고객을 세그먼테이션하자! [프로젝트] - 이유주

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `t-emissary-470201-q2.modulabs_project.data`
LIMIT 10;
```

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

페이지당 결과 수: 50 1 – 10 (전체 10행)

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*) AS row_count
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

행	row_count
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_InvoiceDate,
  COUNT(UnitPrice) AS COUNT_UnitPrice,
  COUNT(CustomerID) AS COUNT_CustomerID,
  COUNT(Country) AS COUNT_Country
FROM `t-emissary-470201-q2.modulabs_project.data`
```

행	COUNT_InvoiceNo	COUNT_Stockcode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUNT_Custome...	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 `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'StockCode',
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) /
COUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'Description',
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) /
COUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'Quantity',
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / C
OUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'InvoiceDate',
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) /
COUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'UnitPrice',
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / C
OUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
SELECT
  'CustomerID',
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) /
COUNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
UNION ALL
```

```
SELECT
  'Country',
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / CO
UNT(*) * 100, 2)
FROM `t-emissary-470201-q2.modulabs_project.data`
ORDER BY missing_percentage DESC;
```

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE stockcode = '85123A'
  AND Description IS NOT NULL
```

행	Description
1	?
2	CREAM HANGING HEART T-LIGHT HOLDER
3	WHITE HANGING HEART T-LIGHT HOLDER
4	wrongly marked carton 22804

결측치 처리

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

```
DELETE FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE CustomerID IS NULL
  OR Description IS NULL
  OR TRIM(Description) = '';
```

이 문으로 data의 행 135,080개가 삭제되었습니다.

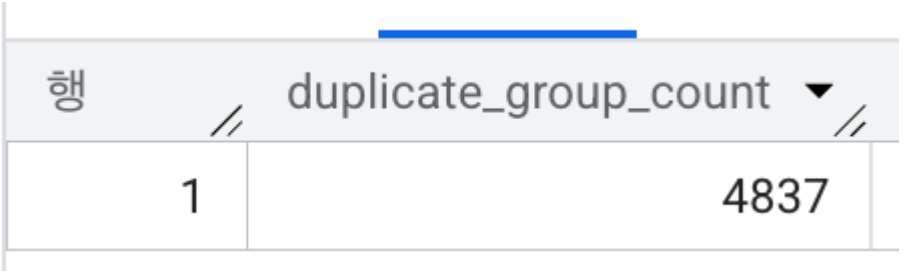
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

- **중복된 행의 수를 세어보기**

- **8개의 컬럼에 그룹 함수를 적용한 후, COUNT가 1보다 큰 데이터를 세어보기**

```
SELECT COUNT(*) AS duplicate_group_count
FROM (
  SELECT
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate,
    UnitPrice, CustomerID, Country
  FROM `t-emissary-470201-q2.modulabs_project.data`
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate,
    UnitPrice, CustomerID, Country
  HAVING COUNT(*) > 1
);
```



행	duplicate_group_count
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 `t-emissary-470201-q2.modulabs_project.
data` AS
SELECT DISTINCT *
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGH...	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT H...	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT ...	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010-12-01 08:34:00 UTC	2.1	13047	United Kingdom
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010-12-01 08:34:00 UTC	2.1	13047	United Kingdom
13	536367	22749	FELTCRAFT PRINCESS CHARLO...	8	2010-12-01 08:34:00 UTC	3.75	13047	United Kingdom
14	536367	22310	IVORY KNITTED MUG COSY	6	2010-12-01 08:34:00 UTC	1.65	13047	United Kingdom
15	536367	84969	BOX OF 6 ASSORTED COLOUR T...	6	2010-12-01 08:34:00 UTC	4.25	13047	United Kingdom
16	536367	22623	BOX OF VINTAGE JIGSAW BLOC...	3	2010-12-01 08:34:00 UTC	4.95	13047	United Kingdom
17	536367	22622	BOX OF VINTAGE ALPHABET B...	2	2010-12-01 08:34:00 UTC	9.95	13047	United Kingdom
18	536367	21754	HOME BUILDING BLOCK WORD	3	2010-12-01 08:34:00 UTC	5.95	13047	United Kingdom
19	536367	21755	LOVE BUILDING BLOCK WORD	3	2010-12-01 08:34:00 UTC	5.95	13047	United Kingdom
20	536367	21777	RECIPE BOX WITH METAL HEART	4	2010-12-01 08:34:00 UTC	7.95	13047	United Kingdom
21	536367	48187	DOORMAT NEW ENGLAND	4	2010-12-01 08:34:00 UTC	7.95	13047	United Kingdom

페이지당 결과 수: 50 1 – 50 (전체 541909행)

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

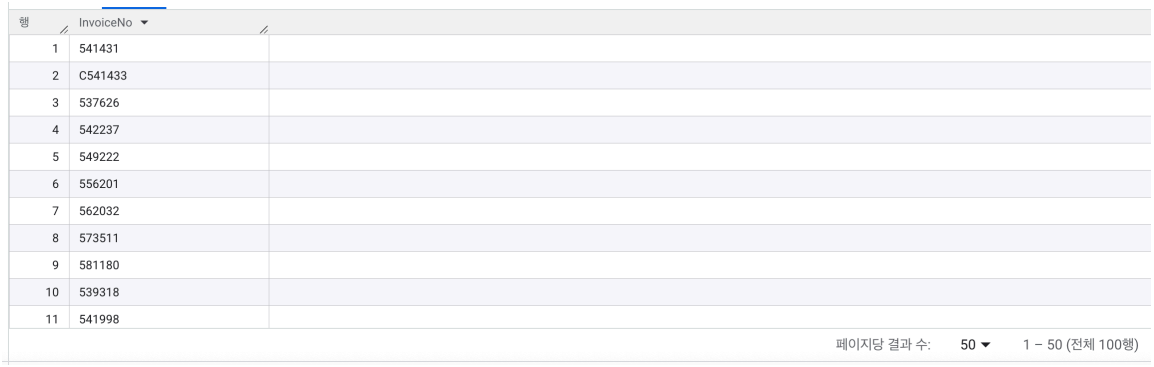
- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

행	unique_invoice_count
1	22190

- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM `t-emissary-470201-q2.modulabs_project.data`
LIMIT 100;
```

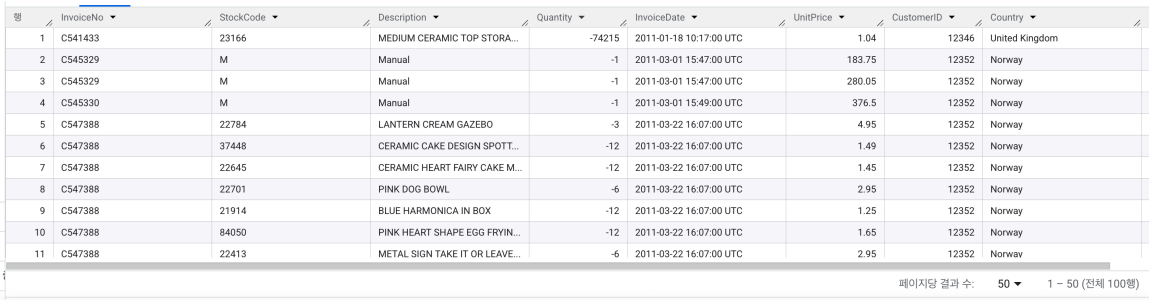


행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100
```



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
6	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
7	C547388	22645	CERAMIC HEART FAIRY CAKE M...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
8	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
9	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
10	C547388	84050	PINK HEART SHAPE EGG FRYIN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
11	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway

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)
* 100.0 / COUNT(*), 1
  ) AS canceled_ratio_percent
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

행	canceled_ratio_percent
1	2.2

StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

행	unique_stockcode_count
1	3684

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

```
SELECT
  StockCode,
```

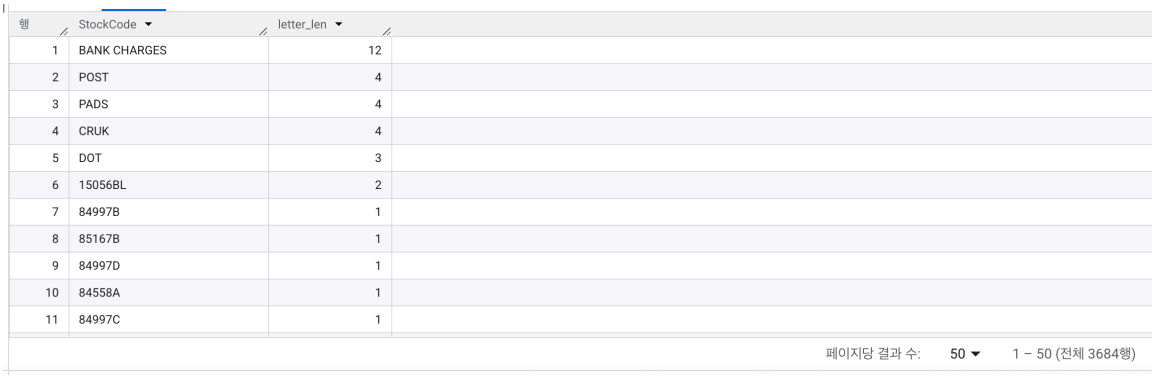
```
  COUNT(*) AS sell_cnt
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

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

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

```
-- StockCode의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고
SELECT DISTINCT
  StockCode,
  LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS letter_len
```

```
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE StockCode IS NOT NULL
ORDER BY letter_len DESC, StockCode;

-- 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 출력하기
SELECT DISTINCT
  StockCode,
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS digit_count
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE StockCode IS NOT NULL
  AND (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) IN (0, 1)
ORDER BY StockCode;
```



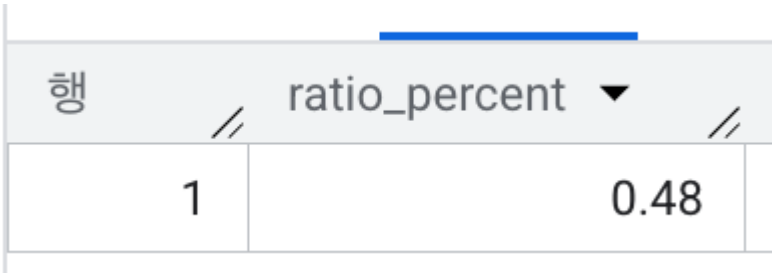
행	StockCode	letter_len
1	BANK CHARGES	12
2	POST	4
3	PADS	4
4	CRUK	4
5	DOT	3
6	15056BL	2
7	84997B	1
8	85167B	1
9	84997D	1
10	84558A	1
11	84997C	1

페이지당 결과 수: 50 1 – 50 (전체 3684행)

행	StockCode	digit_count
1	BANK CHARGES	0
2	C2	1
3	CRUK	0
4	D	0
5	DOT	0
6	M	0
7	PADS	0
8	POST	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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT
  ROUND(
    100 * SAFE_DIVIDE(
      COUNTIF(
        (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode,
r'[0-9]', ''))) IN (0, 1)
      ),
      COUNT(*)
    ),
    2
  ) AS ratio_percent
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE StockCode IS NOT NULL;
```



행	ratio_percent
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM `t-emissary-470201-q2.modulabs_project.data`
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCo
de, r'[0-9]', '')) <=1
  )
);
```

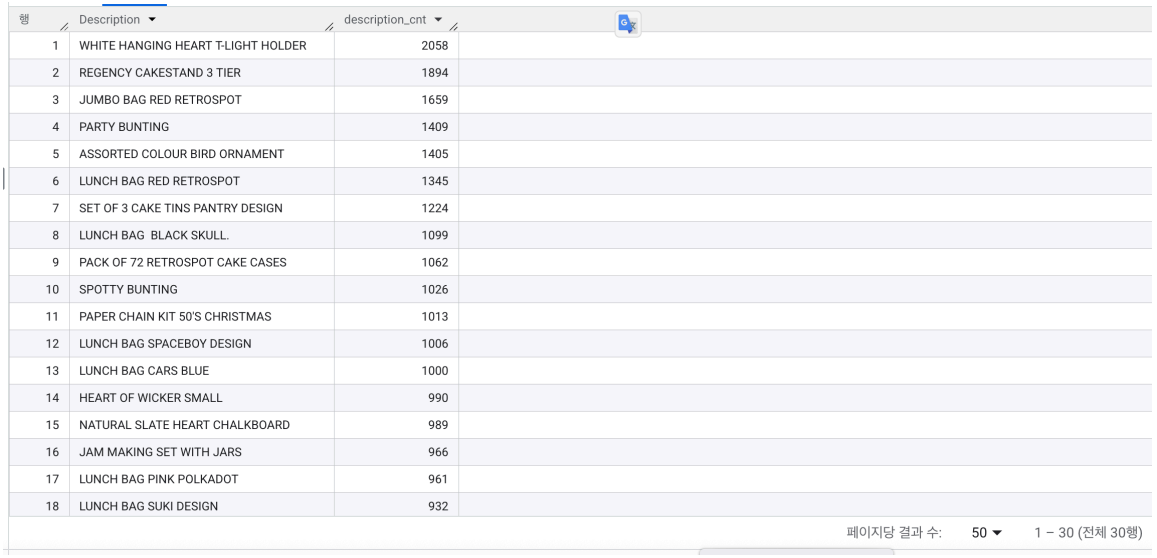
이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT
  Description,
  COUNT(*) AS description_cnt
```

```
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

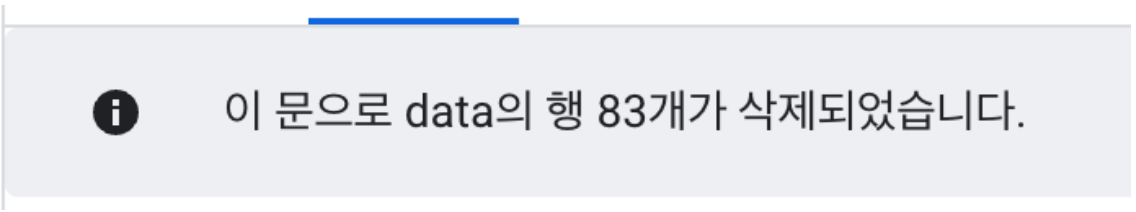


행	Description	description_cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK 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
15	NATURAL SLATE HEART CHALKBOARD	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961
18	LUNCH BAG SUKI DESIGN	932

페이지당 결과 수: 50 1 – 30 (전체 30행)

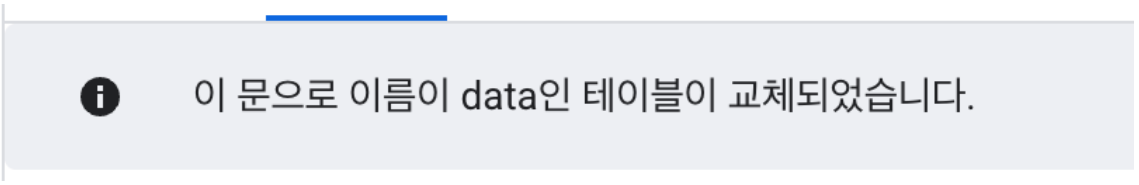
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `t-emissary-470201-q2.modulabs_project.
data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `t-emissary-470201-q2.modulabs_project.data`;
```



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM `t-emissary-470201-q2.modulabs_project.data`;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.9049567574059911

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

```
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(quantity) AS min_quantity,
```

```
  MAX(quantity) AS max_quantity,
  AVG(quantity) AS avg_quantity
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE UnitPrice = 0;
```

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.51515151515139

- `UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE `t-emissary-470201-q2.modulabs_project.data` AS
SELECT *
FROM `t-emissary-470201-q2.modulabs_project.data`
WHERE UnitPrice != 0;
```

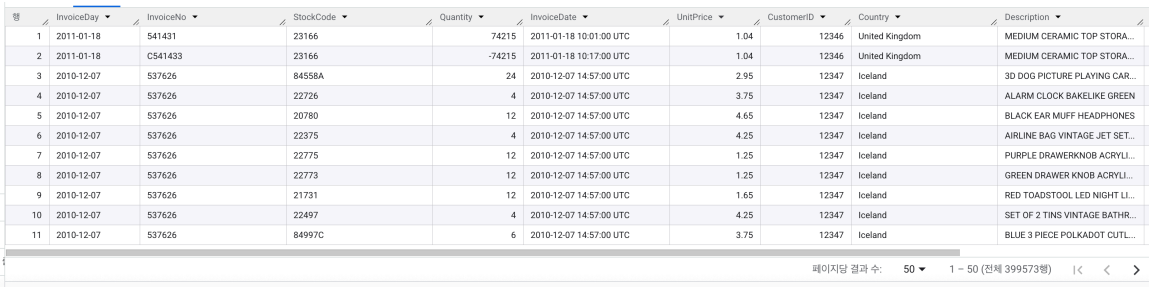
이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM `t-emissary-470201-q2.modulabs_project.data`
```

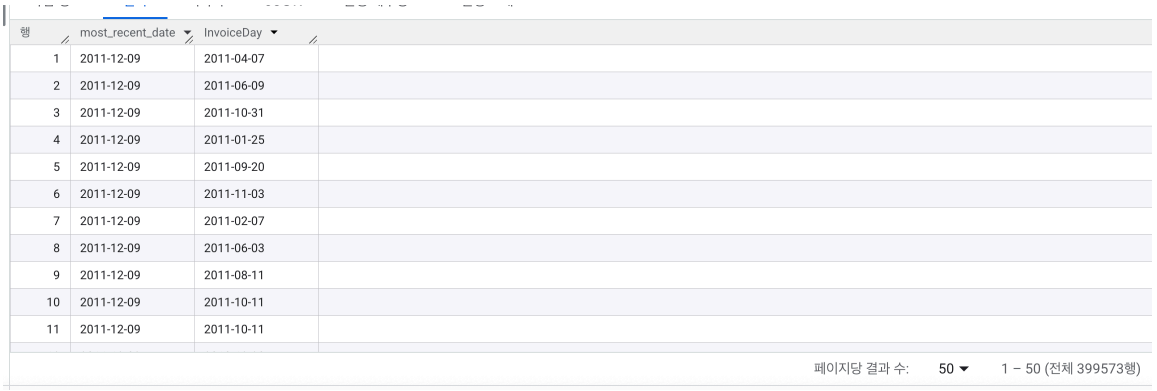


행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORA...
3	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PLAYING CAR...
4	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE GREEN
5	2010-12-07	537626	20780	12	2010-12-07 14:57:00 UTC	4.65	12347	Iceland	BLACK EAR MUFF HEADPHONES
6	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	AIRLINE BAG VINTAGE JET SET...
7	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PURPLE DRAWERKNOB ACRYLI...
8	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNOB ACRYLI...
9	2010-12-07	537626	21731	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	RED TOADSTOOL LED NIGHT LI...
10	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	SET OF 2 TINS VINTAGE BATHR...
11	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PIECE POLKADOT CUTL...

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 `t-emissary-470201-q2.modulabs_project.data`;
```



행	most_recent_date	InvoiceDay
1	2011-12-09	2011-04-07
2	2011-12-09	2011-06-09
3	2011-12-09	2011-10-31
4	2011-12-09	2011-01-25
5	2011-12-09	2011-09-20
6	2011-12-09	2011-11-03
7	2011-12-09	2011-02-07
8	2011-12-09	2011-06-03
9	2011-12-09	2011-08-11
10	2011-12-09	2011-10-11
11	2011-12-09	2011-10-11

페이지당 결과 수: 50 1 – 50 (전체 399573행)

- **유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기**

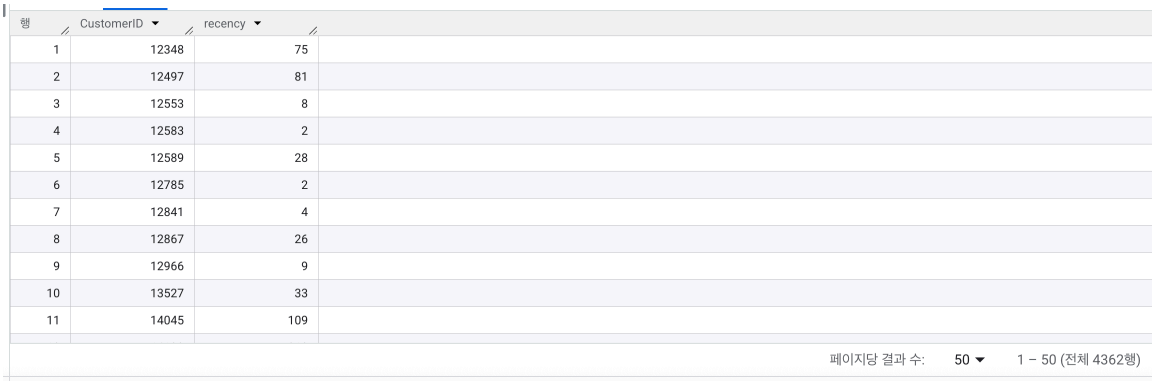
```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY CustomerID
ORDER BY CustomerID;
```

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

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 rece
ncy
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `t-emissary-470201-q2.modulabs_project.data`
  GROUP BY CustomerID
);
```

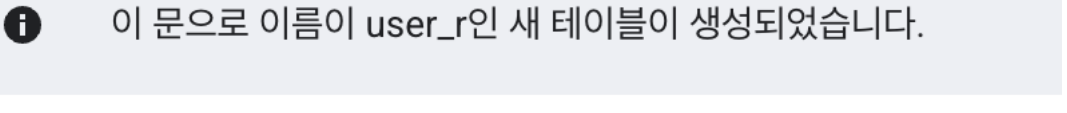


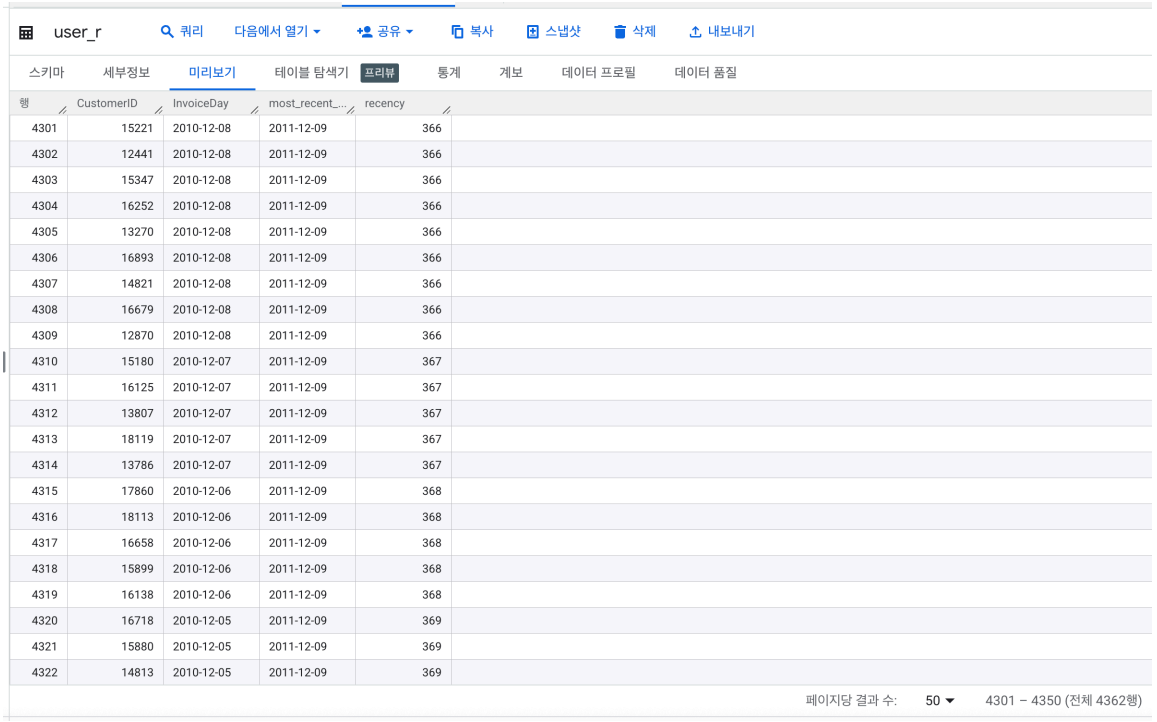
행	CustomerID	recency
1	12348	75
2	12497	81
3	12553	8
4	12583	2
5	12589	28
6	12785	2
7	12841	4
8	12867	26
9	12966	9
10	13527	33
11	14045	109

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 `t-emissary-470201-q2.modulabs_project.
user_r` AS
WITH last_per_customer AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `t-emissary-470201-q2.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT
  CustomerID,
  InvoiceDay,
  MAX(InvoiceDay) OVER() AS most_recent_date,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS rece
ncy
FROM last_per_customer;
```





user_r 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

행	CustomerID	InvoiceDay	most_recent_...	recency
4301	15221	2010-12-08	2011-12-09	366
4302	12441	2010-12-08	2011-12-09	366
4303	15347	2010-12-08	2011-12-09	366
4304	16252	2010-12-08	2011-12-09	366
4305	13270	2010-12-08	2011-12-09	366
4306	16893	2010-12-08	2011-12-09	366
4307	14821	2010-12-08	2011-12-09	366
4308	16679	2010-12-08	2011-12-09	366
4309	12870	2010-12-08	2011-12-09	366
4310	15180	2010-12-07	2011-12-09	367
4311	16125	2010-12-07	2011-12-09	367
4312	13807	2010-12-07	2011-12-09	367
4313	18119	2010-12-07	2011-12-09	367
4314	13786	2010-12-07	2011-12-09	367
4315	17860	2010-12-06	2011-12-09	368
4316	18113	2010-12-06	2011-12-09	368
4317	16658	2010-12-06	2011-12-09	368
4318	15899	2010-12-06	2011-12-09	368
4319	16138	2010-12-06	2011-12-09	368
4320	16718	2010-12-05	2011-12-09	369
4321	15880	2010-12-05	2011-12-09	369
4322	14813	2010-12-05	2011-12-09	369

페이지당 결과 수: 50 4301 – 4350 (전체 4362행)

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY CustomerID
ORDER BY purchase_cnt DESC;
```

행	CustomerID	purchase_cnt
1	14911	242
2	12748	217
3	17841	169
4	14606	125
5	13089	118
6	15311	118
7	12971	88
8	13408	75
9	14646	73
10	16029	66
11	14156	64

페이지당 결과 수: 50 1 – 50 (전체 4362행)

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY CustomerID
ORDER BY item_cnt DESC;
```

행	CustomerID	item_cnt
1	14646	196556
2	12415	76946
3	14911	76823
4	17450	69021
5	18102	64124
6	17511	63014
7	13694	61904
8	14298	58021
9	14156	56896
10	16684	49391
11	15311	37673

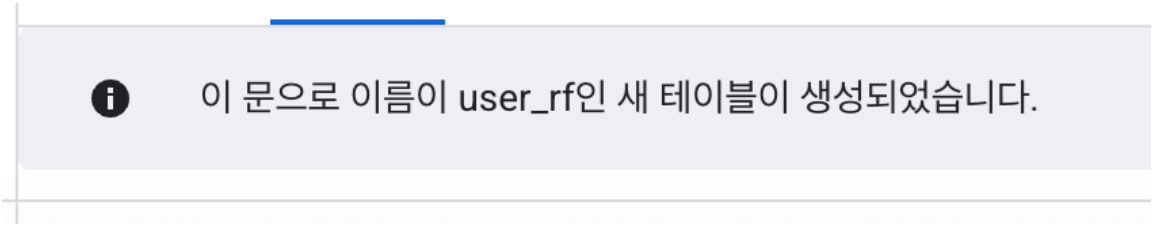
페이지당 결과 수: 50 1 – 50 (전체 4362행)

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

```
-- 전체 거래 건수 계산와 구매한 아이템의 총 수량 계산의 결과를 합쳐서 user_rf 라는 이름의 테이블에 저장하기
CREATE OR REPLACE TABLE `t-emissary-470201-q2.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM `t-emissary-470201-q2.modulabs_project.data`
```

```
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM `t-emissary-470201-q2.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 `t-emissary-470201-q2.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	13436	1	76	1
3	14569	1	79	1
4	13298	1	96	1
5	15520	1	314	1
6	15195	1	1404	2
7	14204	1	72	2
8	15471	1	256	2
9	17914	1	457	3
10	14578	1	240	3
11	15992	1	17	3
12	12650	1	250	3
13	12442	1	181	3
14	15318	1	642	3
15	16528	1	171	3
16	16569	1	93	3
17	12478	1	233	3
18	14219	1	78	4
19	15097	1	170	4
20	16597	1	184	4
21	13790	1	748	4
22	18015	1	157	4

페이지당 결과 수: 50 1 – 50 (전체 4362행)

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `t-emissary-470201-q2.modulabs_project.data`
GROUP BY CustomerID
ORDER BY user_total DESC;
```

행	CustomerID	user_total
1	14646	278778.0
2	18102	259657.3
3	17450	189575.5
4	14911	128768.2
5	12415	123638.2
6	14156	113685.8
7	17511	88138.2
8	16684	65920.1
9	13694	62961.5
10	16029	60369.9
11	15311	59284.2

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 `t-emissary-470201-q2.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ROUND(SAFE_DIVIDE(ut.user_total, rf.purchase_cnt), 1) AS user_average
FROM `t-emissary-470201-q2.modulabs_project.user_rf` AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `t-emissary-470201-q2.modulabs_project.data`
  GROUP By CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

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	794.5	794.5
2	13298	1	96	1	360.0	360.0
3	13436	1	76	1	196.9	196.9
4	14569	1	79	1	227.4	227.4
5	15520	1	314	1	343.5	343.5
6	15471	1	256	2	454.5	454.5
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	150.6	150.6
9	12478	1	233	3	546.0	546.0
10	15992	1	17	3	42.0	42.0
11	15318	1	642	3	312.6	312.6
12	12650	1	250	3	242.4	242.4
13	14578	1	240	3	168.6	168.6
14	16569	1	93	3	124.2	124.2
15	12442	1	181	3	144.1	144.1
16	16528	1	171	3	244.4	244.4
17	17914	1	457	3	329.4	329.4
18	15097	1	170	4	248.1	248.1
19	12367	1	172	4	150.9	150.9
20	13790	1	748	4	348.8	348.8
21	16597	1	184	4	90.0	90.0
22	17383	1	148	4	193.4	193.4

페이지당 결과 수: 50 1 – 50 (전체 4362행)

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `t-emissary-470201-q2.modulabs_project.user_rfm`;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	13298	1	96	1	360.0	360.0
3	13436	1	76	1	196.9	196.9
4	14569	1	79	1	227.4	227.4
5	15520	1	314	1	343.5	343.5
6	15471	1	256	2	454.5	454.5
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	150.6	150.6
9	12478	1	233	3	546.0	546.0
10	15992	1	17	3	42.0	42.0
11	15318	1	642	3	312.6	312.6

페이지당 결과 수: 50 1 – 50 (전체 4362행)

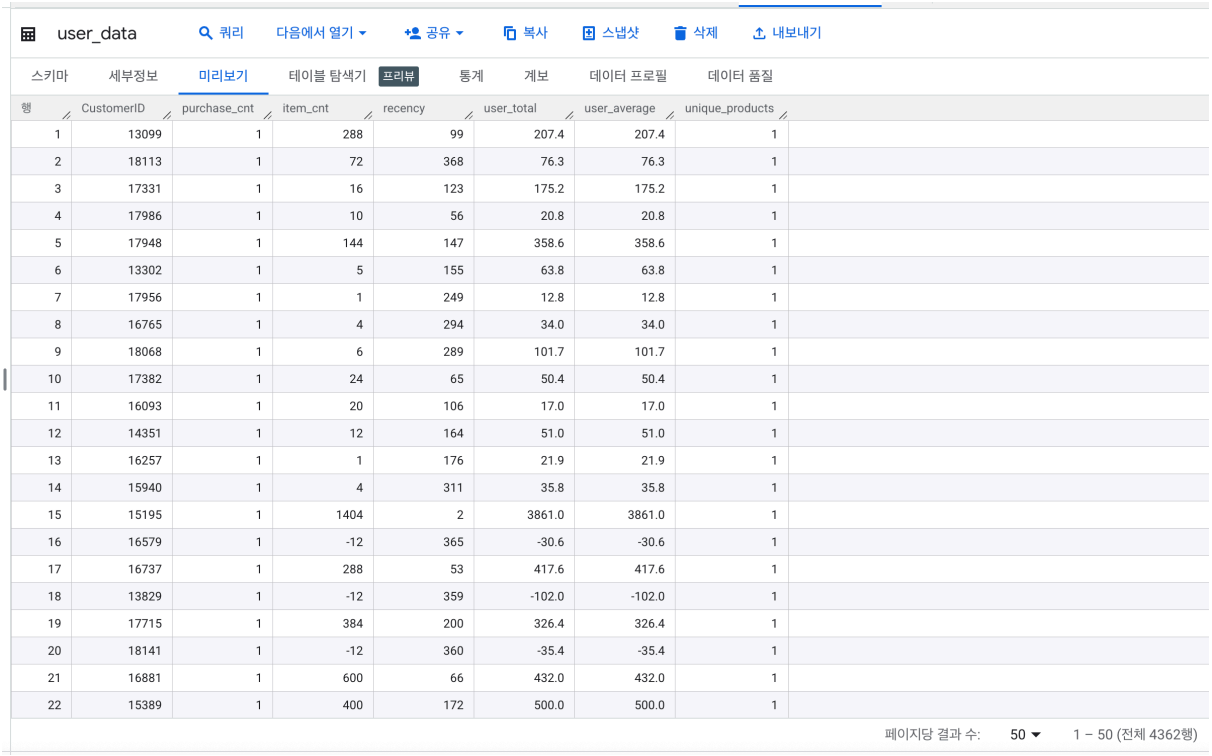
11-8. 추가 Feature 추출

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 `t-emissary-470201-q2.modulabs_project.use
r_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM `t-emissary-470201-q2.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `t-emissary-470201-q2.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.



user_data 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	unique_products
1	13099	1	288	99	207.4	207.4	1
2	18113	1	72	368	76.3	76.3	1
3	17331	1	16	123	175.2	175.2	1
4	17986	1	10	56	20.8	20.8	1
5	17948	1	144	147	358.6	358.6	1
6	13302	1	5	155	63.8	63.8	1
7	17956	1	1	249	12.8	12.8	1
8	16765	1	4	294	34.0	34.0	1
9	18068	1	6	289	101.7	101.7	1
10	17382	1	24	65	50.4	50.4	1
11	16093	1	20	106	17.0	17.0	1
12	14351	1	12	164	51.0	51.0	1
13	16257	1	1	176	21.9	21.9	1
14	15940	1	4	311	35.8	35.8	1
15	15195	1	1404	2	3861.0	3861.0	1
16	16579	1	-12	365	-30.6	-30.6	1
17	16737	1	288	53	417.6	417.6	1
18	13829	1	-12	359	-102.0	-102.0	1
19	17715	1	384	200	326.4	326.4	1
20	18141	1	-12	360	-35.4	-35.4	1
21	16881	1	600	66	432.0	432.0	1
22	15389	1	400	172	500.0	500.0	1

페이지당 결과 수: 50 1 – 50 (전체 4362행)

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE `t-emissary-470201-q2.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
      `t-emissary-470201-q2.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT
  u.*, pi.* EXCEPT (CustomerID)
FROM `t-emissary-470201-q2.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

이 문으로 이름이 user_data인 테이블이 교체되었습니다.

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

스키마 세부정보 미리보기 테이블 탐색기 프리뷰 통계 계보 데이터 프로필 데이터 품질

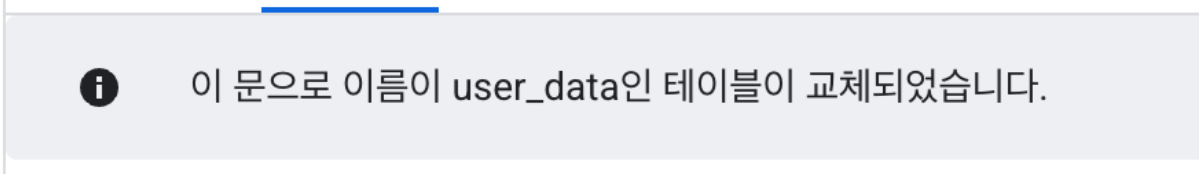
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	17715	1	384	200	326.4	326.4	1	0.0
2	18184	1	60	15	49.8	49.8	1	0.0
3	15118	1	1440	134	244.8	244.8	1	0.0
4	13366	1	144	50	56.2	56.2	1	0.0
5	16737	1	288	53	417.6	417.6	1	0.0
6	17763	1	12	263	15.0	15.0	1	0.0
7	16344	1	18	158	101.1	101.1	1	0.0
8	13841	1	100	252	85.0	85.0	1	0.0
9	16953	1	10	30	20.8	20.8	1	0.0
10	13302	1	5	155	63.8	63.8	1	0.0
11	15940	1	4	311	35.8	35.8	1	0.0
12	16078	1	16	283	79.2	79.2	1	0.0
13	16257	1	1	176	21.9	21.9	1	0.0
14	15313	1	25	110	52.0	52.0	1	0.0
15	16738	1	3	297	3.8	3.8	1	0.0
16	13747	1	8	373	79.6	79.6	1	0.0
17	16995	1	-1	372	-1.3	-1.3	1	0.0
18	15070	1	36	372	106.2	106.2	1	0.0
19	16454	1	2	64	5.9	5.9	1	0.0
20	12603	1	56	21	613.2	613.2	1	0.0
21	13135	1	4300	196	3096.0	3096.0	1	0.0
22	15657	1	24	22	30.0	30.0	1	0.0

페이지당 결과 수: 50 1 – 50 (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 `t-emissary-470201-q2.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT IF(STARTS_WITH(InvoiceNo, 'C'), InvoiceNo, NULL)) AS cancel_frequency
  FROM `t-emissary-470201-q2.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.* EXCEPT(CustomerID),
  SAFE_DIVIDE(t.cancel_frequency, t.total_transactions) AS cancel_rate
FROM `t-emissary-470201-q2.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
  USING (CustomerID);
```



user_data 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	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	15603	1	96	78	48.5	48.5	2	0.0	1	0	0.0
2	15049	1	63	273	121.2	121.2	3	0.0	1	0	0.0
3	15286	1	55	49	240.6	240.6	3	0.0	1	0	0.0
4	14300	1	32	18	43.2	43.2	3	0.0	1	0	0.0
5	16542	1	46	53	119.6	119.6	5	0.0	1	0	0.0
6	15222	1	440	318	585.0	585.0	5	0.0	1	0	0.0
7	14196	1	142	106	335.5	335.5	6	0.0	1	0	0.0
8	17524	1	90	361	145.0	145.0	6	0.0	1	0	0.0
9	13938	1	70	35	71.7	71.7	7	0.0	1	0	0.0
10	15739	1	472	66	449.1	449.1	8	0.0	1	0	0.0
11	14981	1	97	246	102.1	102.1	8	0.0	1	0	0.0
12	12831	1	135	262	215.0	215.0	9	0.0	1	0	0.0
13	12977	1	12	156	39.2	39.2	9	0.0	1	0	0.0
14	14806	1	330	75	193.4	193.4	11	0.0	1	0	0.0
15	15277	1	205	46	255.9	255.9	12	0.0	1	0	0.0
16	14079	1	301	124	375.1	375.1	12	0.0	1	0	0.0
17	14885	1	724	131	765.3	765.3	15	0.0	1	0	0.0
18	17343	1	56	355	193.6	193.6	17	0.0	1	0	0.0
19	13449	1	222	23	291.9	291.9	19	0.0	1	0	0.0
20	15652	1	208	91	337.7	337.7	21	0.0	1	0	0.0
21	15090	1	208	308	381.3	381.3	23	0.0	1	0	0.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

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

```
SELECT *
FROM `t-emissary-470201-q2.modulabs_project.user_data`
ORDER BY CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	12346	2	0	325	0.0	0.0	1	0.0	2	1	0.5
2	12347	7	2458	2	4310.0	615.7	103	2.0	7	0	0.0
3	12348	4	2332	75	1437.2	359.3	21	10.85	4	0	0.0
4	12349	1	630	18	1457.5	1457.5	72	0.0	1	0	0.0
5	12350	1	196	310	294.4	294.4	16	0.0	1	0	0.0
6	12352	8	463	36	1265.4	158.2	57	3.11	8	1	0.13
7	12353	1	20	204	89.0	89.0	4	0.0	1	0	0.0
8	12354	1	530	232	1079.4	1079.4	58	0.0	1	0	0.0
9	12355	1	240	214	459.4	459.4	13	0.0	1	0	0.0
10	12356	3	1573	22	2487.4	829.1	52	5.3	3	0	0.0
11	12357	1	2708	33	6207.7	6207.7	131	0.0	1	0	0.0
12	12358	2	242	1	928.1	464.1	12	9.31	2	0	0.0
13	12359	6	1599	7	6183.0	1030.5	214	1.28	6	2	0.33
14	12360	3	1156	52	2302.1	767.4	104	1.18	3	0	0.0
15	12361	1	90	287	174.9	174.9	9	0.0	1	0	0.0
16	12362	13	2180	3	4665.6	358.9	200	1.09	13	3	0.23
17	12363	2	408	109	552.0	276.0	23	6.0	2	0	0.0
18	12364	4	1499	7	1208.1	302.0	69	1.29	4	0	0.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : 틀린 코드를 그냥 넘어가지 않고 계속 수정해보기

Problem : 그러다보면 시간이 턱없이 부족하다

Try : 처음부터 잘 작성할 수 있기를.. 시간이 지나면 해결이 될까요..?