

고객을 세그멘테이션하자 [프로젝트]

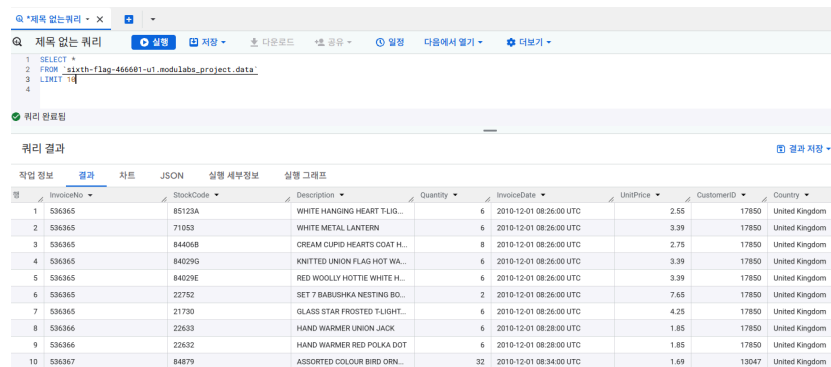
11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM `sixth-flag-466601-u1.modulabs_project.data`
LIMIT 10
```

[결과 이미지를 넣어주세요]



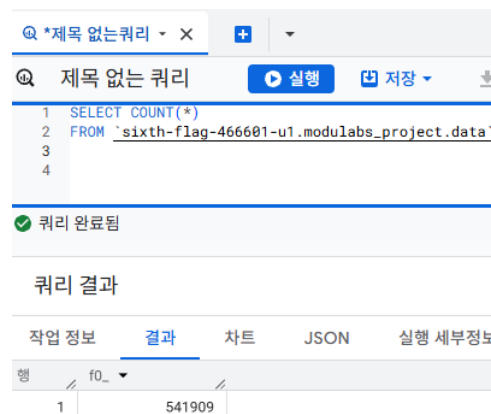
The screenshot shows a BigQuery interface with a query editor and a results table. The query is: `SELECT * FROM `sixth-flag-466601-u1.modulabs_project.data` LIMIT 10`. The results table has 10 rows and 10 columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. The first row is: 1, 536365, 85123A, WHITE HANGING HEART T.LIG., 6, 2019-12-01 08:26:00 UTC, 2.55, 17850, United Kingdom.

명	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T.LIG.	6	2019-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2019-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	844068	CREAM CUPID HEARTS COAT H...	8	2019-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2019-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2019-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2019-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T.LIGHT...	6	2019-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22833	HAND WARMER UNION JACK	6	2019-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22832	HAND WARMER RED POLKA DOT	6	2019-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2019-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT COUNT(*)
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



The screenshot shows a BigQuery interface with a query editor and a results table. The query is: `SELECT COUNT(*) FROM `sixth-flag-466601-u1.modulabs_project.data``. The results table has 1 row and 1 column: f0_1, 541909.

행	f0_1
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
SELECT COUNT(InvoiceNo) COUNT_InvoiceNo, COUNT(StockCode) COUNT_StockCode, COUNT(Description) COUNT_Descrip
```



```

SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `sixth-flag-466601-u1.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 `sixth-flag-466601-u1.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

The screenshot shows a SQL query execution interface. The query is a UNION ALL of three SELECT statements, each calculating the missing percentage for a different column: InvoiceNo, StockCode, and Description. The results table shows the missing percentages for these columns and others.

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

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE StockCode = '85123A';

```

[결과 이미지를 넣어주세요]

*제목 없는 쿼리

제목 없는 쿼리 실행 저장 다운로드 공유

```

1 SELECT DISTINCT Description
2 FROM `sixth-flag-466601-u1.modulabs_project.data`
3 WHERE StockCode = '85123A'

```

실행 시 이 쿼리가 18.43MB를 처리합니다.

쿼리 결과

작업 정보 결과 차트 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 사용하여, WHERE 절을 통해 데이터를 제거할 조건을 제시

```

DELETE FROM project_name.modulabs_project.data
WHERE CustomerID IS NULL
OR Description IS NULL;

```

[결과 이미지를 넣어주세요]

*제목 없는 쿼리

제목 없는 쿼리 실행 저장 다운로드 공유

```

1 DELETE FROM `sixth-flag-466601-u1.modulabs_project.data`
2 WHERE CustomerID IS NULL
3 OR Description IS NULL

```

쿼리 완료됨

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 project_name.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	571034	23494	VINTAGE DOILY DELUXE SEWIN...	3	2011-10-13 12:47:00 UTC	
2	571034	23245	SET OF 3 REGENCY CAKE TINS	4	2011-10-13 12:47:00 UTC	
3	571034	23239	SET OF 4 KNICK KNACK TINS P...	6	2011-10-13 12:47:00 UTC	
4	538826	22749	FELTCRAFT PRINCESS CHARLO...	1	2010-12-14 12:58:00 UTC	
5	577228	23048	SET OF 10 LANTERNS FAIRY LI...	1	2011-11-18 12:07:00 UTC	
6	577228	22435	SET OF 9 HEART SHAPED BALL...	1	2011-11-18 12:07:00 UTC	
7	577228	84580	MOUSE TOY WITH PINK T-SHIRT	1	2011-11-18 12:07:00 UTC	
8	577228	23156	SET OF 5 MINI GROCERY MAG...	1	2011-11-18 12:07:00 UTC	
9	577228	22270	HAPPY EASTER HANGING DEC...	1	2011-11-18 12:07:00 UTC	
10	577228	22144	CHRISTMAS CRAFT LITTLE FRI...	1	2011-11-18 12:07:00 UTC	
11	539419	48138	DOORMAT UNION FLAG	10	2010-12-17 14:10:00 UTC	
12	538174	22326	ROUND SNACK BOXES SET OF4...	12	2010-12-10 09:35:00 UTC	
13	555162	22704	WRAP RED APPLES	25	2011-06-01 10:15:00 UTC	

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 `sixth-flag-466601-u1.modulabs_project.data` AS
SELECT DISTINCT *
FROM `sixth-flag-466601-u1.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

Q *제목 없는 쿼리

제목 없는 쿼리 실행 저장 다운로드 공유 일정 다음에서 열기 더보기

```

1 CREATE OR REPLACE TABLE `sixth-flag-466601-u1.modulabs_project.data` AS
2 SELECT DISTINCT *
3 FROM `sixth-flag-466601-u1.modulabs_project.data`

```

✓ 쿼리 완료됨

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

Q *제목 없는 쿼리

제목 없는 쿼리 실행 저장 다운로드

```

1 SELECT COUNT(*)
2 FROM sixth-flag-466601-u1.modulabs_project.data

```

✓ 쿼리 완료됨

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보

행	f0_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```

# [[YOUR QUERY]]
SELECT COUNT(DISTINCT InvoiceNo)
FROM `sixth-flag-466601-u1.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

```

1 SELECT COUNT(DISTINCT InvoiceNo)
2 FROM `sixth-flag-466601-u1.modulabs_project.data`

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	1	22190		

- 고유한 **InvoiceNo** 를 앞에서부터 100개를 출력하기

```

# [[YOUR QUERY]]
SELECT DISTINCT InvoiceNo
FROM `sixth-flag-466601-u1.modulabs_project.data`
LIMIT 100

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

```

1 SELECT DISTINCT InvoiceNo
2 FROM `sixth-flag-466601-u1.modulabs_project.data`
3 LIMIT 100

```

실행 시 이 쿼리가 3.07MB를 처리합니다.

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	541431				
2	C541433				
3	537626				
4	542237				
5	549222				
6	556201				
7	562032				
8	573511				
9	581180				
10	539318				
11	541998				
12	548955				
13	568172				

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 InvoiceNo LIKE 'C%'
LIMIT 100;

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

제목 없는 쿼리

실행

저장

다른 쿼리

공유

일정

다음에서 열기

더보기

```
1 SELECT *
2 FROM `sixth-flag-466601-u1.modulabs_project.data`
3 WHERE InvoiceNo LIKE 'C%'
4 LIMIT 100
```

쿼리 완료됨

쿼리 결과

결과 저장

작업 정보

결과

차트

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	
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	
5	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	
7	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	
8	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	
9	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	
10	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	
11	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	
12	C549955	22666	RECIPE BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC	
13	C549955	22839	3 TIER CAKE TIN GREEN AND C...	-2	2011-04-13 13:38:00 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]

제목 없는 쿼리

```

1 SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 1)
2 FROM `sixth-flag-466601-u1.modulabs_project.data`
3

```

쿼리 완료됨

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	f0_				
1	2.2				

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```

# [[YOUR QUERY]]
SELECT COUNT(DISTINCT StockCode)
FROM `sixth-flag-466601-u1.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

*제목 없는쿼리 - X	
제목 없는 쿼리	실행 저장
<pre> 1 SELECT COUNT(DISTINCT StockCode) 2 FROM `sixth-flag-466601-u1.modulabs_project.data` 3 </pre>	
쿼리 결과	
작업 정보 결과 차트 JSON 실행 세부정보	
행	f0_
1	3684

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

```

SELECT StockCode, COUNT(*) AS sell_cnt
FROM project_name.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10

```

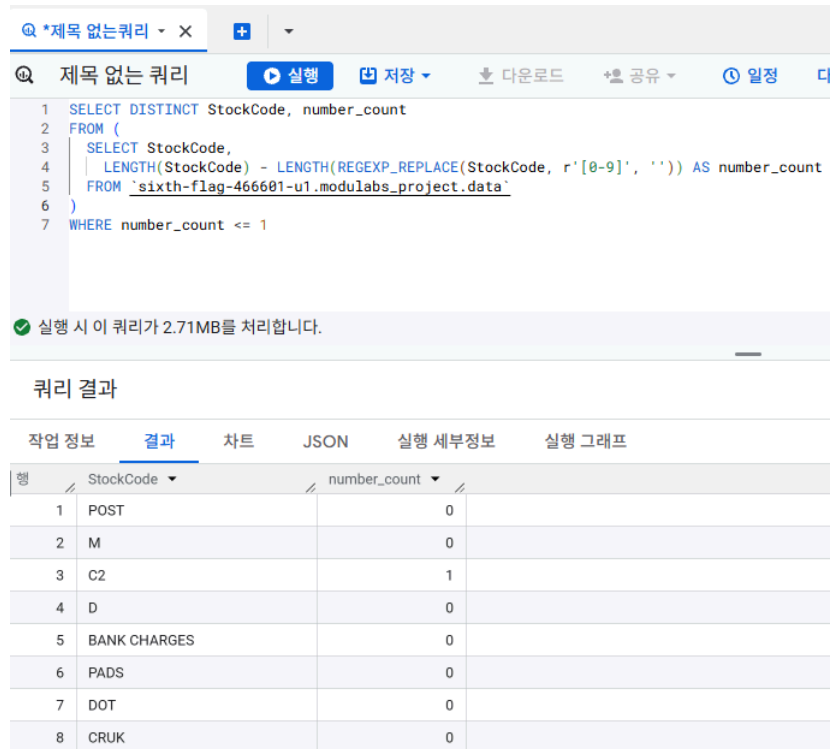
[결과 이미지를 넣어주세요]

*제목 없는쿼리 - X	
제목 없는 쿼리	실행 저장
<pre> 1 SELECT StockCode, COUNT(*) 2 FROM `sixth-flag-466601-u1.modulabs_project.data` 3 GROUP BY StockCode 4 ORDER BY COUNT(*) DESC 5 LIMIT 10 6 </pre>	
<div> 실행 시 이 쿼리가 2.71MB를 처리합니다. </div>	
쿼리 결과	
작업 정보 결과 차트 JSON 실행 세부정보	
행	StockCode f0_
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 <= 1
```

[결과 이미지를 넣어주세요]



The screenshot shows a SQL query execution interface. The query is as follows:

```
1 SELECT DISTINCT StockCode, number_count
2 FROM (
3   SELECT StockCode,
4     LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
5   FROM `sixth-flag-466601-u1.modulabs_project.data`
6 )
7 WHERE number_count <= 1
```

Below the query, a status message indicates: "실행 시 이 쿼리가 2.71MB를 처리합니다." (When executed, this query will process 2.71MB).

The results are displayed in a table titled "쿼리 결과" (Query Results). The table has two columns: "StockCode" and "number_count".

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 <= 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 `sixth-flag-466601-u1.modulabs_project.data`)
```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

```

1 -- 각 행별로 숫자가 몇개인지 카운트하고, 그 카운트가 1 이하인 행들의 개수를 전체 행의 개수로 나눈다
2 SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
3 FROM (SELECT StockCode,
4         LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
5      FROM `sixth-flag-466601-u1.modulabs_project.data`)
6

```

실행 시 이 쿼리가 2.71MB를 처리합니다.

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	f0_
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM sixth-flag-466601-u1.modulabs_project.data
  WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1)

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

```

1 DELETE FROM sixth-flag-466601-u1.modulabs_project.data
2 WHERE StockCode IN (
3   SELECT DISTINCT StockCode
4   FROM sixth-flag-466601-u1.modulabs_project.data
5   WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1)
6

```

쿼리 완료됨

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

SELECT Description, COUNT(*) AS description_cnt
FROM project_name.modulabs_project.data
GROUP BY Description
ORDER BY COUNT(*) DESC
LIMIT 30

```

[결과 이미지를 넣어주세요]

쿼리 실행 결과

```

1 SELECT Description, COUNT(*) description_cnt
2 FROM sixth-flag-466601-u1.modulabs_project.data
3 GROUP BY Description
4 ORDER BY COUNT(*) DESC
5 LIMIT 30
6

```

쿼리 완료됨

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	Description	description_cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224
8	LUNCH BAG BLACK SKULL	1099
9	PACK OF 72 RETROSPOT CAKE CASES	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRISTMAS	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990

페이지당 결과 수: 50 1 - 30 (전체 30행)

서비스 관련 정보를 포함하는 행들을 제거하기

```

DELETE
FROM project_name.modulabs_project.data
WHERE
Description IN ('High Resolution Image', 'Next Day Carriage')

```

[결과 이미지를 넣어주세요]

쿼리 실행 결과

```

1 DELETE
2 FROM sixth-flag-466601-u1.modulabs_project.data
3 WHERE
4 Description IN ('High Resolution Image', 'Next Day Carriage')

```

실행 시 이 쿼리가 34.7MB를 처리합니다.

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 data의 행 83개가 삭제되었습니다.

대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

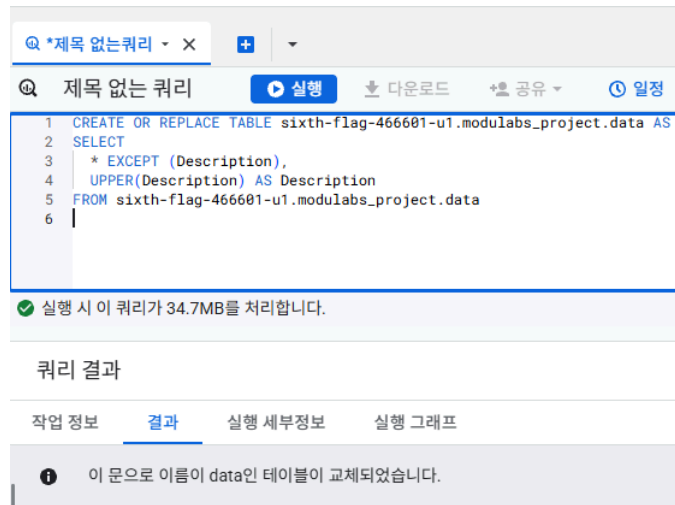
```

CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT

```

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

[결과 이미지를 넣어주세요]



UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
SELECT 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]

The screenshot shows a SQL query execution interface. At the top, there's a search bar with the text '*제목 없는쿼리' and a dropdown menu. Below it, the query text is displayed:


```
1 SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
2 FROM sixth-flag-466601-u1.modulabs_project.data
3
```

 Below the query text, there's a status bar indicating '실행 시 이 쿼리가 3.05MB를 처리합니다.' (When executed, this query will process 3.05MB). Underneath, the title '쿼리 결과' (Query Result) is shown. At the bottom, there's a tabbed interface with '작업 정보' (Job Info), '결과' (Result), '차트' (Chart), 'JSON', '실행 세부정보' (Execution Details), and '실행 그래프' (Execution Graph). The '결과' tab is selected, showing a table with the following data:

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(Quantity) cnt_quantity, MIN(Quantity) min_quantity, MAX(Quantity) max_quantity, AVG(Quantity) avg_quantity
FROM project_name.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

*제목 없는쿼리 - X				
제목 없는 쿼리 실행 다운로드 공유 일정 다음에서 열기 더보기 저장				
<pre> 1 SELECT COUNT(Quantity) cnt_quantity, MIN(Quantity) min_quantity, MAX(Quantity) max_quantity, AVG(Quantity) avg_quantity 2 FROM sixth-flag-466601-u1.modulabs_project.data 3 WHERE UnitPrice = 0 4 </pre>				
실행 시 이 쿼리가 6.1MB를 처리합니다.				
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 = 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]

*제목 없는쿼리 - X				
제목 없는 쿼리 실행 다운로드 공유 일정				
<pre> 1 CREATE OR REPLACE TABLE `sixth-flag-466601-u1.modulabs_project.data` AS 2 SELECT * 3 FROM `sixth-flag-466601-u1.modulabs_project.data` 4 WHERE UnitPrice != 0 5 </pre>				
실행 시 이 쿼리가 34.7MB를 처리합니다.				
쿼리 결과				
작업 정보 결과 실행 세부정보 실행 그래프				
이 문으로 이름이 data인 테이블이 교체되었습니다.				

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```

SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;

```

[결과 이미지를 넣어주세요]

Q *제목 없는쿼리 - X

제목 없는 쿼리 실행 호 다운로드 *호 공유 일정 다음에서 열기 더보기 저장

```

1 SELECT DATE(InvoiceDate) AS InvoiceDay,
2 FROM sixth-flag-466601-u1.modulabs_project.data
3

```

실행 시 이 쿼리가 34.7MB를 처리합니다.

쿼리 결과 결과 저장

작업 정보 결과 차트 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	1;
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	1;
3	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	1;
4	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	1;
5	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	1;
6	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	1;
7	2010-12-07	537626	84997B	6	2010-12-07 14:57:00 UTC	3.75	1;
8	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC	3.25	1;
9	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	1;
10	2010-12-07	537626	22494	12	2010-12-07 14:57:00 UTC	1.25	1;
11	2010-12-07	537626	85116	12	2010-12-07 14:57:00 UTC	2.1	1;
12	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	1;
13	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	1;
14	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	1;

페이지당 결과 수: 50 1 - 50 (전체 399573행)

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```

SELECT
  MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM project_name.modulabs_project.data;

```

[결과 이미지를 넣어주세요]

Q *제목 없는쿼리 - X

제목 없는 쿼리 실행 호 다운로드 *호 공유 일정 다음에서 열기 더보기 저장

```

1 SELECT
2   MAX(DATE(InvoiceDate)) OVER() AS most_recent_date,
3   DATE(InvoiceDate) AS InvoiceDay,
4   *
5 FROM sixth-flag-466601-u1.modulabs_project.data

```

실행 시 이 쿼리가 34.7MB를 처리합니다.

쿼리 결과 결과 저장

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice
1	2011-12-09	2011-09-26	568223	84031A	3	2011-09-26 10:16:00 UTC	;
2	2011-12-09	2011-05-06	552202	47590A	3	2011-05-06 14:51:00 UTC	;
3	2011-12-09	2011-02-15	543989	22423	64	2011-02-15 09:52:00 UTC	1;
4	2011-12-09	2011-03-03	545475	21981	432	2011-03-03 10:59:00 UTC	;
5	2011-12-09	2011-10-05	569650	22725	50	2011-10-05 12:44:00 UTC	;
6	2011-12-09	2011-09-21	567526	23342	36	2011-09-21 09:06:00 UTC	;
7	2011-12-09	2011-09-21	567526	22192	36	2011-09-21 09:06:00 UTC	;
8	2011-12-09	2011-10-14	571255	23055	48	2011-10-14 17:13:00 UTC	;
9	2011-12-09	2011-02-14	C543830	22847	-1	2011-02-14 09:54:00 UTC	1;
10	2011-12-09	2011-07-19	560590	22847	1	2011-07-19 15:55:00 UTC	1;
11	2011-12-09	2011-11-17	576927	23068	8	2011-11-17 11:02:00 UTC	;
12	2011-12-09	2011-01-07	540455	21883	144	2011-01-07 12:07:00 UTC	;
13	2011-12-09	2011-09-11	566233	21402	24	2011-09-11 11:04:00 UTC	;
14	2011-12-09	2011-10-19	571923	21843	2	2011-10-19 16:24:00 UTC	1;

페이지당 결과 수: 50 1 - 50 (전체 399573행)

- 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]

제목 없는 쿼리

실행 다운로드

```

1 SELECT
2   CustomerID,
3   MAX(DATE(InvoiceDate)) AS InvoiceDay
4 FROM `sixth-flag-466601-u1.modulabs_project.data`
5 GROUP BY CustomerID
6

```

실행 시 이 쿼리가 6.1MB를 처리합니다.

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18

- 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]

제목 없는 쿼리

```

1 SELECT
2   CustomerID,
3   EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
4 FROM (
5   SELECT
6     CustomerID,
7     MAX(DATE(InvoiceDate)) AS InvoiceDay
8   FROM sixth-flag-466601-u1.modulabs_project.data
9   GROUP BY CustomerID
10 )

```

쿼리 완료됨

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	12375	2			
2	12415	24			
3	12778	19			
4	12808	36			
5	12870	366			
6	12908	176			
7	12923	64			
8	13090	8			
9	13092	70			
10	13094	21			
11	13210	93			
12	13226	273			
13	13313	22			

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 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 `sixth-flag-466601-u1.modulabs_project.data`
  GROUP BY CustomerID
)

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

```

1 CREATE OR REPLACE TABLE sixth-flag-466601-u1.modulabs_project.user_r AS
2 SELECT
3   CustomerID,
4   EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
5 FROM (
6   SELECT
7     CustomerID,
8     MAX(DATE(InvoiceDate)) AS InvoiceDay
9   FROM `sixth-flag-466601-u1.modulabs_project.data`
10  GROUP BY CustomerID
11 )

```

실행 시 이 쿼리가 6.1MB를 처리합니다.

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.			

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM project_name.modulabs_project.data
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

The screenshot shows a SQL query execution interface. The query is: `SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt FROM sixth-flag-466601-u1.modulabs_project.data GROUP BY CustomerID`. The results are displayed in a table with 13 rows and 2 columns: CustomerID and purchase_cnt. The table is titled '쿼리 결과' (Query Results). The results are as follows:

CustomerID	purchase_cnt
12346	2
12347	7
12348	4
12349	1
12350	1
12352	8
12353	1
12354	1
12355	1
12356	3
12357	1
12358	2
12359	6

페이지당 결과 수: 50 1 - 50 (전체 4362행)

- 각 고객 별로 구매한 아이템의 총 수량 더하기

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

[결과 이미지를 넣어주세요]

제목 없는 쿼리

실행 다운로드 공유 일정 다음에서 열기 더보기 저장

```

1 SELECT
2   CustomerID,
3   SUM(Quantity) AS item_cnt
4 FROM sixth-flag-466601-u1.modulabs_project.data
5 GROUP BY CustomerID
6

```

실행 시 이 쿼리가 6.1MB를 처리합니다.

쿼리 결과

결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573
11	12357	2708
12	12358	242
13	12359	1599

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 project_name.modulabs_project.user_rf AS
```

```
-- (1) 전체 거래 건수 계산
```

```
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
```

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

```
-- 기존의 user_r에 (1)과 (2)를 통합
```

```
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.reccency
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]

제목 없는 쿼리

```

1 CREATE OR REPLACE TABLE `sixth-flag-466601-u1.modulabs_project.user_rf` AS
2
3 -- (1) 전체 거래 건수 계산
4 WITH purchase_cnt AS (
5     SELECT
6         CustomerID,
7         COUNT(DISTINCT InvoiceNo) AS purchase_cnt
8     FROM `sixth-flag-466601-u1.modulabs_project.data`
9     GROUP BY CustomerID
10 ),
11
12 -- (2) 구매한 아이템 총 수량 계산
13 item_cnt AS (
14     SELECT
15         CustomerID,
16         SUM(Quantity) AS item_cnt
17     FROM `sixth-flag-466601-u1.modulabs_project.data`
18 )
19
20

```

쿼리 완료됨

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

제목 없는 쿼리

실행 시 이 쿼리가 9.22MB를 처리합니다.

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency
1	12346	2	0	325
2	12347	7	2458	2
3	12348	4	2332	75
4	12349	1	630	18
5	12350	1	196	310
6	12352	8	463	36
7	12353	1	20	204
8	12354	1	530	232
9	12355	1	240	214
10	12356	3	1573	22
11	12357	1	2708	33
12	12358	2	242	1
13	12359	6	1599	7
14	12360	3	1156	52
15	12361	1	90	287
16	12362	13	2180	3
17	12363	2	408	109
18	12364	4	1499	7
19	12365	1	173	291

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

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

```

[결과 이미지를 넣어주세요]

Q *제목 없는쿼리 X

제목 없는 쿼리 실행 다운로드 공유 일정 다음에서 열기 더보기

```

1 SELECT
2   CustomerID,
3   ROUND(SUM(Quantity * UnitPrice)) AS user_total
4 FROM sixth-flag-466601-u1.modulabs_project.data
5 GROUP BY CustomerID

```

실행 시 이 쿼리가 9.15MB를 처리합니다.

쿼리 결과 결과 저장

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.0
4	12349	1458.0
5	12350	294.0
6	12352	1265.0
7	12353	89.0
8	12354	1079.0
9	12355	459.0
10	12356	2487.0
11	12357	6208.0
12	12358	928.0

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_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,
  ut.user_total / rf.purchase_cnt AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice)) AS user_total
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

실행 다운로드 공유 일정

```

1 CREATE OR REPLACE TABLE sixth-flag-466601-u1.modulabs_project.user_rfm AS
2 SELECT
3   rf.CustomerID AS CustomerID,
4   rf.purchase_cnt,
5   rf.item_cnt,
6   rf.recency,
7   ut.user_total,
8   ut.user_total / rf.purchase_cnt AS user_average
9 FROM sixth-flag-466601-u1.modulabs_project.user_rf rf
10 LEFT JOIN (
11   -- 고객 별 총 지출액
12   SELECT
13     CustomerID,
14     ROUND(SUM(Quantity * UnitPrice)) AS user_total
15 FROM sixth-flag-466601-u1.modulabs_project.data
16 GROUP BY CustomerID
17 ) ut
18 ON rf.CustomerID = ut.CustomerID

```

쿼리 완료됨

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 삭제 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	12713	1	505	0	795.0	795.0		
2	13298	1	96	1	360.0	360.0		
3	14569	1	79	1	227.0	227.0		
4	15520	1	314	1	343.0	343.0		
5	13436	1	76	1	197.0	197.0		
6	15471	1	256	2	454.0	454.0		
7	14204	1	72	2	151.0	151.0		
8	15195	1	1404	2	3861.0	3861.0		
9	16569	1	93	3	124.0	124.0		
10	17914	1	457	3	329.0	329.0		
11	14578	1	240	3	169.0	169.0		
12	15992	1	17	3	42.0	42.0		
13	12478	1	233	3	546.0	546.0		
14	12650	1	250	3	242.0	242.0		
15	16528	1	171	3	244.0	244.0		
16	12442	1	181	3	144.0	144.0		
17	15318	1	642	3	313.0	313.0		
18	15097	1	170	4	248.0	248.0		
19	12367	1	172	4	151.0	151.0		
20	13790	1	748	4	349.0	349.0		
21	17383	1	148	4	193.0	193.0		
22	18015	1	157	4	120.0	120.0		
23	16597	1	184	4	90.0	90.0		

페이지당 결과 수: 50 1 ~ 50 (전체 4362행)

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

SELECT *
FROM sixth-flag-466601-u1.modulabs_project.user_rfm

```

[결과 이미지를 넣어주세요]

제목 없는 쿼리

쿼리 완료됨

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	13298	1	96	1	360.0	360.0
3	14569	1	79	1	227.0	227.0
4	15520	1	314	1	343.0	343.0
5	13436	1	76	1	197.0	197.0
6	15471	1	256	2	454.0	454.0
7	14204	1	72	2	151.0	151.0
8	15195	1	1404	2	3861.0	3861.0
9	16569	1	93	3	124.0	124.0
10	17914	1	457	3	329.0	329.0
11	14578	1	240	3	169.0	169.0
12	15992	1	17	3	42.0	42.0
13	12478	1	233	3	546.0	546.0
14	12650	1	250	3	242.0	242.0
15	16528	1	171	3	244.0	244.0
16	12442	1	181	3	144.0	144.0

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]

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12713	1	505	0	795.0	795.0	37
2	13298	1	96	1	360.0	360.0	2
3	14569	1	79	1	227.0	227.0	10
4	15520	1	314	1	343.0	343.0	18
5	13436	1	76	1	197.0	197.0	12
6	15471	1	256	2	454.0	454.0	67
7	14204	1	72	2	151.0	151.0	36
8	15195	1	1404	2	3861.0	3861.0	1
9	16569	1	93	3	124.0	124.0	5
10	17914	1	457	3	329.0	329.0	72
11	14578	1	240	3	169.0	169.0	24
12	15992	1	17	3	42.0	42.0	3
13	12478	1	233	3	546.0	546.0	35
14	12650	1	250	3	242.0	242.0	19
15	16528	1	171	3	244.0	244.0	17
16	12442	1	181	3	144.0	144.0	11
17	15318	1	642	3	313.0	313.0	33
18	15007	1	170	4	248.0	248.0	25

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]

user_data									
user_data									
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...	
51	15940	1	4	311	36.0	36.0	1	0.0	
52	13366	1	144	50	56.0	56.0	1	0.0	
53	18174	1	50	7	104.0	104.0	1	0.0	
54	16144	1	16	246	175.0	175.0	1	0.0	
55	17291	1	72	308	551.0	551.0	1	0.0	
56	18113	1	72	368	76.0	76.0	1	0.0	
57	16078	1	16	283	79.0	79.0	1	0.0	
58	15316	1	100	326	165.0	165.0	1	0.0	
59	16990	1	100	218	179.0	179.0	1	0.0	
60	18133	1	1350	212	931.0	931.0	1	0.0	
61	14119	1	-2	354	-20.0	-20.0	1	0.0	
62	17307	1	-144	365	-153.0	-153.0	1	0.0	
63	14705	1	100	198	179.0	179.0	1	0.0	
64	15668	1	72	217	76.0	76.0	1	0.0	
65	12943	1	-1	301	-4.0	-4.0	1	0.0	
66	16148	1	72	296	76.0	76.0	1	0.0	
67	18068	1	6	289	102.0	102.0	1	0.0	
68	16323	1	50	196	208.0	208.0	1	0.0	
69	13017	1	48	7	204.0	204.0	1	0.0	
70	16428	1	-1	81	-3.0	-3.0	1	0.0	
71	16257	1	1	176	22.0	22.0	1	0.0	
72	14576	1	12	372	35.0	35.0	1	0.0	
73	15070	1	36	372	106.0	106.0	1	0.0	

페이지당 결과 수: 50 51 ~ 100 (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 sixth-flag-466601-u1.modulabs_project.user_data AS
```

```
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNTIF(DISTINCT (InvoiceNo LIKE 'C%')) AS cancel_frequency
  FROM sixth-flag-466601-u1.modulabs_project.data
  GROUP BY CustomerID
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions, 2) cancel_rate
FROM `sixth-flag-466601-u1.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID
```

[결과 이미지를 넣어주세요]

user_data

스키마 세부정보 미리보기 테이블 탐색 통계 계보 데이터 프로파일 데이터 품질

명	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_prod	average_inter	total_transac	cancel_fre	cancel_rate
1	16990	1	100	218	179.0	179.0	1	0.0	1	0	0.0
2	16565	1	46	364	174.0	174.0	3	0.0	1	0	0.0
3	16387	1	44	322	94.0	94.0	4	0.0	1	0	0.0
4	15703	1	66	264	212.0	212.0	5	0.0	1	0	0.0
5	18281	1	54	180	81.0	81.0	7	0.0	1	0	0.0
6	13002	1	73	318	121.0	121.0	7	0.0	1	0	0.0
7	14059	1	128	266	184.0	184.0	8	0.0	1	0	0.0
8	17881	1	32	304	133.0	133.0	8	0.0	1	0	0.0
9	17245	1	75	204	171.0	171.0	9	0.0	1	0	0.0
10	14436	1	129	98	89.0	89.0	9	0.0	1	0	0.0
11	12618	1	95	21	137.0	137.0	10	0.0	1	0	0.0
12	17517	1	102	308	154.0	154.0	11	0.0	1	0	0.0
13	16641	1	256	105	231.0	231.0	12	0.0	1	0	0.0
14	15341	1	812	80	2021.0	2021.0	17	0.0	1	0	0.0
15	16281	1	934	72	2062.0	2062.0	17	0.0	1	0	0.0
16	13887	1	147	204	350.0	350.0	19	0.0	1	0	0.0
17	12452	1	181	16	349.0	349.0	20	0.0	1	0	0.0
18	12945	1	165	288	463.0	463.0	23	0.0	1	0	0.0
19	17263	1	36	208	63.0	63.0	23	0.0	1	0	0.0
20	15592	1	354	46	389.0	389.0	24	0.0	1	0	0.0
21	14873	1	168	210	520.0	520.0	28	0.0	1	0	0.0
22	12638	1	573	33	512.0	512.0	38	0.0	1	0	0.0

페이지당 결과 수: 50 1 - 50 (전체 4362명)

- 다양한 컬럼들을 활용하여 고객의 구매 패턴과 선호도를 보다 심층적으로 이해할 수 있도록 최종적으로 user_data를 출력하기

```
SELECT *
FROM sixth-flag-466601-u1.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

*제목 없는 쿼리

제목 없는 쿼리 실행 저장 온 다운로드 실패 공유 알림 다음에서 열기 더보기

```
1 SELECT *
2 FROM sixth-flag-466601-u1.modulabs_project.user_data
```

쿼리 완료됨

쿼리 결과 결과 저장 다음에서 열기

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

#	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16990	1	100	218	179.0	179.0	1	0.0	1	0	0.0
2	16565	1	46	364	174.0	174.0	3	0.0	1	0	0.0
3	16387	1	44	322	94.0	94.0	4	0.0	1	0	0.0
4	15703	1	66	264	212.0	212.0	5	0.0	1	0	0.0
5	18281	1	54	180	81.0	81.0	7	0.0	1	0	0.0
6	13002	1	73	318	121.0	121.0	7	0.0	1	0	0.0
7	14059	1	128	266	184.0	184.0	8	0.0	1	0	0.0
8	17881	1	32	304	133.0	133.0	8	0.0	1	0	0.0
9	17245	1	75	204	171.0	171.0	9	0.0	1	0	0.0
10	14436	1	129	98	89.0	89.0	9	0.0	1	0	0.0
11	12618	1	95	21	137.0	137.0	10	0.0	1	0	0.0
12	17517	1	102	308	154.0	154.0	11	0.0	1	0	0.0
13	16641	1	256	105	231.0	231.0	12	0.0	1	0	0.0
14	15341	1	812	80	2021.0	2021.0	17	0.0	1	0	0.0
15	16281	1	934	72	2062.0	2062.0	17	0.0	1	0	0.0
16	13887	1	147	204	350.0	350.0	19	0.0	1	0	0.0
17	12452	1	181	16	349.0	349.0	20	0.0	1	0	0.0
18	12945	1	165	288	463.0	463.0	23	0.0	1	0	0.0

페이지당 결과 수: 50 1 - 50 (전체 4362명) [<

데이터 해석 및 통찰

CustomerID의 경우 결측치 비율이 24.93%로 높은 수준임에도 노이즈가 될 수 있기에 다른 값으로 대체하는 대신 제거하는 것을 통해 분석 목적에 따라 불필요한 데이터는 과감하게 제거해야 한다는 것을 확인할 수 있었으며 데이터 전처리 중 오류값 처리에 가장 많은 시간과 노력이 든다는 것을 직접 깨달을 수 있었습니다.

작업 정보		결과	차트	JSON	실용 세부정보
행	country ▼		cnt ▼		
1	United Kingdom		356008		
2	Germany		9079		
3	France		8152		
4	EIRE		7368		
5	Spain		2462		
6	Netherlands		2326		
7	Belgium		1971		
8	Switzerland		1843		
9	Portugal		1427		
10	Australia		1253		
11	Norway		1059		
12	Italy		783		
13	Channel Islands		752		
14	Finland		653		
15	Cyprus		608		
16	Sweden		436		

쿼리 결과

작업 정보	결과	차트	JSON	실형 세부정보	실형 그래프
행	Description			cnt	
1	WHITE HANGING HEART T-LIGHT HOLDER			2058	
2	REGENCY CAKESTAND 3 TIER			1893	
3	JUMBO BAG RED RETROSPOT			1659	
4	PARTY BUNTING			1408	
5	ASSORTED COLOUR BIRD ORNAMENT			1405	
6	LUNCH BAG RED RETROSPOT			1345	
7	SET OF 3 CAKE TINS PANTRY DESIGN			1224	
8	LUNCH BAG BLACK SKULL			1099	
9	PACK OF 72 RETROSPOT CAKE CASES			1062	
10	SPOTTY BUNTING			1026	
11	PAPER CHAIN KIT 50'S CHRISTMAS			1013	
12	LUNCH BAG SPACEBOY DESIGN			1006	
13	LUNCH BAG CARS BLUE			1000	

 제목 없는 퀴리
 실행
 저장
 다운로드
 공유
 일정
 다음에서 열기
 더보기

쿼리 결과							결과 저장
결과							
작업 정보	자료	JSON	실행 세부정보	실행 그래프			
user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate		
-135.0	4	0.0	1	1	1.0		
-20.0	1	0.0	1	1	1.0		
-27.0	2	0.0	1	1	1.0		
-15.0	1	0.0	1	1	1.0		
-4.0	1	0.0	1	1	1.0		
-81.0	15	0.0	1	1	1.0		
-102.0	1	0.0	1	1	1.0		
-45.0	3	0.0	1	1	1.0		
-3.0	1	0.0	1	1	1.0		
-102.0	5	0.0	1	1	1.0		
-8.0	3	0.0	1	1	1.0		
-32.0	3	0.0	1	1	1.0		
-35.0	1	0.0	1	1	1.0		
-12.0	2	0.0	1	1	1.0		
-31.0	1	0.0	1	1	1.0		
-94.0	2	0.0	1	1	1.0		
--	-	--	-	-	--		

행	StockCode	Description	cancel_stock
1	22423	REGENCY CAKESTAND 3 TIER	180
2	22960	JAM MAKING SET WITH JARS	86
3	22720	SET OF 3 CAKE TINS PANTRY DESIGN	72
4	21232	STRAWBERRY CERAMIC TRINKET BOX	54
5	22699	ROSES REGENCY TEACUP AND SAUCER	53
6	22666	RECIPE BOX PANTRY YELLOW DESIGN	47
7	85099B	JUMBO BAG RED RETROSPOT	44
8	22697	GREEN REGENCY TEACUP AND SAUCER	42
9	20725	LUNCH BAG RED RETROSPOT	42
10	82483	WOOD 2 DRAWER CABINET WHITE FINISH	42
11	85123A	WHITE HANGING HEART T-LIGHT HOLDER	42
12	21843	RED RETROSPOT CAKE STAND	41
13	21314	SMALL GLASS HEART TRINKET POT	40
14	23245	SET OF 3 REGENCY CAKE TINS	37
15	22197	POPCORN HOLDER	36
16	22698	PINK REGENCY TEACUP AND SAUCER	35

페이지당 결과 수: 50 ▼

취소 수량이 많은 상품을 조사해보니, 판매량이 많았던 케이크틀, 가방 등이 보이는걸 봐서 많이 구매한 만큼 취소도 많이 되는 것을 확인할 수 있었고 리뷰나 취소 신청 이유 등을 참고하여 판매량에 비해 취소비율을 줄일 수 있는 방법을 고안해봐야 할 것 같습니다.

회고

Keep :

기존에 많이 연습하고, 사용했던 SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, LIMIT, WINDOW 함수 등의 기초 내용과 기본 사용 방법은 익숙해진 것 같습니다.

위 내용들만 안다고 해서 SQL의 모든 것을 아는 것은 아니기에 제가 더 편하게 구현하고자 했던 기능들이 따로 함수로 만들어져 있는지 검색해보며 새로운 내용을 익히는 능력도 키울 수 있었던 것 같습니다.

지금 알고 있는 내용을 반복 숙달하여 잊지 않도록 연습하는 것이 중요할 것 같습니다.

Problem :

SQL의 기본 문법들은 다 알고 있기에 단순히 이걸 적용만 하면 되지 않나 하고 생각했었습니다. 하지만 역시 도구의 사용 방법을 안다고 해도 이를 문제 해결에 어떻게 잘 적용할지는 또 다른 문제였던 것 같습니다.

특히 GROUP BY, 윈도우 함수 등을 응용하는 것에 있어서 좀 더 연습이 필요하다고 생각합니다.

또한, COUNTIF, REGEXP 등 실무에서 유용하게 쓰일 수 있는 함수들에 대한 학습이 필요할 것 같습니다.

Try :

지금처럼 본인이 알고 있는 내용에 대해서는 검색하지 않고 스스로 해결하려고 노력하려 하고, 구현하고자 하지만 함수 사용법을 까먹어 모르는 것에 대해서는 적절하게 검색하여 알아보려는 태도를 가지는 것이 좋을 것 같습니다.

또한, 위 프로젝트를 해결했다고 해서 넘어갈게 아니라 반복 학습을 통해 위 프로젝트 내용이 쉽게 느껴지고, 가이드 없이도 수월하게 해결할 수 있도록 체득하는 것이 중요할 것 같습니다.