고객을 세그먼테이션하자 [프로젝트]

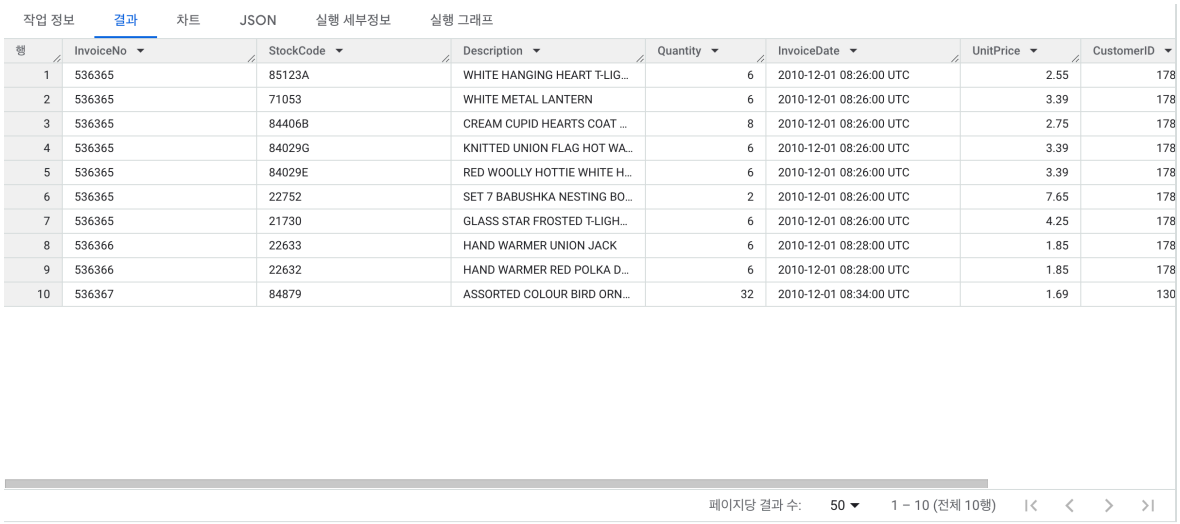
11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]



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

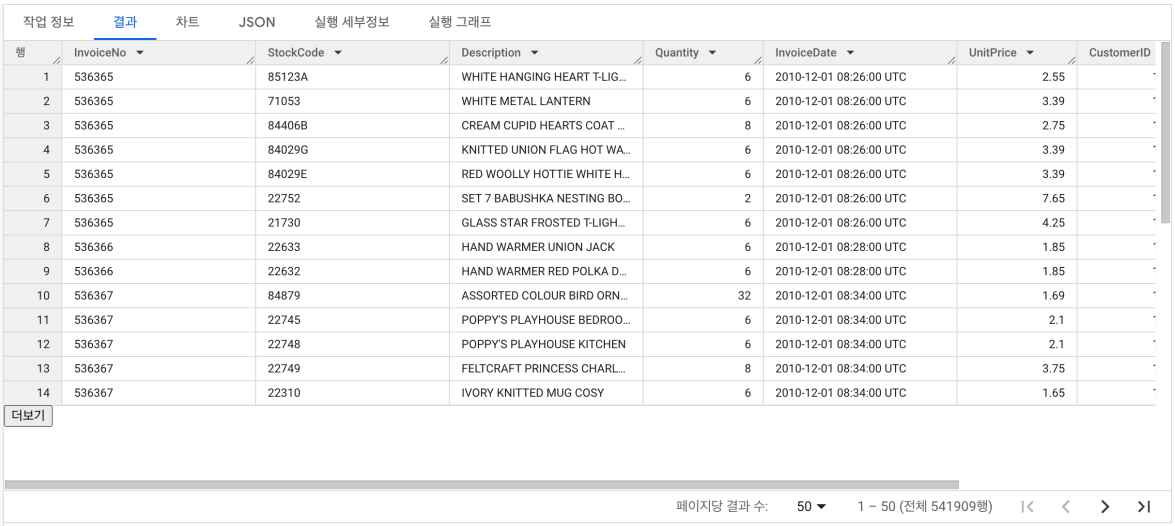
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	178
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	178
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	2.75	178
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	178
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	178
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	178
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	4.25	178
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	178
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	1.85	178
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	130

페이지당 결과 수: 50 1 – 10 (전체 10행)

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
# 전체 데이터는 몇 행으로 구성되어 있는지 확인해 봅시다.
# [[YOUR QUERY]]
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-12-01 08:26:00 UTC	2.55	
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-12-01 08:26:00 UTC	2.75	
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-12-01 08:26:00 UTC	3.39	
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-12-01 08:26:00 UTC	3.39	
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-12-01 08:26:00 UTC	7.65	
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-12-01 08:26:00 UTC	4.25	
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	
9	536366	22632	HAND WARMER RED POLKA D...	6	2010-12-01 08:28:00 UTC	1.85	
10	536367	84879	ASSORTED COLOUR BIRD ORN...	32	2010-12-01 08:34:00 UTC	1.69	
11	536367	22745	POPPY'S PLAYHOUSE BEDROO...	6	2010-12-01 08:34:00 UTC	2.1	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010-12-01 08:34:00 UTC	2.1	
13	536367	22749	FELTCRAFT PRINCESS CHARL...	8	2010-12-01 08:34:00 UTC	3.75	
14	536367	22310	IVORY KNITTED MUG COSY	6	2010-12-01 08:34:00 UTC	1.65	

더보기

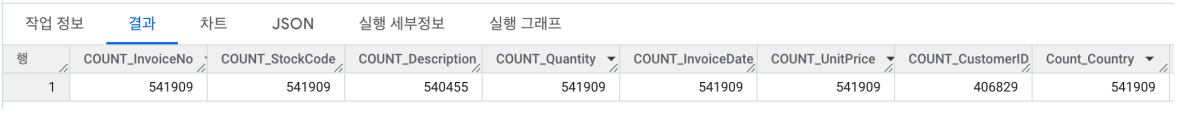
페이지당 결과 수: 50 1 – 50 (전체 541909행)

데이터 수 세기

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

```
# 데이터 수 세기
# [[YOUR QUERY]]
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 concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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

행	COUNT_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**

```
# [[YOUR QUERY]]
SELECT
  'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM concise-isotope-456102-c4.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 concise-isotope-456102-c4.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 concise-isotope-456102-c4.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 concise-isotope-456102-c4.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_percentage
FROM concise-isotope-456102-c4.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 concise-isotope-456102-c4.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 concise-isotope-456102-c4.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 concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

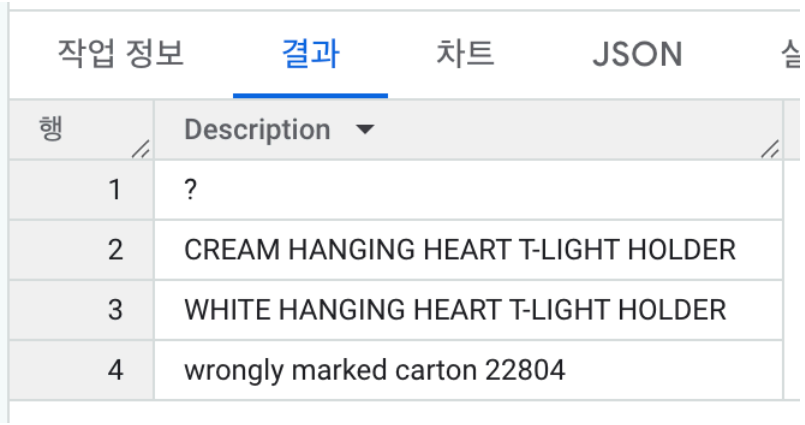
행	column_name	missing_percentage
1	Country	0.0
2	InvoiceNo	0.0
3	Quantity	0.0
4	StockCode	0.0
5	InvoiceDate	0.0
6	CustomerID	24.93
7	UnitPrice	0.0
8	Description	0.27

결측치 처리 전략

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

```
# StockCode = '85123A'의 Description을 추출하는 쿼리문을 작성(중복 제거)
SELECT DISTINCT Description
FROM concise-isotope-456102-c4.modulabs_project.data
WHERE StockCode = '85123A'
ORDER BY Description;
```

[결과 이미지를 넣어주세요]

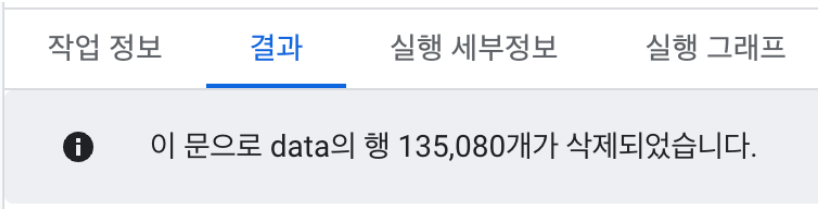


결측치 처리

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

```
# 결측치 처리_결측치 행 제거
DELETE FROM concise-isotope-456102-c4.modulabs_project.data
WHERE Description IS NULL
OR CustomerID IS NULL;
```

[결과 이미지를 넣어주세요]



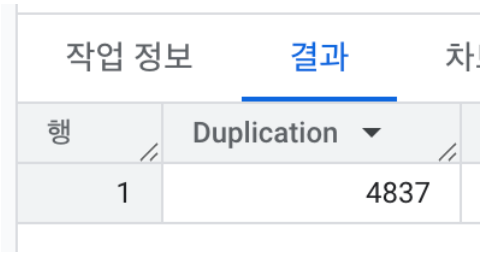
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
-- Q1. 중복된 행의 수 카운트
SELECT COUNT(*) AS Duplication
FROM (
  SELECT InvoiceNo, StockCode,Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY InvoiceNo, StockCode,Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
  HAVING count(*) > 1
);
```

[결과 이미지를 넣어주세요]

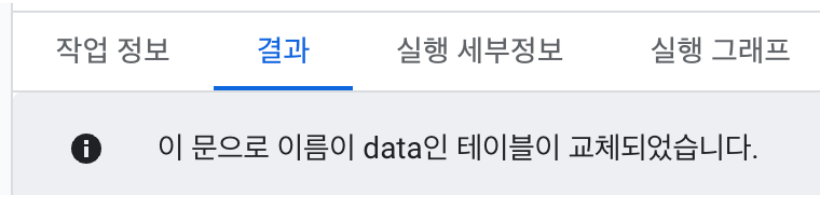


중복값 처리

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

```
-- Q2. 중복값 제거하여 테이블 업데이트
CREATE OR REPLACE TABLE concise-isotope-456102-c4.modulabs_project.data AS
SELECT DISTINCT *
FROM concise-isotope-456102-c4.modulabs_project.data
```

[결과 이미지를 넣어주세요]



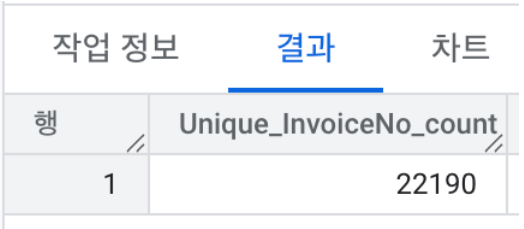
11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- **고유(unique)한 InvoiceNo의 개수를 출력하기**

```
# [[YOUR QUERY]]
-- Q1. InvoiceNo 살펴보기
SELECT COUNT(DISTINCT InvoiceNo) AS Unique_InvoiceNo_count
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



- **고유한 InvoiceNo를 앞에서부터 100개를 출력하기**

```
# [[YOUR QUERY]]
-- Q2. 고유한 InvoiceNo를 100개를 출력
SELECT DISTINCT InvoiceNo
FROM concise-isotope-456102-c4.modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

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

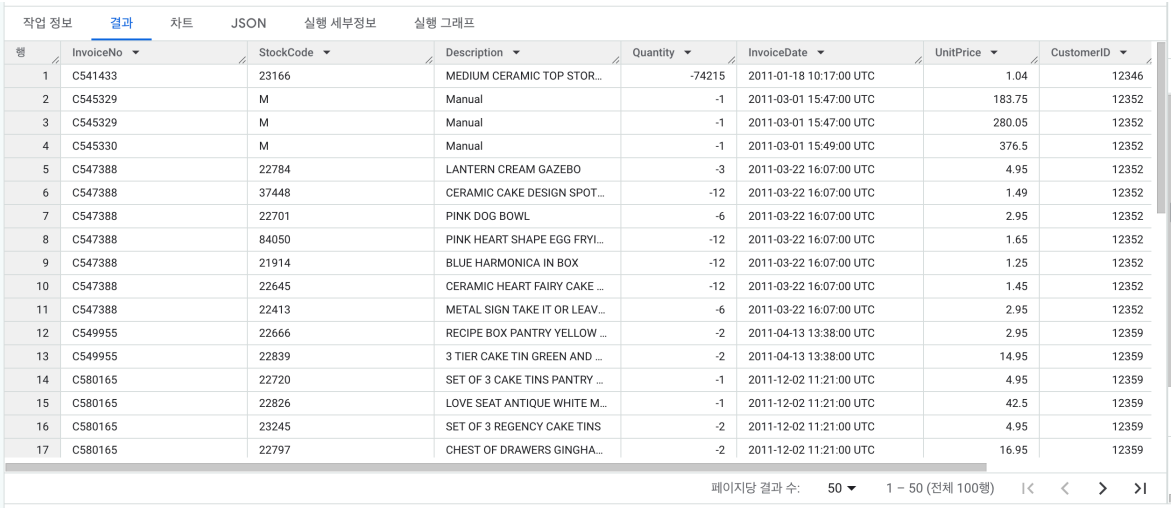
행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998
12	548955
13	568172
14	577609
15	543037
16	544156
17	545323

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
-- Q3. InvoiceNo가 'C'로 시작하는 행을 필터링하여 100행까지 출력
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]



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

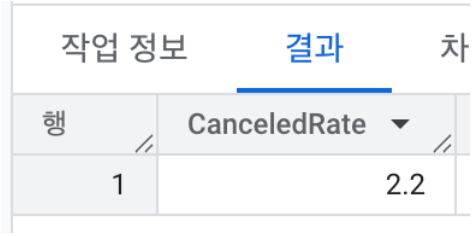
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-74215	2011-01-18 10:17:00 UTC	1.04	12346
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352
5	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352
6	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03-22 16:07:00 UTC	1.49	12352
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352
8	C547388	84050	PINK HEART SHAPE EGG FRYI...	-12	2011-03-22 16:07:00 UTC	1.65	12352
9	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352
10	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352
11	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03-22 16:07:00 UTC	2.95	12352
12	C549955	22666	RECIPE BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC	2.95	12359
13	C549955	22839	3 TIER CAKE TIN GREEN AND ...	-2	2011-04-13 13:38:00 UTC	14.95	12359
14	C580165	22720	SET OF 3 CAKE TINS PANTRY ...	-1	2011-12-02 11:21:00 UTC	4.95	12359
15	C580165	22826	LOVE SEAT ANTIQUE WHITE M...	-1	2011-12-02 11:21:00 UTC	42.5	12359
16	C580165	23245	SET OF 3 REGENCY CAKE TINS	-2	2011-12-02 11:21:00 UTC	4.95	12359
17	C580165	22797	CHEST OF DRAWERS GINGHA...	-2	2011-12-02 11:21:00 UTC	16.95	12359

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
-- Q4. Canceled 인 데이터의 비율(%)_소수점 첫번째 자리
SELECT
  ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(InvoiceNo) * 100, 1) AS CanceledRate
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

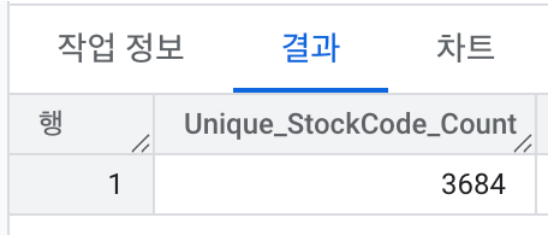


StockCode 살펴보기

- **고유한 StockCode의 개수를 출력하기**

```
-- Q5. 고유한 StockCode 개수 출력
SELECT COUNT(DISTINCT StockCode) AS Unique_StockCode_Count
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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

```
-- Q6. StockCode 별 등장 빈도_상위 10개
SELECT StockCode, COUNT(*) AS sell_cnt
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

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

```
-- Q7. 숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지 확인
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM concise-isotope-456102-c4.modulabs_project.data
)
WHERE number_count = 0
OR number_count = 1;
```

[결과 이미지를 넣어주세요]

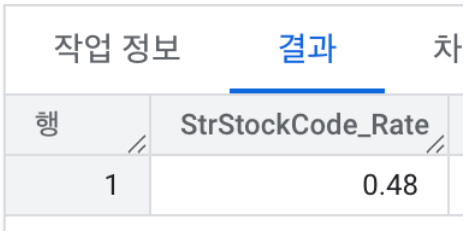
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

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

- `StockCode` **의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
-- Q8. 해당 코드 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트인지_소수점 두번째 자리까지
SELECT
  ROUND(SUM(CASE WHEN number_count = 0 OR number_count = 1 THEN 1 ELSE 0 END) / COUNT(StockCode) * 100, 2) AS St
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM concise-isotope-456102-c4.modulabs_project.data
)
```

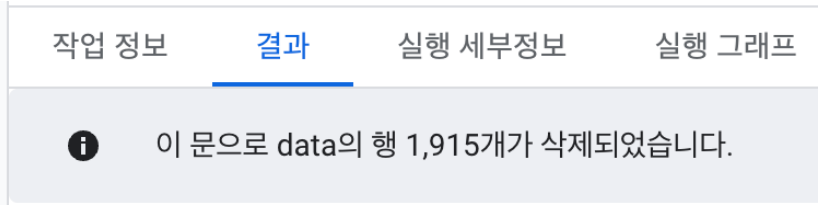
[결과 이미지를 넣어주세요]



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
-- Q9. 제품과 관련되지 않은 거래 기록 제거
DELETE FROM concise-isotope-456102-c4.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 concise-isotope-456102-c4.modulabs_project.data
  )
  WHERE number_count = 0 OR number_count = 1
);
```

[결과 이미지를 넣어주세요]



`Description` 살펴보기

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

```
-- Q10. 고유한 Description 별 출현 빈도를 계산하고 상위 30개 출력
SELECT Description, COUNT(*) AS description_cnt
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]

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

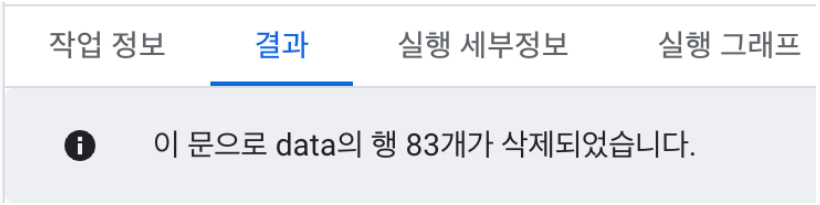
행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990
15	NATURAL SLATE HEART CHAL...	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961

페이지당 결과 수: 50 1 – 30 (전체 30행)

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
-- Q11. 서비스 관련 정보 행들 제거
DELETE
FROM concise-isotope-456102-c4.modulabs_project.data
WHERE
  Description LIKE '%Next Day Carriage%'
  OR Description LIKE '%High Resolution Image%';
```

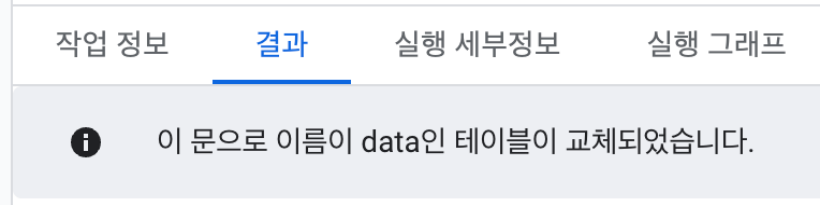
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
-- Q12. 대소문자 혼합 데이터 대문자로 표준화
CREATE OR REPLACE TABLE concise-isotope-456102-c4.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
-- Q13. UnitPrice의 최솟값, 최댓값, 평균
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그

행	min_price	max_price	avg_price
1	0.0	649.5	2.9049567574060684

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

```
-- Q14. 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균
SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM concise-isotope-456102-c4.modulabs_project.data
WHERE UnitPrice = 0.0;
```

[결과 이미지를 넣어주세요]

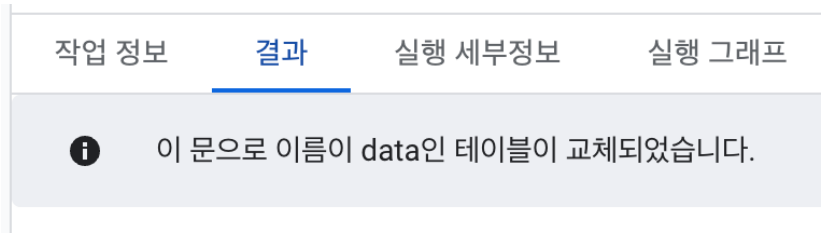
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

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 concise-isotope-456102-c4.modulabs_project.data AS
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.data
WHERE UnitPrice <> 0;
```

[결과 이미지를 넣어주세요]



11-7. RFM 스코어

Recency

- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
-- Q1. DATE 함수를 활용하여 InvoiceDate 컬럼을 연월일 자료형으로 변경
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC
3	2010-12-07	537626	22195	12	2010-12-07 14:57:00 UTC
4	2010-12-07	537626	22725	4	2010-12-07 14:57:00 UTC
5	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC
6	2010-12-07	537626	84997B	6	2010-12-07 14:57:00 UTC
7	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC
8	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC
9	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC
10	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC
11	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC
12	2010-12-07	537626	22726	4	2010-12-07 14:57:00 UTC
13	2010-12-07	537626	22375	4	2010-12-07 14:57:00 UTC
14	2010-12-07	537626	85116	12	2010-12-07 14:57:00 UTC
15	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC
16	2010-12-07	537626	71477	12	2010-12-07 14:57:00 UTC
17	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
-- Q2. 모든 고객을 통틀어 가장 최근 구매 일자
SELECT
  MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM concise-isotope-456102-c4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

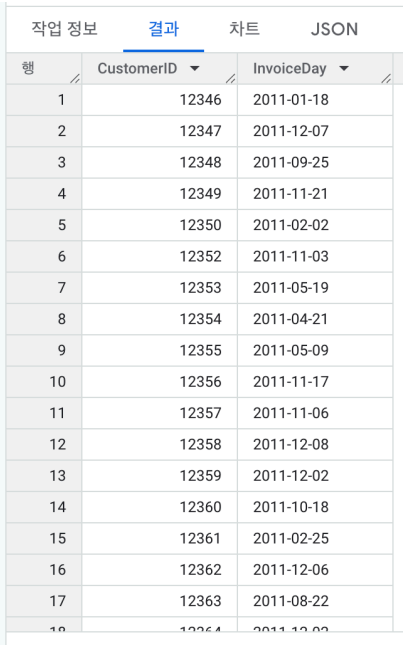
행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
1	2011-12-09	2011-01-18	541431	23166	74215	2011-01-18 10:01
2	2011-12-09	2011-01-18	C541433	23166	-74215	2011-01-18 10:17
3	2011-12-09	2010-12-07	537626	22195	12	2010-12-07 14:57
4	2011-12-09	2010-12-07	537626	22725	4	2010-12-07 14:57
5	2011-12-09	2010-12-07	537626	22805	12	2010-12-07 14:57
6	2011-12-09	2010-12-07	537626	84997B	6	2010-12-07 14:57
7	2011-12-09	2010-12-07	537626	85167B	30	2010-12-07 14:57
8	2011-12-09	2010-12-07	537626	22773	12	2010-12-07 14:57
9	2011-12-09	2010-12-07	537626	21064	6	2010-12-07 14:57
10	2011-12-09	2010-12-07	537626	22729	4	2010-12-07 14:57
11	2011-12-09	2010-12-07	537626	22728	4	2010-12-07 14:57
12	2011-12-09	2010-12-07	537626	22726	4	2010-12-07 14:57
13	2011-12-09	2010-12-07	537626	22375	4	2010-12-07 14:57
14	2011-12-09	2010-12-07	537626	85116	12	2010-12-07 14:57

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

```
-- Q3. 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장
SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
```

```
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]



작업 정보 결과 차트 JSON

행	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17
11	12357	2011-11-06
12	12358	2011-12-08
13	12359	2011-12-02
14	12360	2011-10-18
15	12361	2011-02-25
16	12362	2011-12-06
17	12363	2011-08-22

- **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]

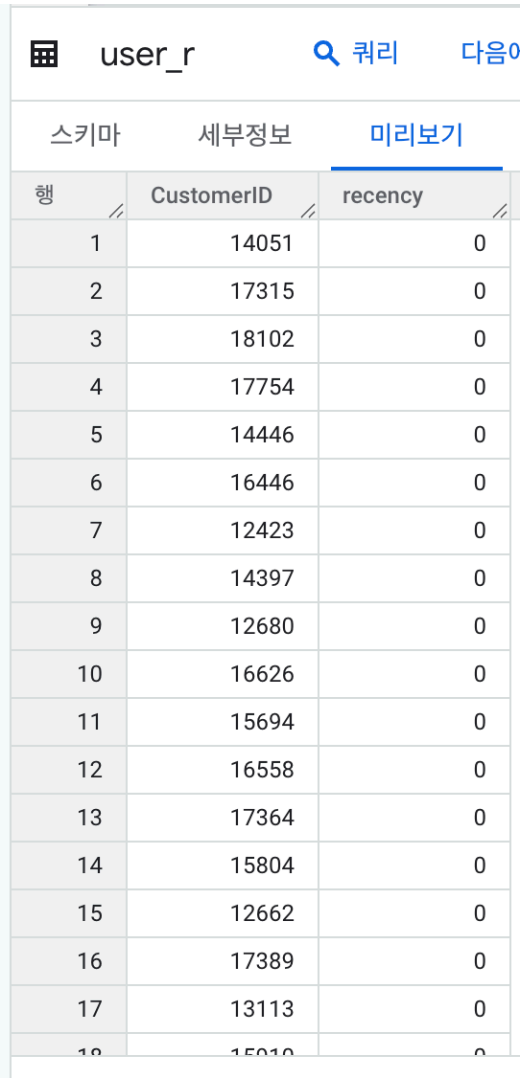
작업 정보 | 결과 | 차트 | JSON

행	CustomerID	recency
5	12638	33
6	12681	14
7	12734	352
8	13085	157
9	13171	8
10	13207	15
11	13246	18
12	13501	242
13	13868	7
14	13898	325
15	14167	39
16	14566	110
17	14850	311
18	14920	212
19	14932	365
20	14950	8
21	14972	128
22	14976	57

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

```
CREATE OR REPLACE TABLE concise-isotope-456102-c4.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 concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]


user_r 쿼리 다음0

스키마 세부정보 미리보기

행	CustomerID	recency
1	14051	0
2	17315	0
3	18102	0
4	17754	0
5	14446	0
6	16446	0
7	12423	0
8	14397	0
9	12680	0
10	16626	0
11	15694	0
12	16558	0
13	17364	0
14	15804	0
15	12662	0
16	17389	0
17	13113	0
18	15910	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
-- Q1. 고객마다 고유한 InvoiceNo의 수
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트 | JSON

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
11	12357	1
12	12358	2
13	12359	6
14	12360	3
15	12361	1
16	12362	13
17	12363	2
18	12364	4

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
-- Q2. 각 고객 별로 구매한 아이템의 총 수량 계산
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트 | JSON

행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196
6	12352	463
7	12353	20
8	12354	530
9	12355	240
10	12356	1573
11	12357	2708
12	12358	242
13	12359	1599
14	12360	1156
15	12361	90
16	12362	2180
17	12363	408

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

```
-- Q3. '1. 전체 거래 건수 계산'과 '2. 구매한 아이템의 총 수량 계산'의 결과를 합쳐서 user_rf라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE concise-isotope-456102-c4.modulabs_project.user_rf AS

WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
),

item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
)

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 concise-isotope-456102-c4.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	14569	1	79	1
3	15520	1	314	1
4	13298	1	96	1
5	13436	1	76	1
6	15471	1	256	2
7	15195	1	1404	2
8	14204	1	72	2
9	15992	1	17	3
10	17914	1	457	3
11	16528	1	171	3
12	12650	1	250	3
13	12478	1	233	3
14	12442	1	181	3
15	15318	1	642	3
16	16569	1	93	3
17	14578	1	240	3
18	13709	1	748	4

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
-- Q1. 고객별 총 지출액을 계산
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
FROM concise-isotope-456102-c4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

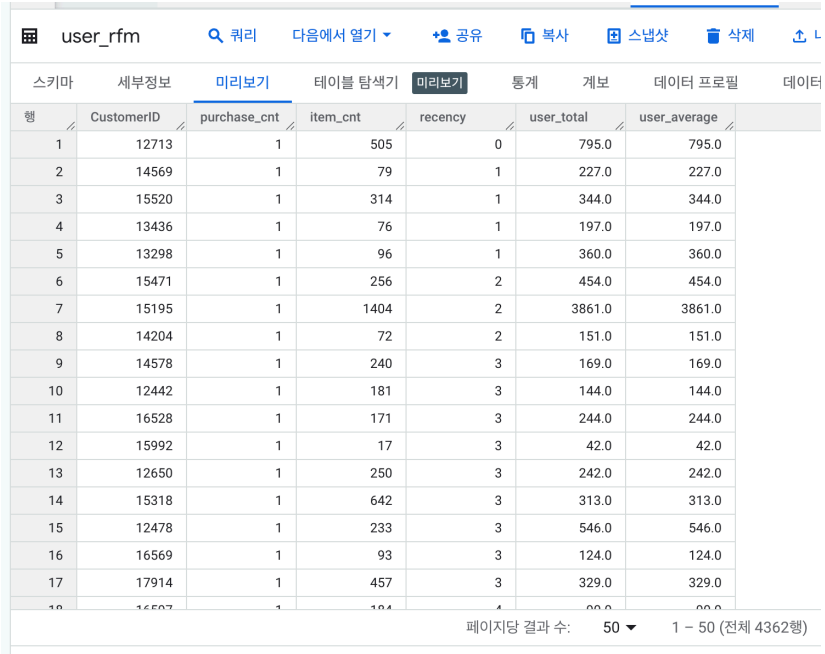
작업 정보 | 결과 | 차트 | JSON

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.0
4	12349	1458.0
5	12350	294.0
6	12352	1265.0
7	12353	89.0
8	12354	1079.0
9	12355	459.0
10	12356	2487.0
11	12357	6208.0
12	12358	928.0
13	12359	6183.0
14	12360	2302.0
15	12361	175.0
16	12362	4666.0
17	12363	552.0
18	12364	1208.0

- **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**

```
-- Q2. 고객별 평균 거래 금액 계산
CREATE OR REPLACE TABLE concise-isotope-456102-c4.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 concise-isotope-456102-c4.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice * Quantity), 0) AS user_total
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	14569	1	79	1	227.0	227.0
3	15520	1	314	1	344.0	344.0
4	13436	1	76	1	197.0	197.0
5	13298	1	96	1	360.0	360.0
6	15471	1	256	2	454.0	454.0
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	151.0	151.0
9	14578	1	240	3	169.0	169.0
10	12442	1	181	3	144.0	144.0
11	16528	1	171	3	244.0	244.0
12	15992	1	17	3	42.0	42.0
13	12650	1	250	3	242.0	242.0
14	15318	1	642	3	313.0	313.0
15	12478	1	233	3	546.0	546.0
16	16569	1	93	3	124.0	124.0
17	17914	1	457	3	329.0	329.0

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
-- 최종 RFM 테이블 출력
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.user_rfm;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	795.0	795.0
2	14569	1	79	1	227.0	227.0
3	15520	1	314	1	344.0	344.0
4	13436	1	76	1	197.0	197.0
5	13298	1	96	1	360.0	360.0
6	15471	1	256	2	454.0	454.0
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	151.0	151.0
9	14578	1	240	3	169.0	169.0
10	12442	1	181	3	144.0	144.0
11	16528	1	171	3	244.0	244.0
12	15992	1	17	3	42.0	42.0
13	12650	1	250	3	242.0	242.0
14	15318	1	642	3	313.0	313.0
15	12478	1	233	3	546.0	546.0
16	16569	1	93	3	124.0	124.0
17	17914	1	457	3	329.0	329.0

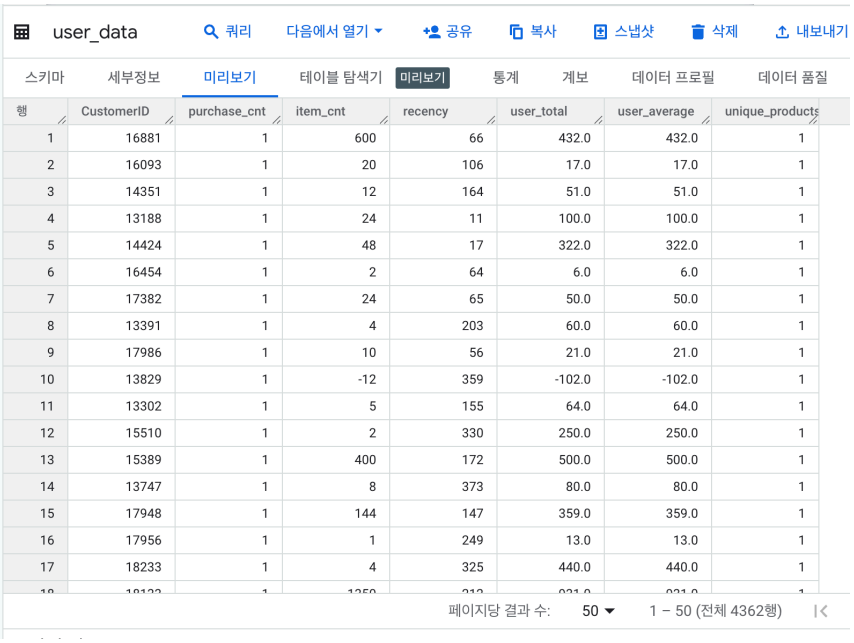
11-8. 추가 Feature 추출

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 concise-isotope-456102-c4.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM concise-isotope-456102-c4.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	16881	1	600	66	432.0	432.0	1
2	16093	1	20	106	17.0	17.0	1
3	14351	1	12	164	51.0	51.0	1
4	13188	1	24	11	100.0	100.0	1
5	14424	1	48	17	322.0	322.0	1
6	16454	1	2	64	6.0	6.0	1
7	17382	1	24	65	50.0	50.0	1
8	13391	1	4	203	60.0	60.0	1
9	17986	1	10	56	21.0	21.0	1
10	13829	1	-12	359	-102.0	-102.0	1
11	13302	1	5	155	64.0	64.0	1
12	15510	1	2	330	250.0	250.0	1
13	15389	1	400	172	500.0	500.0	1
14	13747	1	8	373	80.0	80.0	1
15	17948	1	144	147	359.0	359.0	1
16	17956	1	1	249	13.0	13.0	1
17	18233	1	4	325	440.0	440.0	1

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 concise-isotope-456102-c4.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
    concise-isotope-456102-c4.modulabs_project.data
  WHERE CustomerID IS NOT NULL
 )
 GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM concise-isotope-456102-c4.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	15524	1	4	24	440.0	440.0	1	0.0
2	16138	1	-1	368	-8.0	-8.0	1	0.0
3	13135	1	4300	196	3096.0	3096.0	1	0.0
4	18174	1	50	7	104.0	104.0	1	0.0
5	13391	1	4	203	60.0	60.0	1	0.0
6	15562	1	39	351	135.0	135.0	1	0.0
7	16737	1	288	53	418.0	418.0	1	0.0
8	13185	1	12	267	71.0	71.0	1	0.0
9	17102	1	2	261	26.0	26.0	1	0.0
10	15316	1	100	326	165.0	165.0	1	0.0
11	18068	1	6	289	102.0	102.0	1	0.0
12	16078	1	16	283	79.0	79.0	1	0.0
13	18113	1	72	368	76.0	76.0	1	0.0
14	13747	1	8	373	80.0	80.0	1	0.0
15	16738	1	3	297	4.0	4.0	1	0.0
16	16953	1	10	30	21.0	21.0	1	0.0
17	17986	1	10	56	21.0	21.0	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 concise-isotope-456102-c4.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    SUM(CASE WHEN InvoiceNo LIKE "C%" THEN 1 ELSE 0 END)AS cancel_frequency
  FROM concise-isotope-456102-c4.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency/t.total_transactions, 2) AS cancel_rate
FROM `concise-isotope-456102-c4.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

user_data 쿼리 다음에서 열기 공유 복사 스냅샷 삭제 내보내기 새로고

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactior	cancel_frequenc	cancel_rate
1	17382	1	24	65	50.0	50.0	1	0.0	1	0	0.0
2	18113	1	72	368	76.0	76.0	1	0.0	1	0	0.0
3	16454	1	2	64	6.0	6.0	1	0.0	1	0	0.0
4	18184	1	60	15	50.0	50.0	1	0.0	1	0	0.0
5	15070	1	36	372	106.0	106.0	1	0.0	1	0	0.0
6	16579	1	-12	365	-31.0	-31.0	1	0.0	1	1	1.0
7	12603	1	56	21	613.0	613.0	1	0.0	1	0	0.0
8	15389	1	400	172	500.0	500.0	1	0.0	1	0	0.0
9	14576	1	12	372	35.0	35.0	1	0.0	1	0	0.0
10	18174	1	50	7	104.0	104.0	1	0.0	1	0	0.0
11	16061	1	-1	269	-30.0	-30.0	1	0.0	1	1	1.0
12	17925	1	72	372	244.0	244.0	1	0.0	1	0	0.0
13	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0
14	14351	1	12	164	51.0	51.0	1	0.0	1	0	0.0
15	15118	1	1440	134	245.0	245.0	1	0.0	1	0	0.0
16	16093	1	20	106	17.0	17.0	1	0.0	1	0	0.0
17	15668	1	72	217	76.0	76.0	1	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**

```
# [[YOUR QUERY]];
SELECT *
FROM concise-isotope-456102-c4.modulabs_project.user_data
ORDER BY CustomerID;
```

[결과 이미지를 넣어주세요]

쿼리 결과 결과 저장 다음에서 열기

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_ri
3601	17242	2	611	108	1135.0	567.5	30	1.85	2	0	
3602	17243	30	3308	1	8483.0	282.7666666666...	179	0.92	30	7	
3603	17244	1	616	53	955.0	955.0	31	0.0	1	0	
3604	17245	1	75	204	171.0	171.0	9	0.0	1	0	
3605	17247	1	105	15	280.0	280.0	53	0.0	1	0	
3606	17248	1	213	120	319.0	319.0	68	0.0	1	0	
3607	17250	2	167	3	382.0	191.0	55	4.67	2	0	
3608	17251	1	108	362	270.0	270.0	37	0.0	1	0	
3609	17252	2	192	116	349.0	174.5	86	0.81	2	0	
3610	17253	1	43	18	178.0	178.0	18	0.0	1	0	
3611	17254	2	251	4	271.0	135.5	100	0.09	2	0	
3612	17255	2	227	40	291.0	145.5	69	3.53	2	0	
3613	17256	2	141	107	240.0	120.0	12	13.6	2	0	
3614	17259	2	388	148	581.0	290.5	104	1.91	2	0	
3615	17262	5	367	52	1253.0	250.6	9	28.45	5	0	
3616	17263	1	36	208	63.0	63.0	23	0.0	1	0	
3617	17265	2	389	138	550.0	275.0	106	0.34	2	0	
3618	17266	3	153	2	297.0	99.0	60	2.0	3	0	
3619	17267	2	158	127	317.0	158.5	37	0.0	2	0	

페이지당 결과 수: 50 3601 – 3650 (전체 4362행)

회고

[회고 내용을 작성해주세요]

Keep : 순차적으로 쿼리를 이해하고 넘어가면서 다음 쿼리에 적용할 수 있도록 하는 점

Problem : 이해하고 넘어가려다 보니까 시간이 오래 걸린다..

Try : 조금 더 빠르게 보고 적용할 수 있도록 구문 연습!