

고객을 세그멘테이션하자 [프로젝트] - 혁

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM modulabs_project.data
LIMIT 10
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프			
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT
COUNT(*)
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	1
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT
COUNT(InvoiceNo) AS InvoiceNo_rows,
COUNT(StockCode) AS StockCode_rows,
COUNT(Description) AS Description_rows,
COUNT(Quantity) AS Quantity_rows,
COUNT(InvoiceDate) AS InvoiceDate_rows,
COUNT(UnitPrice) AS UnitPrice_rows,
COUNT(CustomerID) AS CustomerID_rows,
COUNT(Country) AS Country_rows
FROM
modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo_rows	StockCode_rows	Description_rows	Quantity_rows	InvoiceDate_rows	UnitPrice_rows	CustomerID_rows	Country_rows
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

```
# [[YOUR QUERY]]
SELECT column_name, ROUND((total - column_value) / total * 100, 2) AS missing_percentage
FROM
(
    SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM modulabs_project.
data UNION ALL
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM modulabs_project.
data ) AS column_data;
ORDER BY missing_percentage DESC
```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT
    DISTINCT(Description)
FROM
    modulabs_project.data
WHERE
    StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	Description				
1	WHITE HANGING HEART T-LIG...				
2	?				
3	wrongly marked carton 22804				
4	CREAM HANGING HEART T-LIG...				

결측치 처리

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

```
DELETE FROM modulabs_project.data
WHERE # [[YOUR QUERY]];
(CustomerID IS NULL)
OR
(Description IS NULL);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT
COUNT(duplicated_cnt) AS total_duplicate_rows
FROM(
SELECT
COUNT(*) AS duplicated_cnt
FROM modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING COUNT(*) > 1
)AS DuplicateCounts;
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 시각화 JSON 실행 세부정보 실행 그래프

행	total_duplicate_r...
1	4837

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 modulabs_project.data AS
SELECT DISTINCT *
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

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

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT(InvoiceNo))
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	f0_				
1	22190				

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
# [[YOUR QUERY]]
SELECT
  DISTINCT(InvoiceNo)
FROM
  modulabs_project.data
ORDER BY
  InvoiceNo ASC
LIMIT
  100;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	536365				
2	536366				
3	536367				
4	536368				
5	536369				
6	536370				
7	536371				
8	536372				
9	536373				
10	536374				
11	536375				
12	536376				
13	536377				
14	536378				

페이지당 결과 수: 50 1 ~ 50 (전체 100행) |< < > >|

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
87	536559				
88	536560				
89	536561				
90	536562				
91	536563				
92	536564				
93	536566				
94	536567				
95	536568				
96	536569				
97	536570				
98	536571				
99	536572				
100	536573				

페이지당 결과 수: 50 51 ~ 100 (전체 100행) |< < > >|

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

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

[결과 이미지를 넣어주세요]

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5
5	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45
6	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95
7	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95
8	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25
9	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49
10	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65
11	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95

페이지당 결과 수: 501 - 50 (전체 100행)

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

```
SELECT
  ROUND(SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM
  modulabs_project.data
```

[결과 이미지를 넣어주세요]

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

행	missing_percenta...
1	2.21

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT COUNT(DISTINCT(StockCode))
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

작업 정보결과시각화JSON실행 세부정보실행 그래프

f0_

13684

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 **StockCode** 별 등장 빈도를 출력하기
 - 상위 10개의 제품들을 출력하기

```
SELECT
  StockCode,
  COUNT(*) AS sell_cnt
FROM
  modulabs_project.data
GROUP BY
  StockCode
```

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

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	StockCode	sell_cnt			
1	85123A	2065			
2	22423	1894			
3	85099B	1659			
4	47566	1409			
5	84879	1405			
6	20725	1346			
7	22720	1224			
8	POST	1196			
9	22197	1110			
10	23203	1108			

- **StockCode** 의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
 - 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM project_name.modulabs_project.data
)
WHERE number_count between 0 and 1;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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			

- **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 between 0 and 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS percentage
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
```

```
FROM modulabs_project.data
)
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	percentage				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    # [[YOUR QUERY]]
    SELECT StockCode
    FROM
      (SELECT StockCode,
        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
      FROM modulabs_project.data)
    WHERE number_count between 0 and 1
  );
```

[결과 이미지를 넣어주세요]

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 modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30 ;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	Description ▾	description_cnt ▾			
1	WHITE HANGING HEART T-LIG...	2058			
2	REGENCY CAKESTAND 3 TIER	1894			
3	JUMBO BAG RED RETROSPOT	1659			
4	PARTY BUNTING	1409			
5	ASSORTED COLOUR BIRD ORN...	1405			
6	LUNCH BAG RED RETROSPOT	1345			
7	SET OF 3 CAKE TINS PANTRY D...	1224			
8	LUNCH BAG BLACK SKULL.	1099			
9	PACK OF 72 RETROSPOT CAKE ...	1062			
10	SPOTTY BUNTING	1026			
11	PAPER CHAIN KIT 50'S CHRIST...	1013			
12	LUNCH BAG SPACEBOY DESIGN	1006			
13	LUNCH BAG CARS BLUE	1000			
14	HEART OF WICKER SMAI I	990			

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	Description ▾	description_cnt ▾			
17	LUNCH BAG PINK POLKADOT	961			
18	LUNCH BAG SUKI DESIGN	932			
19	ALARM CLOCK BAKELIKE RED	917			
20	REX CASH+CARRY JUMBO SHO...	900			
21	WOODEN PICTURE FRAME WHI...	900			
22	JUMBO BAG PINK POLKADOT	897			
23	LUNCH BAG APPLE DESIGN	890			
24	SET OF 4 PANTRY JELLY MOUL...	890			
25	BAKING SET 9 PIECE RETROSP...	885			
26	JAM MAKING SET PRINTED	883			
27	RECIPE BOX PANTRY YELLOW ...	883			
28	LUNCH BAG WOODLAND	850			
29	ROSES REGENCY TEACUP AND ...	844			
30	VICTORIAN GLASS HANGING T...	843			

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM modulabs_project.data
WHERE
Description LIKE('Next%') OR Description LIKE('High%')
```

[결과 이미지를 넣어주세요]


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 modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	실행 세부정보	실행 그래프
<div>  이 문으로 이름이 data인 테이블이 교체되었습니다. </div>			

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 modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	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 최소값, 최대값, 평균 구하기

```
SELECT COUNT(Quantity) cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS a
vg_quantity
FROM modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
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 modulabs_project.data AS
SELECT *
FROM 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 modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보		결과	시각화	JSON	실행 세부정보	실행 그래프	
행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	
3	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	
4	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	1.25	
5	2010-12-07	537626	849978	6	2010-12-07 14:57:00 UTC	3.75	
6	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	
7	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	
8	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	
9	2010-12-07	537626	22772	12	2010-12-07 14:57:00 UTC	1.25	

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  MAX(InvoiceDate) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay
FROM modulabs_project.data
GROUP BY InvoiceDate
ORDER BY InvoiceDate DESC
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	most_recent_date ▼	InvoiceDay ▼			
1	2011-12-09 12:50:00 UTC	2011-12-09			
2	2011-12-09 12:49:00 UTC	2011-12-09			
3	2011-12-09 12:31:00 UTC	2011-12-09			
4	2011-12-09 12:25:00 UTC	2011-12-09			
5	2011-12-09 12:23:00 UTC	2011-12-09			
6	2011-12-09 12:21:00 UTC	2011-12-09			
7	2011-12-09 12:20:00 UTC	2011-12-09			
8	2011-12-09 12:19:00 UTC	2011-12-09			
9	2011-12-09 12:16:00 UTC	2011-12-09			
10	2011-12-09 12:09:00 UTC	2011-12-09			
11	2011-12-09 12:08:00 UTC	2011-12-09			
12	2011-12-09 12:00:00 UTC	2011-12-09			
13	2011-12-09 11:59:00 UTC	2011-12-09			
14	2011-12-09 11:58:00 UTC	2011-12-09			
15	2011-12-09 11:57:00 UTC	2011-12-09			

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID ▼	InvoiceDay ▼			
34	12386	2011-01-06			
35	12388	2011-11-24			
36	12390	2011-09-21			
37	12391	2011-11-18			
38	12393	2011-09-28			
39	12394	2011-10-07			
40	12395	2011-11-24			
41	12397	2011-11-04			
42	12398	2011-10-25			
43	12399	2011-08-12			
44	12401	2011-02-09			
45	12402	2011-01-20			
46	12403	2011-10-21			
47	12405	2011-07-14			
48	12406	2011-11-17			

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]

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	12414	217			
2	12658	18			
3	12670	10			
4	12971	3			
5	13356	80			
6	13448	16			
7	13884	7			
8	14030	18			
9	14381	52			
10	14627	311			
11	14753	87			
12	14765	29			
13	14915	85			
14	14953	35			

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 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 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.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  Count(InvoiceNo) AS purchase_cnt
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 다시

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID	purchase_cnt			
1	12346	2			
2	12347	182			
3	12348	27			
4	12349	72			
5	12350	16			
6	12352	84			
7	12353	4			
8	12354	58			
9	12355	13			
10	12356	58			
11	12357	131			
12	12358	17			
13	12359	251			

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  Count(Quantity) AS item_cnt
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 다시

작업 정보	결과	시각화	JSON	실행 세부정보	실행 그래프
행	CustomerID	item_cnt			
36	12390	31			
37	12391	93			
38	12393	64			
39	12394	25			
40	12395	147			
41	12397	124			
42	12398	84			
43	12399	55			
44	12401	4			
45	12402	10			
46	12403	4			
47	12405	53			
48	12406	108			
49	12407	72			
50	12408	108			

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 modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    Count(InvoiceNo) AS purchase_cnt
  FROM modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    Count(Quantity) AS item_cnt
  FROM 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 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.

스키마 세부정보 **미리보기** 테이블 탐색기 프리뷰

행	CustomerID	purchase_cnt	item_cnt	recency
1	13120	1	1	238
2	16995	1	1	372
3	17948	1	1	147
4	14705	1	1	198
5	13135	1	1	196
6	14679	1	1	371
7	12943	1	1	301
8	15118	1	1	134
9	17331	1	1	123
10	13017	1	1	7
11	13307	1	1	120
12	14424	1	1	17
13	13841	1	1	252
14	15510	1	1	330
15	18113	1	1	368
16	16061	1	1	269
17	17925	1	1	372
18	15657	1	1	22

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity),1) AS user_total
FROM modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1
13	12359	6183.0
14	12360	2302.1
15	12361	174.9

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_cnt로 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,
  # [[YOUR QUERY]] AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    # [[YOUR QUERY]]
  ) 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.

user_rfm

쿼리

다음에서 열기

공유

복사

스냅샷

식재

내보내기

스키마

세부정보

미리보기

테이블 탐색기

프리뷰

통계

계보

데이터 프로필

데이터 품질

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	
1	14424	1	1	17	322.1	322.1	
2	17102	1	1	261	25.5	25.5	
3	16738	1	1	297	3.8	3.8	
4	15316	1	1	326	165.0	165.0	
5	13135	1	1	196	3096.0	3096.0	
6	18113	1	1	368	76.3	76.3	
7	17948	1	1	147	358.6	358.6	
8	12943	1	1	301	-3.8	-3.8	
9	14090	1	1	324	76.3	76.3	
10	15668	1	1	217	76.3	76.3	
11	13829	1	1	359	-102.0	-102.0	
12	13270	1	1	366	590.0	590.0	
13	16454	1	1	64	5.9	5.9	
14	13366	1	1	50	56.2	56.2	
15	12814	1	1	101	85.9	85.9	
16	17925	1	1	372	244.1	244.1	
17	16737	1	1	53	417.6	417.6	
18	13307	1	1	120	15.0	15.0	

페이지당 결과 수:

50

1 - 50 (전체 4362행)

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보

결과

시각화

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	
1	14424	1	1	17	322.1	322.1	
2	17102	1	1	261	25.5	25.5	
3	16738	1	1	297	3.8	3.8	
4	15316	1	1	326	165.0	165.0	
5	13135	1	1	196	3096.0	3096.0	
6	18113	1	1	368	76.3	76.3	
7	17948	1	1	147	358.6	358.6	
8	12943	1	1	301	-3.8	-3.8	
9	14090	1	1	324	76.3	76.3	
10	15668	1	1	217	76.3	76.3	
11	13829	1	1	359	-102.0	-102.0	
12	13270	1	1	366	590.0	590.0	
13	16454	1	1	64	5.9	5.9	
14	13366	1	1	50	56.2	56.2	

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 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.

스키마	세부정보	미리보기	테이블 탐색기	프리뷰	통계	개보	데이터 프로필	데이터 품질
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	
1	13339	54	54	200	860.1	15.92777777...	54	
2	12772	54	54	59	752.5	13.93518518...	54	
3	16270	54	54	353	1141.2	21.13333333...	54	
4	13979	54	54	73	869.9	16.10925925...	54	
5	13439	54	54	255	283.7	5.253703703...	54	
6	15832	54	54	254	836.8	15.49629629...	54	
7	14953	54	54	25	285.7	5.290740740...	54	
8	13122	55	55	94	922.4	16.77090909...	54	
9	17608	56	56	33	192.0	3.428571428...	54	
10	16406	58	58	18	154.4	2.662068965...	54	
11	13107	60	60	44	1524.1	25.40166666...	54	
12	17832	61	61	49	155.4	2.547540983...	61	
13	16024	61	61	12	245.7	4.027868852...	60	
14	13635	62	62	67	1071.0	17.27419354...	62	
15	12534	62	62	130	981.2	15.82580645...	62	
16	12371	62	62	59	1528.0	24.64516129...	62	
17	16685	62	62	61	324.2	5.229032258...	60	
18	17070	62	62	114	304.2	4.906451612...	62	

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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 interval_
    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.

스키마	세부정보	미리보기	테이블 탐색기	프리뷰	통계	계보	데이터 프로필	데이터 품질
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	
1	13339	54	54	200	860.1	15.92777777...	54	
2	12772	54	54	59	752.5	13.93518518...	54	
3	16270	54	54	353	1141.2	21.13333333...	54	
4	13979	54	54	73	869.9	16.10925925...	54	
5	13439	54	54	255	283.7	5.253703703...	54	
6	15832	54	54	254	836.8	15.49629629...	54	
7	14953	54	54	25	285.7	5.290740740...	54	
8	13122	55	55	94	922.4	16.77090909...	54	
9	17608	56	56	33	192.0	3.428571428...	54	
10	16406	58	58	18	154.4	2.662068965...	54	
11	13107	60	60	44	1524.1	25.40166666...	54	
12	17832	61	61	49	155.4	2.547540983...	61	
13	16024	61	61	12	245.7	4.027868852...	60	
14	13635	62	62	67	1071.0	17.27419354...	62	
15	12534	62	62	130	981.2	15.82580645...	62	
16	12371	62	62	59	1528.0	24.64516129...	62	
17	16685	62	62	61	324.2	5.229032258...	60	
18	17070	62	62	114	304.2	4.906451612...	62	

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 modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNTIF(LEFT(InvoiceNo, 1) = 'C' AND Quantity < 0) AS cancel_frequency
  FROM modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID),
  ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM 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



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마	세부정보	미리보기	테이블 탐색기	프리뷰	통계	계보	데이터 프로파일	데이터 품질
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod...	average_inter...
1	16692	2	2	256	1126.0	563.0	1	32.0
2	13452	2	2	259	590.0	295.0	1	11.0
3	12875	2	2	143	343.2	171.6	1	219.0
4	17186	2	2	46	144.0	72.0	1	13.0
5	16073	2	2	291	94.3	47.15	1	5.0
6	18084	2	2	16	87.5	43.75	2	284.0
7	14682	2	2	187	52.0	26.0	1	13.0
8	15649	2	2	336	816.0	408.0	1	31.0
9	13068	2	2	10	344.0	172.0	1	309.0
10	18080	2	2	18	1231.5	615.75	2	223.0
11	17747	2	2	100	64.7	32.35	1	12.0
12	13106	2	2	128	59.5	29.75	1	4.0
13	17029	2	2	110	716.0	358.0	1	126.0
14	18273	3	3	2	204.0	68.0	1	127.5
15	14616	3	3	242	406.1	135.3666666...	2	36.5
16	16716	3	3	266	319.8	106.6000000...	1	22.0
17	17616	3	3	173	571.2	190.4	1	96.5
18	16506	3	3	19	90.3	30.09999999...	3	10.0

페이지당 결과 수: 50 1 - 50 (전체 4362행)

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

```
SELECT *
FROM modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

쿼리 결과 쿼리 저장

작업 정보 결과 시각화 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	16692	2	2	256	1126.0	563.0	1	32.0
2	13452	2	2	259	590.0	295.0	1	11.0
3	12875	2	2	143	343.2	171.6	1	219.0
4	17186	2	2	46	144.0	72.0	1	13.0
5	16073	2	2	291	94.3	47.15	1	5.0
6	18084	2	2	16	87.5	43.75	2	284.0
7	14682	2	2	187	52.0	26.0	1	13.0
8	15649	2	2	336	816.0	408.0	1	31.0
9	13068	2	2	10	344.0	172.0	1	309.0
10	18080	2	2	18	1231.5	615.75	2	223.0
11	17747	2	2	100	64.7	32.35	1	12.0
12	13106	2	2	128	59.5	29.75	1	4.0
13	17029	2	2	110	716.0	358.0	1	126.0

페이지당 결과 수: 50 1 - 50 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : SQL 문법이 아주 조금은 익숙해 진거같아서, 조금더 잘쓰게 노력해야 될거같다. 특히 이번 과제에서 배운 백분율 계산법이나 문자중 숫자가 있는 문자열 처리등은 앞으로든 매우 유용하게 쓸수 있을거같다.

Problem : 데이터 전처리에 시간이 많이 걸렸다. 중간에 잘못된 쿼리를 넣어서 Data 테이블이 잘못되어, 기존 테이블을 Drop하고 다시 테이블을 생성한 게 작업시간이 오래걸렸다.

Try : 아직 부족한 SQL 구문처리를 더 연마해야겠다. 일단 쿼리시간이 10초가까이 걸리는 것들도 있었다. 시간을 줄이도록 노력해야 겠다. 처음으로 큰 데이터를 다루어 봤는데, 클라우드에서 쿼리용량에 따라 비용이 발생하는 만큼, 쿼리 용량이나 횟수를 줄이려고 SQL구문을 더 최적화 하는 방법을 연마해야겠다. 이번에는 많으면 30MB정도가 1회 쿼리에 필요했는데, SQL구문을 더 효율적으로 쓰면 10MB까지 줄일수 있지 않을까 생각된다.