

11 - 미니프로젝트 - 고객세그멘테이션

<https://www.kaggle.com/datasets/carrie1/ecommerce-data>

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select * from poetic-glass-456004-g2.modulabs_project.data limit 10;
```

Query results

Save results

Open in

Job information		Results	Chart	JSON	Execution details	Execution graph	
Row	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitP	
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC		
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC		
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC		
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC		
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC		
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC		
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC		
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC		
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC		
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC		

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*) from poetic-glass-456004-g2.modulabs_project.data;
```

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	total_row
1	541909

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 poetic-glass-456004-g2.modulabs_project.data;
```

Query results

Save results

Open in

Job information		Results	Chart	JSON	Execution details	Execution graph		
Row	unt_InvoiceNo	count_StockCode	count_Description	count_Quantity	count_InvoiceDate	count_UnitPrice	count_CustomerID	count_Country
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

```
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'StockCode' AS column_name,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'Description' AS column_name,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'Quantity' AS column_name,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'UnitPrice' AS column_name,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.modulabs_project.data
union all
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM poetic-glass-456004-g2.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 poetic-glass-456004-g2.modulabs_project.data;
```

Query results [Save results](#) [Open in](#)

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	column_name	missing_percentage			
1	StockCode	0.0			
2	Country	0.0			
3	Description	0.27			
4	Quantity	0.0			
5	CustomerID	24.93			
6	UnitPrice	0.0			
7	InvoiceNo	0.0			

결측치 처리 전략

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

```
select distinct Description
from poetic-glass-456004-g2.modulabs_project.data
where StockCode = '85123A';
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	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 poetic-glass-456004-g2.modulabs_project.data
where Description is null or CustomerID is null;
```

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>This statement removed 135,080 rows from data. Go to table</p>			

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(*) duplicate from poetic-glass-456004-g2.modulabs_project.data
group by InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
having count(*) > 1;
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	duplicate				
1	2				
2	2				
3	2				
4	2				
5	3				
6	2				
7	2				
8	2				
9	2				

Results per page: 50 1 - 50 of 4837

중복값 처리

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

create or replace table poetic-glass-456004-g2.modulabs_project.data
as select distinct * from poetic-glass-456004-g2.modulabs_project.data;

Query results		Save results	Open in	
Job information	Results	Execution details	Execution graph	
<div> This statement replaced the table named data. </div>		Go to table		

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
select count(distinct InvoiceNo) as count_unique_invNo from poetic-glass-456004-g2.modulabs_project.data;
```

Query results		Save results	Open in	
Job information	Results	Chart	JSON	Execution details
Execution graph				
Row	count_unique_invNo			
1	22190			

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
-- 고유 invoice list 100
select distinct InvoiceNo
from poetic-glass-456004-g2.modulabs_project.data
order by InvoiceNo
limit 100;
```

Query results		Save results	Open in	
Job information	Results	Chart	JSON	Execution details
Execution graph				
Row	InvoiceNo			
1	536365			
2	536366			
3	536367			
4	536368			
5	536369			
6	536370			
7	536371			
8	536372			
9	536373			

Results per page: 50 1 - 50 of 100

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

```
select * from poetic-glass-456004-g2.modulabs_project.data
where InvoiceNo like 'C%' or InvoiceNo like 'c%'
limit 100;
```

Query results

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-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	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC
7	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC
8	C547388	84050	PINK HEART SHAPE EGG FRYL...	-12	2011-03-22 16:07:00 UTC

Results per page: 50 1 - 50 of 100

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

```
select round(sum(case when InvoiceNo like 'C%' then 1 when InvoiceNo like 'c%' then 1 else 0 end) * 100 / count(*), 1) as CAX_
from poetic-glass-456004-g2.modulabs_project.data;
```

Query results

Save results

Open in

Job informationResultsChartJSONExecution detailsExecution graph

Row	CAX_rate
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
select count(distinct StockCode) uniqueStockCode from poetic-glass-456004-g2.modulabs_project.data;
```

Query results

Save results

Open

Job informationResultsChartJSONExecution detailsExecution graph

Row	uniqueStockCode
1	3684

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 **StockCode** 별 등장 빈도를 출력하기

- 상위 10개의 제품들을 출력하기

```
select StockCode, count(*) sell_cnt
from poetic-glass-456004-g2.modulabs_project.data
group by StockCode
order by sell_cnt desc
limit 10;
```

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	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	22107	1110

Results per page: 501 - 10 of 10<<>>

- 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 poetic-glass-456004-g2.modulabs_project.data
)
where number_count in (0,1);
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	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			

Results per page: 50 1 - 8 of 8

- **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(cast(countif(number_count in (0,1)) as int64)/ count(*) * 100 , 2) rate
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM poetic-glass-456004-g2.modulabs_project.data);
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	rate				
1	0.48				

Results per page: 50 1 - 1 of 1

- 제품과 관련되지 않은 거래 기록을 제거하기

```
delete from poetic-glass-456004-g2.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 poetic-glass-456004-g2.modulabs_project.data
  )
  where number_count in (0,1)
);
```

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>i This statement removed 1,915 rows from data. Go to table</p>			

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
select Description,count(*) cnt from poetic-glass-456004-g2.modulabs_project.data
group by Description
order by cnt desc
limit 30;
```

Job information				Results	Chart	JSON	Execution details	Execution graph
Row	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						

Results per page: 50 1 - 30 of 30 |< > |

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
delete from poetic-glass-456004-g2.modulabs_project.data
where Description like '%High Resolution Image%' or Description like '%Next Day Carriage%';
```

Job information				Results	Execution details	Execution graph
This statement removed 83 rows from data.						

Go to table

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs_project.data AS
SELECT * EXCEPT (Description),
upper(Description) Description
FROM poetic-glass-456004-g2.modulabs_project.data;
```

Job information				Results	Execution details	Execution graph
This statement replaced the table named data.						

Go to table

UnitPrice 살펴보기

- UnitPrice 의 최솟값, 최댓값, 평균을 구하기

```
select min(UnitPrice) min, max(UnitPrice) max, avg(UnitPrice) avg
from poetic-glass-456004-g2.modulabs_project.data;
```

Job information				Results	Chart	JSON	Execution details	Execution graph
Row	min	max	avg					
1	0.0	649.5	2.904956757406...					

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

```
select count(InvoiceNo) cnt, min(Quantity) Qmin, max(Quantity) Qmax, avg(Quantity) Qavg
from poetic-glass-456004-g2.modulabs_project.data
where UnitPrice = 0;
```

Query results [Save results](#) [Open in](#)

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	cnt	Qmin	Qmax	Qavg	
1	33	1	12540	420.5151515151...	

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs_project.data AS
SELECT *
FROM poetic-glass-456004-g2.modulabs_project.data
WHERE UnitPrice != 0;
```

Query results [Save results](#) [Open in](#)

Job information	Results	Execution details	Execution graph
This statement replaced the table named data.			

[Go to table](#)

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
select date(InvoiceDate) as InvoiceDay, *
from poetic-glass-456004-g2.modulabs_project.data;
```

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.0
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.0
3	2010-12-07	537626	851678	30	2010-12-07 14:57:00 UTC	1.2
4	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.2
5	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.2

Results per page:

50

1 - 50 of 399573

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
select date(max(InvoiceDate)) most_recent_date
from poetic-glass-456004-g2.modulabs_project.data;
```

Query results [Save results](#) [Open in](#)

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	most_recent_date				
1	2011-12-09				

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기


```
select CustomerID, Date(Max(invoiceDate)) as InvoiceDay
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID;
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	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			

Results per page: 50 1 - 50 of 4362

- 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 poetic-glass-456004-g2.modulabs_project.data
  GROUP BY CustomerID
);
```

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	CustomerID	recency			
1	12479	3			
2	12527	81			
3	12666	359			
4	12669	150			
5	12931	21			
6	13300	164			

Results per page: 50 1 - 50 of 4362

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

```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.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 poetic-glass-456004-g2.modulabs_project.data
  GROUP BY CustomerID
);
```

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>This statement created a new table named user_r.</p> <p>Go to table</p>			

225 select * from poetic-glass-456004-g2.modulabs_project.user_rf;
226

Query results

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	recency
1	16446	0
2	12423	0
3	17581	0
4	13069	0
5	15344	0
6	12662	0

Results per page: 50 1 - 50 of 4362

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
select CustomerID, count(InvoiceNo) as purchase_cnt
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID;
```

Query results

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84

Results per page: 50 1 - 50 of 4362

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
select CustomerID, count(InvoiceNo) as purchase_cnt
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID;
```

Query results

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	purchase_cnt
1	12346	2
2	12347	182
3	12348	27
4	12349	72
5	12350	16
6	12352	84

Results per page: 50 1 - 50 of 4362

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

```
CREATE OR REPLACE TABLE poetic-glass-456004-g2.modulabs_project.user_rf AS
WITH purchase_cnt AS (
  select CustomerID, count(InvoiceNo) as purchase_cnt
  from poetic-glass-456004-g2.modulabs_project.data
  group by CustomerID), -- (1) 전체 거래 건수 계산
item_cnt AS (
  select CustomerID, sum(Quantity) item_cnt
  from poetic-glass-456004-g2.modulabs_project.data
  group by CustomerID) -- (2) 구매한 아이템 총 수량 계산
SELECT
  -- 기존의 user_r에 (1)과 (2)를 통합
  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 poetic-glass-456004-g2.modulabs_project.user_r AS ur
ON pc.CustomerID = ur.CustomerID;
```

Query results

Job information Results Execution details Execution graph

This statement created a new table named user_rf. [Go to table](#)

266 `select * from poetic-glass-456004-g2.modulabs_project.user_rf;` Press Option+F1 for Accessibility Options.

Query results

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	purchase_cnt	item_cnt	recency
1	15910	266	1013	0
2	17315	482	3805	0
3	12423	118	1312	0
4	14422	222	2906	0
5	12526	68	624	0
6	17754	90	1767	0

Results per page: 50 1 - 50 of 4362

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
select CustomerID , round(SUM(UnitPrice * Quantity),1) user_total
from poetic-glass-456004-g2.modulabs_project.data
group by CustomerID;
```

Query results

Job information Results Chart JSON Execution details Execution graph

Row	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

Results per page: 50 1 - 50 of 4362

- 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 poetic-glass-456004-g2.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 / rf.purchase_cnt,1) user_average
FROM poetic-glass-456004-g2.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT CustomerID , round(SUM(UnitPrice * Quantity),1) user_total
  from poetic-glass-456004-g2.modulabs_project.data
```

```
group by CustomerID ) ut
ON rf.CustomerID = ut.CustomerID;
```

Query results

Save results Open in

Job information Results Execution details Execution graph

This statement created a new table named user_rfm. Go to table

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
select * from poetic-glass-456004-g2.modulabs_project.user_rfm;
```

282 select * from poetic-glass-456004-g2.modulabs_project.user_rfm

Press Option+F1 for Accessibility Options.

Query results

Save results Open in

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	16705	284	5458	0	13946.1	49.1
2	17581	451	5849	0	10716.3	23.8
3	12433	420	11071	0	13375.9	31.8
4	17754	90	1767	0	1632.3	18.1
5	15804	273	2513	0	3848.5	14.1
6	14051	214	3740	0	15462.3	72.3

Results per page: 50 1 - 50 of 4362

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 poetic-glass-456004-g2.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM poetic-glass-456004-g2.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM poetic-glass-456004-g2.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

Query results

Save results Open in

Job information Results Execution details Execution graph


This statement created a new table named user_data. Go to table

302 `select * from poetic-glass-456004-g2.modulabs_project.user_data;`

303

Press Option+F1 for Accessibility Options.

Query results

[Save results](#) [Open in](#) 

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	17763	1	12	263	15.0	15.0	1
2	17331	1	16	123	175.2	175.2	1
3	13391	1	4	203	59.8	59.8	1
4	17291	1	72	308	550.8	550.8	1
5	17925	1	72	372	244.1	244.1	1
6	16953	1	10	30	20.8	20.8	1

Results per page: 50 1 – 50 of 4362

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 poetic-glass-456004-g2.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
      poetic-glass-456004-g2.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM poetic-glass-456004-g2.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

Query results

Save results

Open in

Job information

Results

Execution details

Execution graph

This statement replaced the table named user_data.

Go to table

342

select * from poetic-glass-456004-g2.modulabs_project.user_data;

Press Option+F1 for Accessibility

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

low	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	13747	1	8	373	79.6	79.6	1	0.0
2	17347	1	216	86	229.0	229.0	1	0.0
3	13703	1	10	318	99.5	99.5	1	0.0
4	15657	1	24	22	30.0	30.0	1	0.0
5	12814	1	48	101	85.9	85.9	1	0.0
6	16995	1	-1	372	-1.3	-1.3	1	0.0

Results per page:

50

1 - 50 of 4362

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 poetic-glass-456004-g2.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(InvoiceNo) AS total_transactions,
    countif(InvoiceNo like 'C%' or InvoiceNo like 'c%' ) AS cancel_frequency
  FROM poetic-glass-456004-g2.modulabs_project.data
  group by CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round(cancel_frequency/total_transactions* 100,2) AS cancel_rate
FROM poetic-glass-456004-g2.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

Query results

Save results Open in

Job information Results Execution details Execution graph

This statement replaced the table named user_data. Go to table

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

```
select * from poetic-glass-456004-g2.modulabs_project.user_data;
```

Query results

Save results Open in

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	17715	1	384	200	326.4	326.4	1	0.0	1	0	0.0
2	18174	1	50	7	104.0	104.0	1	0.0	1	0	0.0
3	17925	1	72	372	244.1	244.1	1	0.0	1	0	0.0
4	16737	1	288	53	417.6	417.6	1	0.0	1	0	0.0
5	17382	1	24	65	50.4	50.4	1	0.0	1	0	0.0
6	13135	1	4300	196	3096.0	3096.0	1	0.0	1	0	0.0

Results per page: 50 1 - 50 of 4362

회고

[회고 내용을 작성해주세요]

Keep : 스스로 생각해서 문제를 해결하려고 했다.

Problem : 자주 나오는 것임에도 코드를 잘 외우고 있지는 않은 거 같다 (근데, 모든 코드를 암기하는게 효율적인가? 의구심)

Try : 프로그래밍을 하다보면 활용해야 하는 요구하는 사항이 비슷한데, 자주 쓰이는 코드는 오피시디언 같은 걸로 정리해서 각 프로그래밍별로 비교 가능하게 정리를 해 뒤서 빨리 찾아 쓸 수 있게 해야겠다...고 계속 생각하고 있는데 못하고 있네요;;;