	235	152.0	152.0	5	0.0	1	0	0.0
5	15428	1	102	15	94.0	94.0	6	0.0	1	0	0.0
6	14632	1	96	262	115.0	115.0	8	0.0	1	0	0.0
7	16487	1	140	218	208.0	208.0	9	0.0	1	0	0.0
8	17148	1	82	51	125.0	125.0	9	0.0	1	0	0.0
9	13229	1	63	225	161.0	161.0	10	0.0	1	0	0.0
10	15554	1	74	277	217.0	217.0	14	0.0	1	0	0.0
11	12812	1	164	44	230.0	230.0	15	0.0	1	0	0.0

페이지당 결과 수: 50

1 ~ 50 (전체 4362행)

< > >>

회고

sql에 대해서는 어느 정도 공부했었지만 sql의 활용에 대해서는 문제 풀이 말고는 겪어본 적이 없었다.

그래서 오늘 진행한 세그먼테이션 프로젝트가 더 의미 있을 수 있었다.

어제 배웠던 퍼널 분석, 리텐션 분석, rfm 분석 모두 새로 배우는 거라 흥미로웠고,

실제 현업에서 많이 활용되는 rfm 분석에 대해 (실질적으로는 맛보기라고 봐야겠지만) sql 측면에서는 데이터를 정제하고 가공하는 일련의 과정을 경험해 볼 수 있어 좋았다.

사실 데이터사이언스를 공부하면서 어떤 식으로 프로젝트를 진행해야 할지, 어떤 문제를 어떻게 해결할지에 대해서 막막한 부분들이 많았는데,

문제가 주어졌을 때 어떤 식으로 파악하고 문제를 해결할 수 있다는 가이드라인이 어느 정도 머리에 각인되었다.

물론 새로운 문제가 주어졌을 때 척척 풀어낼 수 있다고 보장하지는 못하겠지만

새로운 데이터 분석 과제를 몇 번 더 직접 수행해보면 익숙하게 만들 수 있을 것 같다.

제출 후 문제의 결과값과 내 결과값을 비교해보니 틀린 부분이 발생해서 수정했는데,

데이터 사이언티스트가 가져야 하는 꼼꼼함에 대해 생각을 해보게 하는 계기가 됐다.

토이 데이터로 작업을 할 때는 크게 상관이 없는 문제지만, 대량의 데이터를 다루다 보면 내가 원하지 않는 값이 쿼리 문의 아주 사소한 부분 때문에 추출 될 수 있다.

그런 값들로 인사이트를 도출할 경우, 최종 결론에 심각한 문제가 생길 수 있기 때문에, 항상 내가 만든 로직을 검토하고 또 검토해보아야 한다는 결론이 내려졌다.