

고객을 세그멘테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *
FROM modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	536414	22139	null	56	2010-12-01 11:52:00 UTC
2	536545	21134	null	1	2010-12-01 14:32:00 UTC
3	536546	22145	null	1	2010-12-01 14:33:00 UTC
4	536547	37509	null	1	2010-12-01 14:33:00 UTC
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC
6	536550	85044	null	1	2010-12-01 14:34:00 UTC
7	536552	20950	null	1	2010-12-01 14:34:00 UTC
8	536553	37461	null	3	2010-12-01 14:35:00 UTC
9	536554	84670	null	23	2010-12-01 14:35:00 UTC
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행
행	f0_			
1	541909			

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_InvoiceData,
COUNT(UnitPrice) AS COUNT_UnitPrice,
COUNT(CustomerID) AS COUNT_CustomerID,
COUNT(Country) AS COUNT_Country
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과									
작업 정보		결과	자트	JSON	실행 세부정보	실행 그래프			
행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceData	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country	
1	541909	541909	540455	541909	541909	541909	406829	541909	

5-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```

SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM modulabs_project.data

UNION ALL

SELECT
    'StockCode' AS column_name,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM modulabs_project.data

UNION ALL

SELECT
    'Description' AS column_name,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM modulabs_project.data

UNION ALL

SELECT
    'Quantity' AS column_name,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percenta
FROM modulabs_project.data

UNION ALL

SELECT
    'InvoiceDate' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM modulabs_project.data

UNION ALL

SELECT
    'UnitPrice' AS column_name,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM modulabs_project.data

UNION ALL

SELECT
    'CustomerID' AS column_name,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM modulabs_project.data

UNION ALL

SELECT
    'Country' AS column_name,

```

```
ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보
행	column_name		missing_percentage	
1	Country		0.0	
2	StockCode		0.0	
3	InvoiceNo		0.0	
4	Description		0.27	
5	InvoiceDate		0.0	
6	CustomerID		24.93	
7	UnitPrice		0.0	
8	Quantity		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]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보
행	Description			
1	?			
2	wrongly marked carton 22804			
3	CREAM HANGING HEART T-LIGHT HOLDER			
4	WHITE HANGING HEART T-LIGHT HOLDER			

결측치 처리

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

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

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 data의 행 135,080개가 삭제되었습니다.			

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

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

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	row				
1	4837				

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]

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data인 테이블이 교체되었습니다.			

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	row				
1	401604				

5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	row				
1	22190				

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo				
1	574301				
2	C575531				
3	557305				
4	543008				
5	549735				
6	554032				
7	561387				
8	574868				
9	574827				
10	546015				
11	551859				

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

결과 저장 다음에서 열기

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC
2	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC
3	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC
6	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC
7	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC
8	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC
9	C543620	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC
10	C546858	21534	DAIRY MAID LARGE MILK JUG	-1	2011-03-17 14:24:00 UTC

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

```
SELECT ROUND(SUM(CASE WHEN SUBSTR(InvoiceNo, 1, 1) = 'C' THEN 1 ELSE 0 END) / COUNT(*) * 100, 1) AS
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	CancelRatio				
1	2.2				

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	자트	JSON	실행 세부정보	실행 그래프
행	Count				
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 modulabs_project.data
GROUP BY StockCode
```

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

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

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

페이지당 결과 수: 50

1 ~ 10 (전체 10행)

<

<

>

>

- **StockCode**의 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM project_name.modulabs_project.data
)

SELECT
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
  COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보
행	number_count	stock_cnt		
1	5	3676		
2	0	7		
3	1	1		

- **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 IN (0, 1);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	StockCode	number_count
1	POST	0
2	M	0
3	PADS	0
4	D	0
5	BANK CHARGES	0
6	DOT	0
7	CRUK	0
8	C2	1

- **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)

```
-- version 1

SELECT ROUND(SUM(CASE WHEN StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRU
FROM modulabs_project.data;

-- version 2
SELECT ROUND(SUM(CASE WHEN StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
      LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM modulabs_project.data
  )
  WHERE number_count IN (0, 1))
  THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS StockCodeBiasRatio
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	StockCodeBiasRatio
1	0.48

- 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, r'[0-9]', '')) AS number_count
    FROM modulabs_project.data
  )
  WHERE number_count IN (0, 1)
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

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

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

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

페이지당 결과 수: 501 - 30 (전체 30행) |<<>>|

- 대소문자가 혼합된 Description이 있는지 확인하기

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	Description
1	BAG 125g SWIRLY MARBLES
2	3 TRADITIONAL BISCUIT CUTTERS SET
3	BAG 250g SWIRLY MARBLES
4	ESSENTIAL BALM 3.5g TIN IN ENVELOPE
5	FOLK ART GREETING CARD,pack/12
6	BAG 500g SWIRLY MARBLES
7	POLYESTER FILLER PAD 45x45cm
8	POLYESTER FILLER PAD 40x40cm
9	Next Day Carriage

페이지당 결과 수:

50

1 - 19 (전체 19행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

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

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 다음에서 열기

작업 정보 | **결과** | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 83개가 선택되었습니다. [테이블로 이동](#)

- 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]

쿼리 결과

결과 저장 | 다음에서 열기

작업 정보 | **결과** | 실행 세부정보 | 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다. [테이블로 이동](#)

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, ROUND(AVG(UnitPrice), 3) AS avg_pri
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 | 다음에서 열기

작업 정보 | **결과** | 차트 | JSON | 실행 세부정보 | 실행 그래프

행	min_price	max_price	avg_price
1	0.0	649.5	2.905

- 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]

쿼리 결과

결과 저장 | 다음에서 열기

작업 정보 | **결과** | 차트 | JSON | 실행 세부정보 | 실행 그래프

행	cnt_quantity	min_quantity	max_quantity
1	33	1	12540

- UnitPrice = 0를 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		결과 저장	다음에서 열기
작업 정보	결과	실행 세부정보	실행 그래프
이 문으로 이름이 data인 테이블이 교체되었습니다.		타이틀로 이동	

5-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

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

[결과 이미지를 넣어주세요]

- 가장 최근 구매 일자를 **MAX()** 함수로 찾아보기

```
-- version 1
SELECT
  (SELECT DATE(MAX(InvoiceDate)) FROM modulabs_project.data) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM modulabs_project.data;

-- version 2
SELECT
  DATE(MAX(InvoiceDate) OVER ()) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

자트

JSON

실행 세부정보

실행 그래프

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-12-09	2011-11-03	574239	23318	12	2011-11-03 12:43:0
2	2011-12-09	2011-04-10	549561	23194	1	2011-04-10 13:36:0
3	2011-12-09	2011-04-04	C548723	85063	-1	2011-04-04 09:02:0
4	2011-12-09	2010-12-16	539308	22610	36	2010-12-16 17:48:0
5	2011-12-09	2011-10-03	569247	23317	12	2011-10-03 10:08:0
6	2011-12-09	2011-05-26	554817	23108	2	2011-05-26 14:08:0
7	2011-12-09	2011-07-19	560580	22569	24	2011-07-19 15:23:0
8	2011-12-09	2011-11-23	578305	23007	5	2011-11-23 15:44:0

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 modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	InvoiceDay			
1	12544	2011-11-10			
2	13568	2011-06-19			
3	13824	2011-11-07			
4	14080	2011-11-07			
5	14336	2011-11-23			
6	14592	2011-11-04			
7	15104	2011-06-26			
8	15360	2011-10-31			
9	15872	2011-11-25			

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

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

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

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장 다음에서 열기

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID	recency			
1	14854	78			
2	16650	71			
3	13068	10			
4	18191	261			
5	16400	94			
6	16666	19			
7	16932	53			
8	17190	58			
9	12843	65			

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]

Frequency

```

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.rececy
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

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다. 테이블로 이동

쿼리 결과

작업 정보 결과 자트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	rececy
1	12713	1	505	0
2	18010	1	60	256
3	12792	1	215	256
4	15083	1	38	256
5	13298	1	96	1
6	14569	1	79	1
7	15520	1	314	1
8	13436	1	76	1
9	14476	1	110	257

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

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT CustomerID,
       ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
FROM modulabs_project.data
GROUP BY CustomerID;

```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 자트 JSON 실행 세부정보 실행 그래프

행	CustomerID	user_total
1	12544	300.0
2	13568	187.0
3	13824	1699.0
4	14080	46.0
5	14336	1615.0
6	14592	558.0
7	15104	969.0
8	15360	428.0
9	15872	316.0

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

- 고객별 평균 거래 금액 계산

- 고객별 평균 거래 금액을 구하기 위해 1) `data` 테이블을 `user_rf` 테이블과 조인(LEFT JOIN) 한 후, 2) `purchase_cnt` 로 나누어서 3) `user_rfm` 테이블로 저장하기

```
CREATE OR REPLACE TABLE 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 modulabs_project.user_rf AS rf
LEFT JOIN (
  SELECT CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
  FROM modulabs_project.data
  GROUP BY CustomerID
) AS 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. [테이블로 이동](#)

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

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

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 결과 자트 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	18010	1	60	256	175.0	175.0
3	15083	1	38	256	88.0	88.0
4	12792	1	215	256	345.0	345.0
5	14569	1	79	1	227.0	227.0
6	13436	1	76	1	197.0	197.0
7	15520	1	314	1	344.0	344.0
8	13298	1	96	1	360.0	360.0
9	14476	1	110	257	193.0	193.0

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

5-8. 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 modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM 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

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다. [테이블로 이동](#)

쿼리 결과

작업 정보 결과 자트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	15753	1	144	304	79.0	79.0	1
2	17307	1	-144	365	-153.0	-153.0	1
3	13307	1	4	120	15.0	15.0	1
4	16990	1	100	218	179.0	179.0	1
5	12943	1	-1	301	-4.0	-4.0	1
6	17752	1	192	359	81.0	81.0	1
7	16061	1	-1	269	-30.0	-30.0	1
8	15070	1	36	372	106.0	106.0	1
9	16257	1	1	176	22.0	22.0	1

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

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 균 구매 소요 일수를 계산하고, 그 결과를 `user_data`에 통합

```
CREATE OR REPLACE TABLE 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
      modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM modulabs_project.user_data AS u
```

```
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

실행 세부정보

실행 그래프

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

테이블로 이동

쿼리 결과

결과 저장

다음에서 열기

작업 정보

결과

자트

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	14432	6	2013	9	2248.0	374.666666666666...	256	0.2
2	12428	11	3477	25	6366.0	578.727272727272...	256	0.87
3	13268	14	3525	17	3106.0	221.857142857142...	256	0.56
4	16737	1	288	53	418.0	418.0	1	0.0
5	17923	1	50	282	208.0	208.0	1	0.0
6	17331	1	16	123	175.0	175.0	1	0.0
7	17986	1	10	56	21.0	21.0	1	0.0
8	13307	1	4	120	15.0	15.0	1	0.0
9	16061	1	-1	269	-30.0	-30.0	1	0.0

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(InvoiceNo) AS total_transactions,
    SUM(CASE WHEN Quantity < 0 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 * 100, 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

결과 저장 | 다음에서 열기

작업 정보 | **결과** | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_data인 테이블이 교체되었습니다. [테이블로 이동](#)

- 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

결과 저장

다음에서 열기

작업 정보

결과

자식

JSON

실행 세부정보

실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transaction	cancel_frequency	cancel_rate
1	18233	1	4	325	440.0	440.0	1	0.0	1	0	0.0
2	15510	1	2	330	250.0	250.0	1	0.0	1	0	0.0
3	15657	1	24	22	30.0	30.0	1	0.0	1	0	0.0
4	18184	1	60	15	50.0	50.0	1	0.0	1	0	0.0
5	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0
6	13391	1	4	203	60.0	60.0	1	0.0	1	0	0.0
7	18068	1	6	289	102.0	102.0	1	0.0	1	0	0.0
8	12943	1	-1	301	-4.0	-4.0	1	0.0	1	1	100.0
9	17382	1	24	65	50.0	50.0	1	0.0	1	0	0.0

페이지당 결과 수:

50

1 ~ 50 (전체 4362행)

|<

<

>

>|

인사이트 / 회고 정리

인사이트

- 결측치가 전체 데이터에서 차지 하는 비율이 어느 정도이냐에 따라서 결측치 처리 전략이 달라질 수 있을 듯 합니다.
- 구하고자 하는 것이 무엇인지를 명확하게 정의하는 것이 필요하다고 생각합니다.
- 거래 건수가 1개이면서 구매한 아이템 수량이나 총 지출액 등이 (-)인 데이터가 일부 있습니다. 이런 특이 데이터는 실제 분석 과정에서 체크가 되어야 할 듯 합니다.
- 구매 건수, 구매액 등과 같은 지표 이외에 제품 다양성, 평균 구매 주기, 구매 취소 경험 등과 같은 다양한 지표가 있다는 것을 알았습니다.

회고

- 실제 데이터를 가지고 프로젝트 형식으로 sql을 공부하니까 구문을 익히고, 이해하는데 도움이 많이 되었습니다.
- 방대한 양의 데이터를 다루기 위해서는 꼼꼼함이 필요할 듯
- 최종 작업을 마치고, 프로젝트 제출 단계에서 BigQuery에서 데이터를 다운로드 받을 때 실수 아닌 실수로 500줄 가까이 작성한 쿼리 파일이 날라갔습니다.. 쿼리 작성할 때도 그 때 그 때 저장을 꼭 해야 겠다는 것을 느꼈습니다. (노션 pdf 파일이 있어서 그나마 다행...)
-