

# 고객을 세그멘테이션하자 [프로젝트]

## 11-2. 데이터 불러오기

### 데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *
from modulabs_project.data
limit 10;
```

[결과 이미지를 넣어주세요]

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select *
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

### 데이터 수 세기

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

```
select
count(invoiceno) as count_invoiceNo,
count(stockcode) as count_stockcode,
count(description) as count_description,
count(quantity) as count_quantity,
count(InvoiceDate) as count_InvoiceDate,
count(UnitPrice) as count_UnitPrice,
count(CustomerID) as count_CustomerID,
count(Country) as count_Country
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

## 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

```
# 예시
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM 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_perc
```

```

FROM 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_pe
FROM 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_perce
FROM 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_pe
FROM 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_perce
FROM 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_per
FROM 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_perce

FROM modulabs_project.data;

```

[결과 이미지를 넣어주세요]

## 결측치 처리 전략

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

```

select description
from modulabs_project.data
where stockcode = '85123A';

```

[결과 이미지를 넣어주세요]

## 결측치 처리

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

```

delete from modulabs_project.data
where CustomerID is null or Description is null;

```

[결과 이미지를 넣어주세요]

## 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 InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, country as a
from modulabs_project.data
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, country
having count(*) > 1;
```

[결과 이미지를 넣어주세요]

### 중복값 처리

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

```
create or replace table modulabs_project.data as
select distinct * from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

## 11-6. 데이터 전처리(3): 오류값 처리

### InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
select distinct invoiceNo
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과		
작업 정보   결과   차트   JSON   실행 세부정보   실행		
행	invoiceNo	
1	574301	
2	C575531	
3	557305	
4	543008	
5	549735	
6	554032	
7	561387	
8	574868	
9	574827	
10	546015	
11	551859	
12	554665	
13	578187	

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
select distinct invoiceNo
from modulabs_project.data
limit 100;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

일	InvoiceNo
5	549735
6	554032
7	561387
8	574868
9	574827
10	546015
11	551859
12	554665
13	578187
14	569943
15	571241
16	574573
17	545419
18	554917
19	C558080
20	573418

페이지당 결과 수: 50 1 ~ 50 (전체 100행)

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

```
select *
from modulabs_project.data
where invoiceNo like 'C%'
limit 100;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

일	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	C575531	22960	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain	JAM MAKING SET WITH JARS
2	C558080	22840	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom	ROUND CAKE TIN VINTAGE RED
3	C558080	22847	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom	BREAD BIN DINER STYLE IVORY
4	C554983	47590A	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom	BLUE HAPPY BIRTHDAY BUNTL.
5	C554983	47590B	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom	PINK HAPPY BIRTHDAY BUNTL.
6	C539709	22822	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom	BROCCANTE SHELF WITH HOOKS
7	C539709	21485	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom	RETROSPOT HEART HOT WAT.
8	C539709	84978	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom	HANGING HEART JAR TIGHT...
9	C543020	21217	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom	RED RETROSPOT ROUND CAK.
10	C546858	21534	-1	2011-03-17 14:24:00 UTC	4.95	14081	United Kingdom	DAIRY MAID LARGE MILK JUG
11	C546858	22839	-1	2011-03-17 14:24:00 UTC	14.95	14081	United Kingdom	3 TIER CAKE TIN GREEN AND ...
12	C542263	22699	-1	2011-01-26 17:16:00 UTC	2.95	14849	United Kingdom	ROSES REGENCY TEACUP AN...
13	C553534	21467	-1	2011-05-17 15:15:00 UTC	3.75	14849	United Kingdom	CHERRY CROCHET FOOD COV...
14	C570996	22909	-12	2011-10-13 12:02:00 UTC	0.85	14849	United Kingdom	SET OF 20 VINTAGE CHRISTM...
15	C570996	23376	-24	2011-10-13 12:02:00 UTC	0.39	14849	United Kingdom	PACK OF 12 VINTAGE CHRIST...
16	C543818	22138	-2	2011-02-13 15:45:00 UTC	4.95	16897	United Kingdom	BAKING SET 9 PIECE RETROSP...

페이지당 결과 수: 50 1 ~ 50 (전체 100행)

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

```
select round(sum(case when InvoiceNo like 'C%' then 1 else 0 end) * 100/ count(*),1) as canceled_per
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

일	canceled_per
1	2.1

## StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
select distinct StockCode
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과		
작업 정보 <b>결과</b> 차트   JSON   실행 세부정보   실행 그래프		
행	StockCode ▾	
1	22960	
2	23511	
3	85049A	
4	22144	
5	23240	
6	22750	
7	20749	
8	22734	
9	23514	
10	22086	
11	20971	

- 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 1
order by sell_cnt desc
limit 10;
```

[결과 이미지를 넣어주세요]

- 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 modulabs_project.data
)
WHERE number_count <= 1;
```

[결과 이미지를 넣어주세요]

- 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 LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) <= 1
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
```

```

SELECT StockCode,
       LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
FROM modulabs_project.data
) AS sub
WHERE number_count <= 1
);

```

[결과 이미지를 넣어주세요]

## Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```

SELECT Description, COUNT(*) AS description_cnt
FROM modulabs_project.data
group by 1
order by description_cnt desc
limit 30;

```

[결과 이미지를 넣어주세요]

- 서비스 관련 정보를 포함하는 행들을 제거하기

```

delete
from modulabs_project.data
where Description IN (
    SELECT DISTINCT Description
    FROM modulabs_project.data
    WHERE REGEXP_CONTAINS(Description, '[a-z]')
);

```

[결과 이미지를 넣어주세요]

- 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]

## UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```

select min(unitprice) as min_price, max(unitprice) as max_price, avg(unitprice) as avg_price
from modulabs_project.data;

```

[결과 이미지를 넣어주세요]

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

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

[결과 이미지를 넣어주세요]

- **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]

## 11-7. RFM 스코어

### Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
select date(invoicedate) as invoiceday, *
from modulabs_project.data;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과 결과 저장 데이터 탐색 ↕

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	invoiceday	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-11-03	574301	23512	6	2011-11-03 16:15:00 UTC
2	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC
3	2011-11-03	574301	23240	6	2011-11-03 16:15:00 UTC
4	2011-11-03	574301	85049A	12	2011-11-03 16:15:00 UTC
5	2011-11-03	574301	22734	6	2011-11-03 16:15:00 UTC
6	2011-11-03	574301	85049E	12	2011-11-03 16:15:00 UTC
7	2011-11-03	574301	22750	4	2011-11-03 16:15:00 UTC
8	2011-11-03	574301	23511	6	2011-11-03 16:15:00 UTC
9	2011-11-03	574301	22144	6	2011-11-03 16:15:00 UTC
10	2011-11-03	574301	22960	6	2011-11-03 16:15:00 UTC
11	2011-11-03	574301	20971	12	2011-11-03 16:15:00 UTC
12	2011-11-03	574301	22910	6	2011-11-03 16:15:00 UTC
13	2011-11-03	574301	22077	12	2011-11-03 16:15:00 UTC
14	2011-11-03	574301	84879	8	2011-11-03 16:15:00 UTC
15	2011-11-03	574301	23514	6	2011-11-03 16:15:00 UTC

페이지당 결과 수: 50 1 - 50 (전체 398277행) |< < > >|

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

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

[결과 이미지를 넣어주세요]

← 쿼리 결과

결과

자표

JSON

실행 세부정보

실행 그래프

항	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-12-09 12:50:00 UTC	2011-03-02	545419	467750	1	2011-03-02 14:16:00 UTC	16.95	15104	United Kingdom	SUNSET COLOUR CHUNKY KN...
2	2011-12-09 12:50:00 UTC	2011-09-14	566674	23318	6	2011-09-14 11:41:00 UTC	2.49	15105	United Kingdom	BOX OF 6 MINI VINTAGE CRAC...
3	2011-12-09 12:50:00 UTC	2011-11-22	577814	22119	3	2011-11-22 09:16:00 UTC	6.95	15106	United Kingdom	PEACE WOODEN BLOCK LEFT...
4	2011-12-09 12:50:00 UTC	2011-04-04	C548723	85063	-1	2011-04-04 09:02:00 UTC	16.95	15874	United Kingdom	CREAM SWEETHEART MAGAZI...
5	2011-12-09 12:50:00 UTC	2011-04-08	549317	23010	1	2011-04-08 09:58:00 UTC	16.95	15109	United Kingdom	CIRCUS PARADE BABY GIFT SET
6	2011-12-09 12:50:00 UTC	2011-11-22	577855	23118	1	2011-11-22 11:16:00 UTC	7.5	15877	United Kingdom	PARISIENNE JEWELLERY PRA...
7	2011-12-09 12:50:00 UTC	2011-04-21	550823	22653	1	2011-04-21 09:36:00 UTC	39.95	16133	United Kingdom	IVORY EMBROIDERED QUILT
8	2011-12-09 12:50:00 UTC	2011-05-19	553742	22448	6	2011-05-19 09:38:00 UTC	3.35	16133	United Kingdom	PINK CUSHION BABUSHKA RED
9	2011-12-09 12:50:00 UTC	2011-08-11	562966	20971	144	2011-08-11 10:53:00 UTC	1.06	16133	United Kingdom	PINK BLUE FELT CRAFT TRINK...
10	2011-12-09 12:50:00 UTC	2011-05-13	553144	23200	100	2011-05-13 13:29:00 UTC	1.79	17669	United Kingdom	JUMBO BAG PEARS
11	2011-12-09 12:50:00 UTC	2011-09-25	566202	21843	1	2011-09-25 15:01:00 UTC	10.95	15366	United Kingdom	RED RETROSPOT CAKE STAND
12	2011-12-09 12:50:00 UTC	2011-06-02	555400	23091	21	2011-06-02 17:50:00 UTC	5.39	14088	United Kingdom	ZINC HERB GARDEN CONTAIN...
13	2011-12-09 12:50:00 UTC	2011-11-23	578305	23539	8	2011-11-23 15:44:00 UTC	5.35	14088	United Kingdom	WALL ART LOVES SECRET
14	2011-12-09 12:50:00 UTC	2011-06-30	556535	84762	1	2011-06-30 12:04:00 UTC	4.65	16904	United Kingdom	ENCHANTED BIRD COATHANG...
15	2011-12-09 12:50:00 UTC	2011-01-19	541525	94946	72	2011-01-19 10:51:00 UTC	1.06	14090	United Kingdom	ANTIQUE SILVER TEA GLASS E...
16	2011-12-09 12:50:00 UTC	2011-11-25	576881	21317	2	2011-11-25 08:41:00 UTC	5.45	13579	United Kingdom	GLASS SPHERE CANDLE STAN...

쿼리지달 결과 수: 50 1 ~ 50 (전체 398277행)

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
select
  customerID,
  max(invoicedate) as InvoiceDay,
from modulabs_project.data
group by customerID;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

결과

자표

JSON

실행 세부정보

실행 그래프

항	customerID	InvoiceDay
1	12544	2011-11-10 11:12:00 UTC
2	13568	2011-06-19 14:42:00 UTC
3	13824	2011-11-07 12:41:00 UTC
4	14080	2011-11-07 11:09:00 UTC
5	14336	2011-11-23 11:40:00 UTC
6	14592	2011-11-04 16:35:00 UTC
7	15104	2011-09-26 11:35:00 UTC
8	15360	2011-10-31 09:35:00 UTC
9	15872	2011-11-25 11:55:00 UTC
10	16128	2011-11-22 11:14:00 UTC
11	16384	2011-09-11 11:19:00 UTC
12	17152	2011-05-29 12:19:00 UTC
13	17408	2011-06-29 12:53:00 UTC
14	17664	2011-11-21 11:29:00 UTC
15	17920	2011-12-05 14:29:00 UTC
16	18176	2010-12-21 12:30:00 UTC

쿼리지달 결과 수: 50 1 ~ 50 (전체 4361행)

- 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 modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

결과

자표

JSON

실행 세부정보

실행 그래프

항	CustomerID	recency
1	14031	267
2	14650	5
3	14854	5
4	17175	5
5	17965	156
6	17460	147
7	15992	70
8	16021	22
9	15669	40
10	17107	9
11	15515	65
12	13026	23
13	13437	10
14	17208	2
15	15107	8
16	16016	22

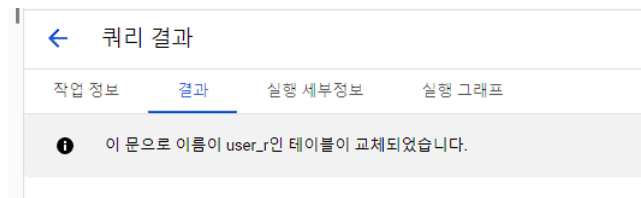
쿼리지달 결과 수: 50 1 ~ 50 (전체 4361행)

- 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]



## Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
select
  customerID,
  count(distinct invoiceNo) as purchase_cnt
from 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			
9	15872	2			
10	16128	5			
11	16384	2			

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM 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	199			
9	15872	187			
10	16128	976			
11	16384	260			
12	17152	477			
13	17408	3			
14	17664	604			
15	17920	2466			
16	18176	279			

- 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(distinct invoiceNo) as purchase_cnt
  from modulabs_project.data
  group by customerID
),
```

```
-- (2) 구매한 아이템 총 수량 계산
```

```
item_cnt AS (
  SELECT
    CustomerID,
    SUM(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 <b>결과</b> 실행 세부정보   실행 그래프
<b>i</b> 이 문으로 이름이 user_rf인 테이블이 교체되었습니다.

## Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  customerID,
  round(sum(quantity * unitprice), 1) AS user_total
FROM 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	299.7
2	13568	187.0
3	13824	1698.9
4	14080	45.6
5	14336	1614.9
6	14592	557.9
7	15104	968.6
8	15360	417.9
9	15872	316.2
10	16128	1855.0
11	16384	584.5
12	17152	1503.5
13	17408	32.6
14	17664	604.6

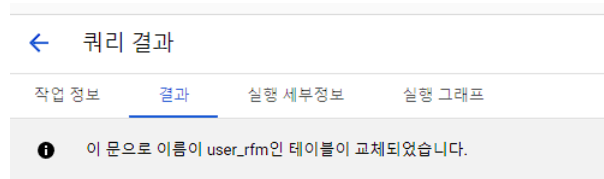
- 고객별 평균 거래 금액 계산

- 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 modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  round(ut.user_total / nullif(rf.purchase_cnt, 0), 1) AS user_average
FROM modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    customerID,
    sum(quantity * unitprice) AS user_total
```

```
FROM modulabs_project.data
group by customerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



## RFM 통합 테이블 출력하기

- 최종 user\_rfm 테이블을 출력하기

```
-- 최종 출력 (왜 4362명인데 1명이 덜 나왔지..?)
select *
from modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

← 쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 user\_rfm인 테이블이 교체되었습니다.

항	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5000000000...	794.5
2	15083	1	38	256	88.2	88.2
3	18010	1	60	256	174.8	174.8
4	12792	1	215	256	344.5000000000...	344.5
5	14059	1	79	1	227.8000000000...	227.8
6	13298	1	96	1	360.0	360.0
7	13426	1	76	1	196.89	196.9
8	15520	1	314	1	343.4000000000...	343.5
9	13357	1	318	257	603.1	603.1
10	14476	1	110	257	192.9000000000...	193.0
11	14204	1	72	2	150.6100000000...	150.6
12	15195	1	1404	2	3861.0	3861.0
13	15471	1	256	2	454.4700000000...	454.5
14	15992	1	17	3	41.0800000000...	42.0
15	16569	1	93	3	124.2	124.2
16	12478	1	233	3	545.99	546.0

페이지당 결과 수: 50 1 ~ 50 (현재 4361명)

## 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]



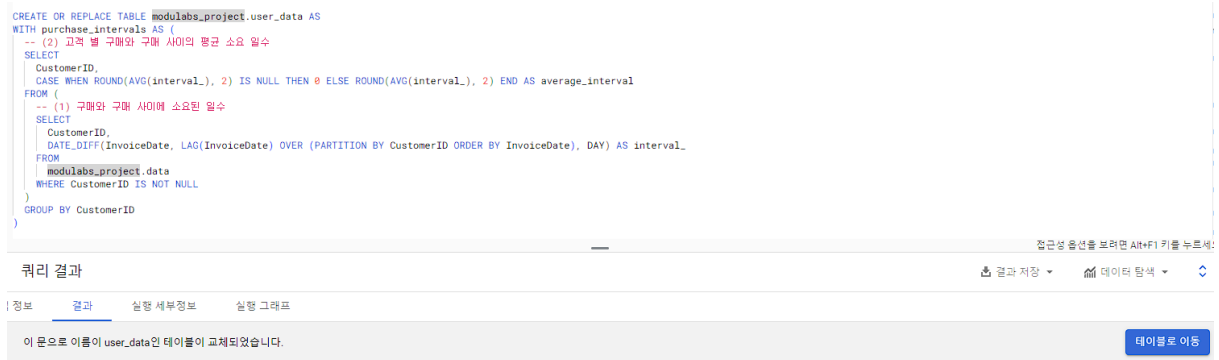
## 2. 평균 구매 주기

- 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]



### 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)

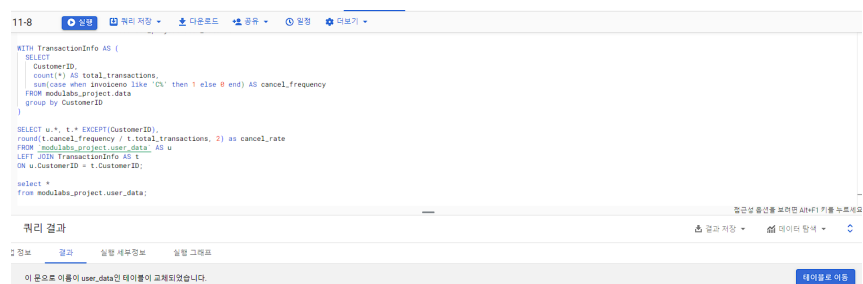
```
CREATE OR REPLACE TABLE modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(*) AS total_transactions,
    sum(case when invoiceno like 'C%' then 1 else 0 end) 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;

select *
from modulabs_project.user_data;
```

[결과 이미지를 넣어주세요]



- 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]

11-8

실행

쿼리 저장

다운로드

공유

설정

다보기

```
WITH TransactionInfo AS (
    SELECT
        CustomerID,
        count(*) AS total_transactions,
        sum(case when InvoiceNo like 'C%' then 1 else 0 end) 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;

select *
from modulabs_project.user_data;
```

결과 생성을 보려면 Athena

결과 저장

데이터 탐색

쿼리 결과

결과

자본

JSON

실시간 세부정보

실시간 그래프

	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	12428	11	3477	25	6366.0000000000...	578.7	256	0.87	292	5	0.02
2	13366	1	144	50	56.160000000000...	56.2	1	0.0	1	0	0.0
3	15118	1	1440	134	244.8	244.8	1	0.0	1	0	0.0
4	13135	1	4300	196	3096.0	3096.0	1	0.0	1	0	0.0
5	16079	1	-12	265	36.599999999999...	-30.6	1	0.0	1	1	1.0
6	16344	1	18	150	101.6000000000...	101.1	1	0.0	2	0	0.0
7	16148	1	72	296	76.320000000000...	76.3	1	0.0	1	0	0.0
8	17291	1	72	308	550.8000000000...	550.8	1	0.0	1	0	0.0
9	16428	1	-1	81	-2.95	-3.0	1	0.0	1	1	1.0