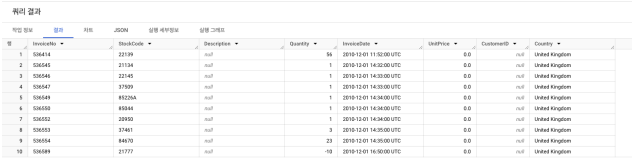
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `upheld-pursuit-439402-b3.modulabs_project.data`
LIMIT 10;
```



- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `upheld-pursuit-439402-b3.modulabs_project.data`;
```

데이터 수 세기

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

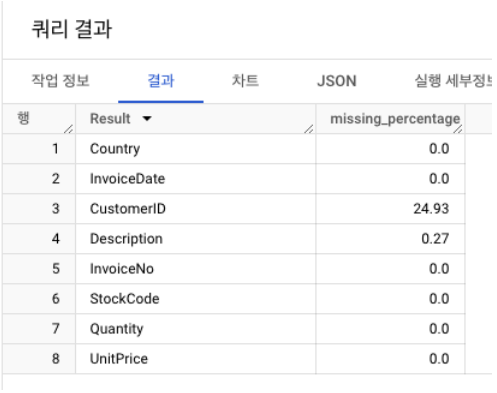
```
SELECT
  COUNT(InvoiceNo) AS COUNT_InvoiceNo,
  COUNT(StockCode) AS COUNT_StockCode,
  COUNT(Description) AS COUNT_Description,
  COUNT(Quantity) AS COUNT_Quantity,
  COUNT(InvoiceDate) AS COUNT_InvoiceDate,
  COUNT(UnitPrice) AS COUNT_UnitPrice,
  COUNT(CustomerID) AS COUNT_CustomerID,
  COUNT(Country) AS COUNT_Country,
FROM `upheld-pursuit-439402-b3.modulabs_project.data`;
```

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 Result,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'StockCode' AS Result,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'Description' AS Result,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'Quantity' AS Result,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percenta
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'InvoiceDate' AS InvoiceDate,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_perce
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'UnitPrice' AS Result,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'CustomerID' AS Result,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percen
FROM `upheld-pursuit-439402-b3.modulabs_project.data`

UNION ALL

SELECT
    'Country' AS Result,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
FROM `upheld-pursuit-439402-b3.modulabs_project.data`;
```

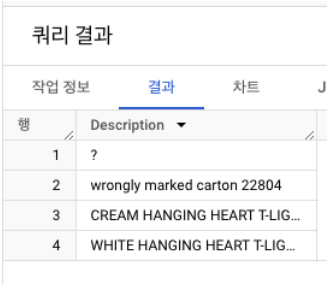


행	Result	missing_percentage
1	Country	0.0
2	InvoiceDate	0.0
3	CustomerID	24.93
4	Description	0.27
5	InvoiceNo	0.0
6	StockCode	0.0
7	Quantity	0.0
8	UnitPrice	0.0

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM `upheld-pursuit-439402-b3.modulabs_project.data`
WHERE StockCode = '85123A';
```



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

결측치 처리

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

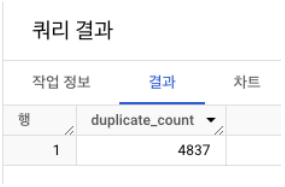
```
DELETE FROM `upheld-pursuit-439402-b3.modulabs_project.data`
WHERE InvoiceNo IS NULL
   OR StockCode IS NULL
   OR Description IS NULL
   OR Quantity IS NULL
   OR InvoiceDate IS NULL
   OR UnitPrice IS NULL
   OR CustomerID IS NULL
   OR Country IS NULL;
```

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

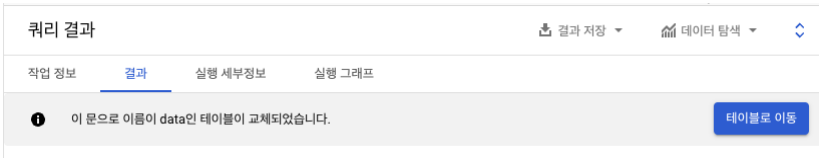
```
WITH duplicates AS (
    SELECT
        InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country,
        COUNT(*) AS count_per_group
    FROM `upheld-pursuit-439402-b3.modulabs_project.data`
    GROUP BY
        InvoiceNo,
        StockCode,
        Description,
        Quantity,
        InvoiceDate,
        UnitPrice,
        CustomerID,
        Country
    HAVING COUNT(*) > 1
)

SELECT COUNT(*) AS duplicate_count
FROM duplicates;
```



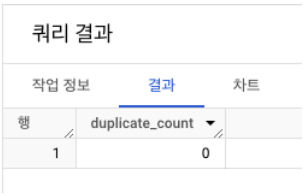
중복값 처리

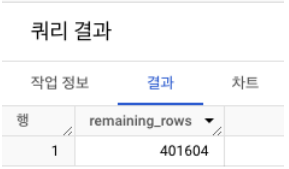
- **중복값을 제거하는 쿼리문 작성하기**
  - **CREATE OR REPLACE TABLE 구문을 활용하여 모든 컬럼(*)을 DISTINCT 한 데이터로 업데이트**

```
CREATE OR REPLACE TABLE `upheld-pursuit-439402-b3.modulabs_project.data` AS
SELECT DISTINCT *
FROM `upheld-pursuit-439402-b3.modulabs_project.data`;
```



  - 중복값 사라짐 확인



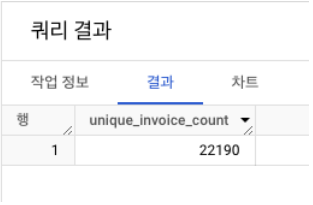


11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

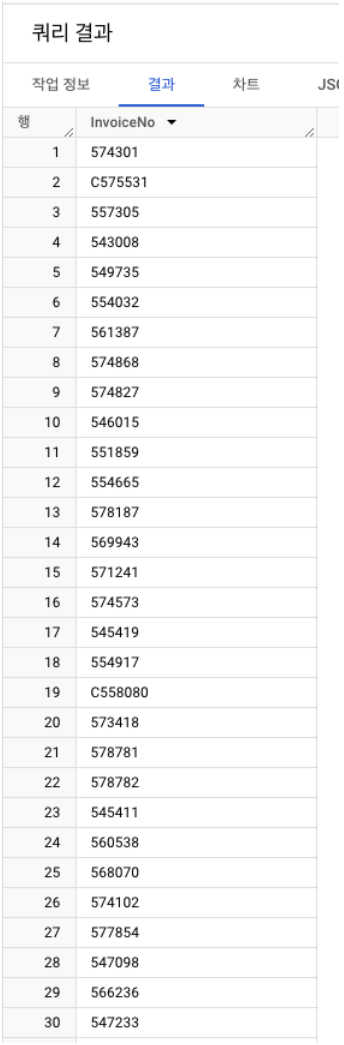
- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```



- **고유한 `InvoiceNo`를 앞에서부터 100개를 출력하기**

```
SELECT DISTINCT InvoiceNo
FROM upheld-pursuit-439402-b3.modulabs_project.data
LIMIT 100;
```



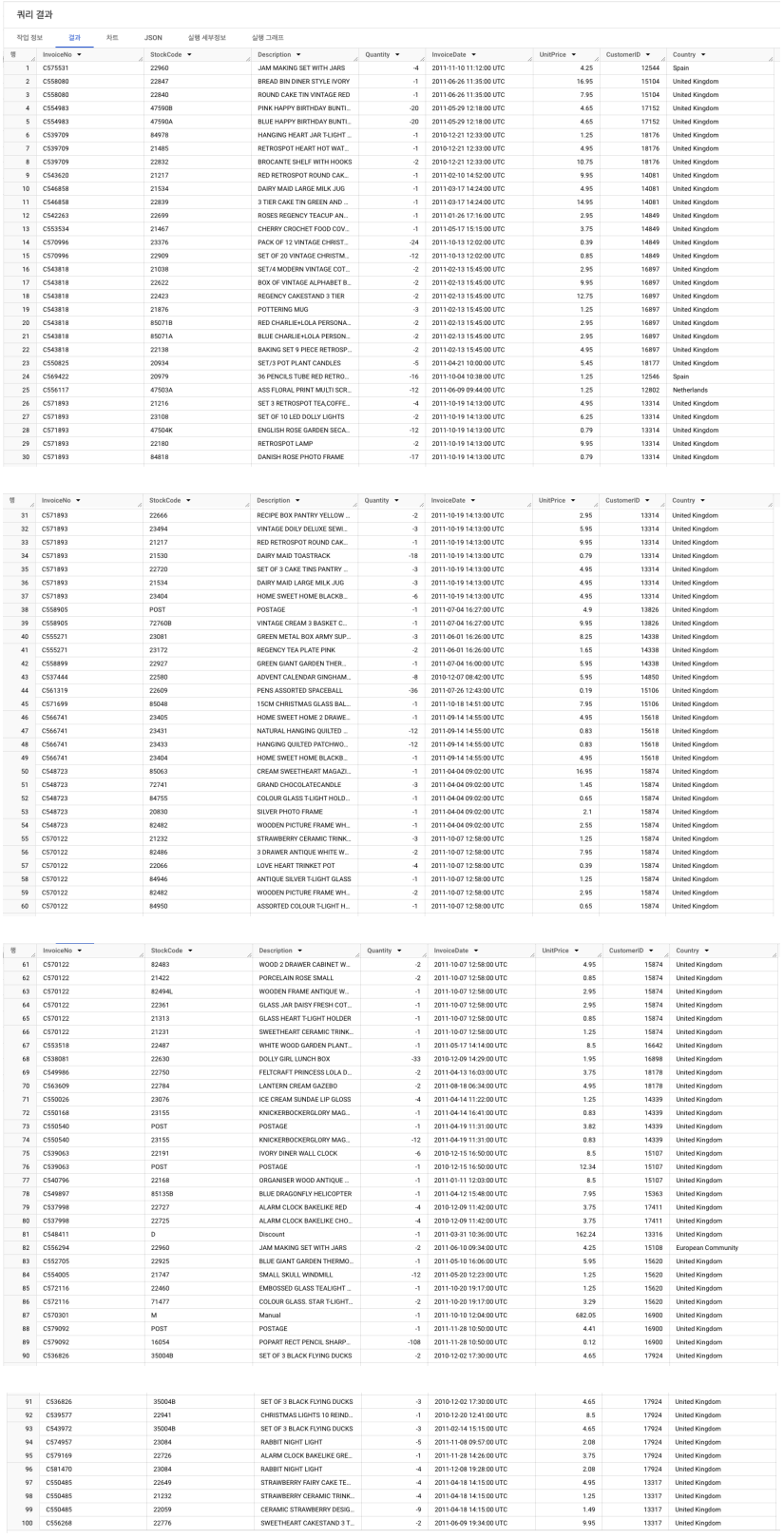
행	InvoiceNo
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868
9	574827
10	546015
11	551859
12	554665
13	578187
14	569943
15	571241
16	574573
17	545419
18	554917
19	C558080
20	573418
21	578781
22	578782
23	545411
24	560538
25	568070
26	574102
27	577854
28	547098
29	566236
30	547233

행	InvoiceNo
31	554982
32	554984
33	C554983
34	558440
35	552835
36	577689
37	536412
38	537435
39	542590
40	550305
41	554337
42	558700
43	560937
44	561381
45	564647
46	567804
47	571546
48	574721
49	575943
50	578818
51	579089
52	580119
53	580672
54	539281
55	C539709
56	547790
57	568179
58	539993
59	552288
60	560915

행	InvoiceNo
61	574393
62	577099
63	537595
64	578017
65	537219
66	543631
67	C543620
68	C546858
69	554603
70	554617
71	577297
72	536460
73	536463
74	536466
75	539142
76	539143
77	542087
78	543987
79	543988
80	548916
81	552614
82	557117
83	557118
84	560918
85	563110
86	563112
87	565610
88	565611
89	569766
90	569767
91	574239
92	574241
93	574245
94	574246
95	577173
96	577175
97	C542263
98	C553534
99	C570996
100	561542

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

```
SELECT *
FROM upheld-pursuit-439402-b3.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```



쿼리 결과

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

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
7	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
8	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
9	C543620	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom
10	C546858	21534	DAIRY MAID LARGE MILK JUG	-1	2011-03-17 14:24:00 UTC	4.95	14081	United Kingdom
11	C546858	22839	3 TIER CAKE TIN GREEN AND ...	-1	2011-03-17 14:24:00 UTC	14.95	14081	United Kingdom
12	C542263	22699	ROSES REGENCY TEACUP AN...	-1	2011-01-26 17:16:00 UTC	2.95	14849	United Kingdom
13	C553534	21467	CHERRY CROCHET FOOD COV...	-1	2011-05-17 15:15:00 UTC	3.75	14849	United Kingdom
14	C570996	23376	PACK OF 12 VINTAGE CHRIST...	-24	2011-10-13 12:02:00 UTC	0.39	14849	United Kingdom
15	C570996	22909	SET OF 20 VINTAGE CHRISTM...	-12	2011-10-13 12:02:00 UTC	0.85	14849	United Kingdom
16	C543818	21038	SET/4 MODERN VINTAGE COT...	-2	2011-02-13 15:45:00 UTC	2.95	16897	United Kingdom
17	C543818	22622	BOX OF VINTAGE ALPHABET B...	-2	2011-02-13 15:45:00 UTC	9.95	16897	United Kingdom
18	C543818	22423	REGENCY CAKESTAND 3 TIER	-2	2011-02-13 15:45:00 UTC	12.75	16897	United Kingdom
19	C543818	21876	POTTERING MUG	-3	2011-02-13 15:45:00 UTC	1.25	16897	United Kingdom
20	C543818	85071B	RED CHARLIE+LOLA PERSONA...	-2	2011-02-13 15:45:00 UTC	2.95	16897	United Kingdom
21	C543818	85071A	BLUE CHARLIE+LOLA PERSON...	-2	2011-02-13 15:45:00 UTC	2.95	16897	United Kingdom
22	C543818	22138	BAKING SET 9 PIECE RETROSP...	-2	2011-02-13 15:45:00 UTC	4.95	16897	United Kingdom
23	C550825	20934	SET/3 POT PLANT CANDLES	-5	2011-04-21 10:00:00 UTC	5.45	18177	United Kingdom
24	C569422	20979	36 PENCILS TUBE RED RETRO...	-16	2011-10-04 10:38:00 UTC	1.25	12546	Spain
25	C556117	47503A	ASS FLORAL PRINT MULTI SCR...	-12	2011-06-09 09:44:00 UTC	1.25	12802	Netherlands
26	C571893	21216	SET 3 RETROSPOT TEA,COFFE...	-4	2011-10-19 14:13:00 UTC	4.95	13314	United Kingdom
27	C571893	23108	SET OF 10 LED DOLLY LIGHTS	-2	2011-10-19 14:13:00 UTC	6.25	13314	United Kingdom
28	C571893	47504K	ENGLISH ROSE GARDEN SECA...	-12	2011-10-19 14:13:00 UTC	0.79	13314	United Kingdom
29	C571893	22180	RETROSPOT LAMP	-2	2011-10-19 14:13:00 UTC	9.95	13314	United Kingdom
30	C571893	84818	DANISH ROSE PHOTO FRAME	-17	2011-10-19 14:13:00 UTC	0.79	13314	United Kingdom

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
31	C571893	22666	RECIPE BOX PANTRY YELLOW ...	-2	2011-10-19 14:13:00 UTC	2.95	13314	United Kingdom
32	C571893	23494	VINTAGE DOILY DELUXE SEWI...	-3	2011-10-19 14:13:00 UTC	5.95	13314	United Kingdom
33	C571893	21217	RED RETROSPOT ROUND CAK...	-1	2011-10-19 14:13:00 UTC	9.95	13314	United Kingdom
34	C571893	21530	DAIRY MAID TOASTRACK	-18	2011-10-19 14:13:00 UTC	0.79	13314	United Kingdom
35	C571893	22720	SET OF 3 CAKE TINS PANTRY ...	-3	2011-10-19 14:13:00 UTC	4.95	13314	United Kingdom
36	C571893	21534	DAIRY MAID LARGE MILK JUG	-3	2011-10-19 14:13:00 UTC	4.95	13314	United Kingdom
37	C571893	23404	HOME SWEET HOME BLACKB...	-6	2011-10-19 14:13:00 UTC	4.95	13314	United Kingdom
38	C558905	POST	POSTAGE	-1	2011-07-04 16:27:00 UTC	4.9	13826	United Kingdom
39	C558905	72760B	VINTAGE CREAM 3 BASKET C...	-1	2011-07-04 16:27:00 UTC	9.95	13826	United Kingdom
40	C555271	23081	GREEN METAL BOX ARMY SUP...	-3	2011-06-01 16:26:00 UTC	8.25	14338	United Kingdom
41	C555271	23172	REGENCY TEA PLATE PINK	-2	2011-06-01 16:26:00 UTC	1.65	14338	United Kingdom
42	C558899	22927	GREEN GIANT GARDEN THER...	-1	2011-07-04 16:00:00 UTC	5.95	14338	United Kingdom
43	C537444	22580	ADVENT CALENDAR GINGHAM...	-8	2010-12-07 08:42:00 UTC	5.95	14850	United Kingdom
44	C561319	22609	PENS ASSORTED SPACEBALL	-36	2011-07-26 12:43:00 UTC	0.19	15106	United Kingdom
45	C571699	85048	15CM CHRISTMAS GLASS BAL...	-1	2011-10-18 14:51:00 UTC	7.95	15106	United Kingdom
46	C566741	23405	HOME SWEET HOME 2 DRAWE...	-1	2011-09-14 14:55:00 UTC	4.95	15618	United Kingdom
47	C566741	23431	NATURAL HANGING QUILTED ...	-12	2011-09-14 14:55:00 UTC	0.83	15618	United Kingdom
48	C566741	23433	HANGING QUILTED PATCHWO...	-12	2011-09-14 14:55:00 UTC	0.83	15618	United Kingdom
49	C566741	23404	HOME SWEET HOME BLACKB...	-1	2011-09-14 14:55:00 UTC	4.95	15618	United Kingdom
50	C548723	85063	CREAM SWEETHEART MAGAZI...	-1	2011-04-04 09:02:00 UTC	16.95	15874	United Kingdom
51	C548723	72741	GRAND CHOCOLATECANDLE	-3	2011-04-04 09:02:00 UTC	1.45	15874	United Kingdom
52	C548723	84755	COLOUR GLASS T-LIGHT HOLD...	-1	2011-04-04 09:02:00 UTC	0.65	15874	United Kingdom
53	C548723	20830	SILVER PHOTO FRAME	-1	2011-04-04 09:02:00 UTC	2.1	15874	United Kingdom
54	C548723	82482	WOODEN PICTURE FRAME WH...	-1	2011-04-04 09:02:00 UTC	2.55	15874	United Kingdom
55	C570122	21232	STRAWBERRY CERAMIC TRINK...	-3	2011-10-07 12:58:00 UTC	1.25	15874	United Kingdom
56	C570122	82486	3 DRAWER ANTIQUE WHITE W...	-2	2011-10-07 12:58:00 UTC	7.95	15874	United Kingdom
57	C570122	22066	LOVE HEART TRINKET POT	-4	2011-10-07 12:58:00 UTC	0.39	15874	United Kingdom
58	C570122	84946	ANTIQUE SILVER T-LIGHT GLASS	-1	2011-10-07 12:58:00 UTC	1.25	15874	United Kingdom
59	C570122	82482	WOODEN PICTURE FRAME WH...	-2	2011-10-07 12:58:00 UTC	2.95	15874	United Kingdom
60	C570122	84950	ASSORTED COLOUR T-LIGHT H...	-1	2011-10-07 12:58:00 UTC	0.65	15874	United Kingdom

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
61	C570122	82483	WOOD 2 DRAWER CABINET W...	-2	2011-10-07 12:58:00 UTC	4.95	15874	United Kingdom
62	C570122	21422	PORCELAIN ROSE SMALL	-2	2011-10-07 12:58:00 UTC	0.85	15874	United Kingdom
63	C570122	82494L	WOODEN FRAME ANTIQUE W...	-1	2011-10-07 12:58:00 UTC	2.95	15874	United Kingdom
64	C570122	22361	GLASS JAR DAISY FRESH COT...	-1	2011-10-07 12:58:00 UTC	2.95	15874	United Kingdom
65	C570122	21313	GLASS HEART T-LIGHT HOLDER	-1	2011-10-07 12:58:00 UTC	0.85	15874	United Kingdom
66	C570122	21231	SWEETHEART CERAMIC TRINK...	-1	2011-10-07 12:58:00 UTC	1.25	15874	United Kingdom
67	C553518	22487	WHITE WOOD GARDEN PLANT...	-1	2011-05-17 14:14:00 UTC	8.5	16642	United Kingdom
68	C538081	22630	DOLLY GIRL LUNCH BOX	-33	2010-12-09 14:29:00 UTC	1.95	16898	United Kingdom
69	C549986	22750	FELTCRAFT PRINCESS LOLA D...	-2	2011-04-13 16:03:00 UTC	3.75	18178	United Kingdom
70	C563609	22784	LANTERN CREAM GAZEBO	-2	2011-08-18 06:34:00 UTC	4.95	18178	United Kingdom
71	C550026	23076	ICE CREAM SUNDAE LIP GLOSS	-4	2011-04-14 11:22:00 UTC	1.25	14339	United Kingdom
72	C550168	23155	KNICKERBOCKERGLORY MAG...	-1	2011-04-14 16:41:00 UTC	0.83	14339	United Kingdom
73	C550540	POST	POSTAGE	-1	2011-04-19 11:31:00 UTC	3.82	14339	United Kingdom
74	C550540	23155	KNICKERBOCKERGLORY MAG...	-12	2011-04-19 11:31:00 UTC	0.83	14339	United Kingdom
75	C539063	22191	IVORY DINER WALL CLOCK	-6	2010-12-15 16:50:00 UTC	8.5	15107	United Kingdom
76	C539063	POST	POSTAGE	-1	2010-12-15 16:50:00 UTC	12.34	15107	United Kingdom
77	C540796	22168	ORGANISER WOOD ANTIQUE ...	-1	2011-01-11 12:03:00 UTC	8.5	15107	United Kingdom
78	C549897	85135B	BLUE DRAGONFLY HELICOPTER	-1	2011-04-12 15:48:00 UTC	7.95	15363	United Kingdom
79	C537998	22727	ALARM CLOCK BAKELIKE RED	-4	2010-12-09 11:42:00 UTC	3.75	17411	United Kingdom
80	C537998	22725	ALARM CLOCK BAKELIKE CHO...	-4	2010-12-09 11:42:00 UTC	3.75	17411	United Kingdom
81	C548411	D	Discount	-1	2011-03-31 10:36:00 UTC	162.24	13316	United Kingdom
82	C556294	22960	JAM MAKING SET WITH JARS	-2	2011-06-10 09:34:00 UTC	4.25	15108	European Community
83	C552705	22925	BLUE GIANT GARDEN THERMO...	-1	2011-05-10 16:06:00 UTC	5.95	15620	United Kingdom
84	C554005	21747	SMALL SKULL WINDMILL	-12	2011-05-20 12:23:00 UTC	1.25	15620	United Kingdom
85	C572116	22460	EMBOSSED GLASS TEALIGHT ...	-1	2011-10-20 19:17:00 UTC	1.25	15620	United Kingdom
86	C572116	71477	COLOUR GLASS. STAR T-LIGHT...	-2	2011-10-20 19:17:00 UTC	3.29	15620	United Kingdom
87	C570301	M	Manual	-1	2011-10-10 12:04:00 UTC	682.05	16900	United Kingdom
88	C579092	POST	POSTAGE	-1	2011-11-28 10:50:00 UTC	4.41	16900	United Kingdom
89	C579092	16054	POPART RECT PENCIL SHARP...	-108	2011-11-28 10:50:00 UTC	0.12	16900	United Kingdom
90	C536826	35004B	SET OF 3 BLACK FLYING DUCKS	-2	2010-12-02 17:30:00 UTC	4.65	17924	United Kingdom

91	C536826	35004B	SET OF 3 BLACK FLYING DUCKS	-3	2010-12-02 17:30:00 UTC	4.65	17924	United Kingdom
92	C539577	22941	CHRISTMAS LIGHTS 10 REIND...	-1	2010-12-20 12:41:00 UTC	8.5	17924	United Kingdom
93	C543972	35004B	SET OF 3 BLACK FLYING DUCKS	-3	2011-02-14 15:15:00 UTC	4.65	17924	United Kingdom
94	C574957	23084	RABBIT NIGHT LIGHT	-5	2011-11-08 09:57:00 UTC	2.08	17924	United Kingdom
95	C579169	22726	ALARM CLOCK BAKELIKE GRE...	-1	2011-11-28 14:26:00 UTC	3.75	17924	United Kingdom
96	C581470	23084	RABBIT NIGHT LIGHT	-4	2011-12-08 19:28:00 UTC	2.08	17924	United Kingdom
97	C550485	22649	STRAWBERRY FAIRY CAKE TE...	-4	2011-04-18 14:15:00 UTC	4.95	13317	United Kingdom
98	C550485	21232	STRAWBERRY CERAMIC TRINK...	-4	2011-04-18 14:15:00 UTC	1.25	13317	United Kingdom
99	C550485	22059	CERAMIC STRAWBERRY DESIG...	-9	2011-04-18 14:15:00 UTC	1.49	13317	United Kingdom
100	C556268	22776	SWEETHEART CAKESTAND 3 T...	-2	2011-06-09 19:34:00 UTC	9.95	13317	United Kingdom

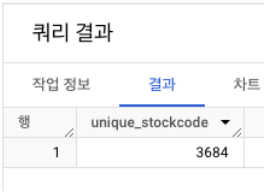
- **구매 건 상태가 `Canceled` 인 데이터의 비율(%) - 소수점 첫번째 자리까지**

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' AND Quantity < 0 THEN 1 ELSE 0 END) / COUNT(*) * 100,
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```

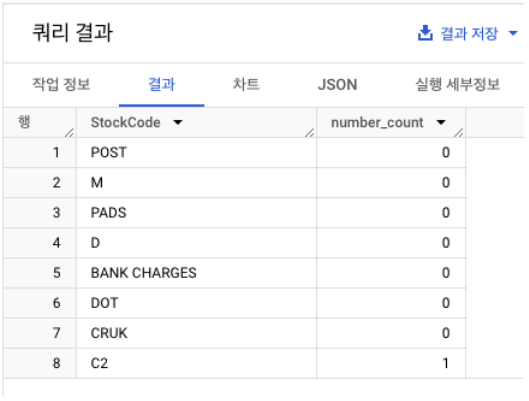


- **어떤 제품이 가장 많이 판매되었는지 보기 위하여 `StockCode` 별 등장 빈도를 출력하기**
  - 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

- **`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

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM upheld-pursuit-439402-b3.modulabs_project.data
) AS stock_counts
WHERE number_count IN (0, 1);
```



쿼리 결과

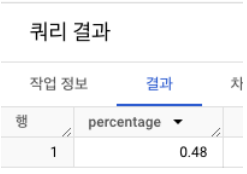
결과 저장

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
WITH StockCodeCounts AS (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM upheld-pursuit-439402-b3.modulabs_project.data
),
FilteredStockCodes AS (
  SELECT *
  FROM StockCodeCounts
  WHERE number_count IN (0, 1)
)

SELECT ROUND(COUNT(*) / (SELECT COUNT(*) FROM upheld-pursuit-439402-b3.modulabs_project.data) * 100,
FROM FilteredStockCodes;
```



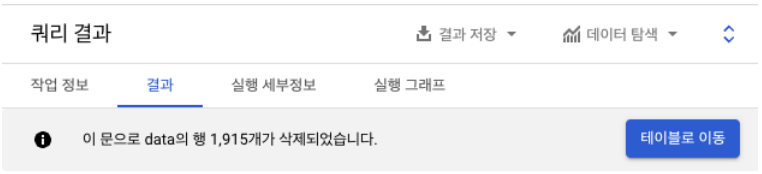
쿼리 결과

작업 정보 | 결과 | 차

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM upheld-pursuit-439402-b3.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')
);
```



`Description` 살펴보기

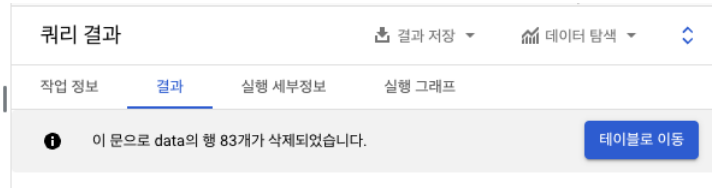
- **고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기**

```
SELECT Description, COUNT(*) AS description_cnt
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

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
18	LUNCH BAG SUKI DESIGN	932
19	ALARM CLOCK BAKELIKE RED	917
20	WOODEN PICTURE FRAME WH...	900
21	REX CASH+CARRY JUMBO SH...	900
22	JUMBO BAG PINK POLKADOT	897
23	LUNCH BAG APPLE DESIGN	890
24	SET OF 4 PANTRY JELLY MOU...	890
25	BAKING SET 9 PIECE RETROSP...	885
26	RECIPE BOX PANTRY YELLOW ...	883
27	JAM MAKING SET PRINTED	883
28	LUNCH BAG WOODLAND	850
29	ROSES REGENCY TEACUP AN...	844
30	VICTORIAN GLASS HANGING T...	843

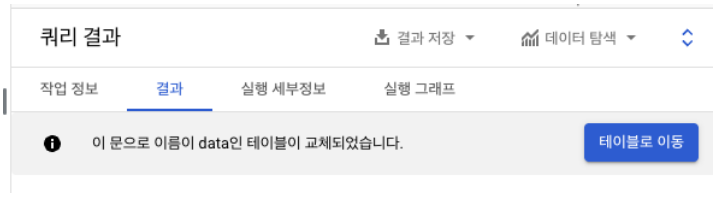
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM upheld-pursuit-439402-b3.modulabs_project.data
WHERE Description IN (
  'Next Day Carriage',
  'High Resolution Image'
);
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

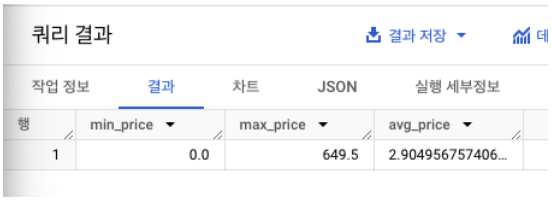
```
CREATE OR REPLACE TABLE upheld-pursuit-439402-b3.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```



`UnitPrice` 살펴보기

- `UnitPrice`**의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```



- **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 upheld-pursuit-439402-b3.modulabs_project.data
WHERE UnitPrice = 0;
```

- `UnitPrice = 0`를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE upheld-pursuit-439402-b3.modulabs_project.data AS
SELECT *
FROM upheld-pursuit-439402-b3.modulabs_project.data
WHERE UnitPrice <> 0;
```

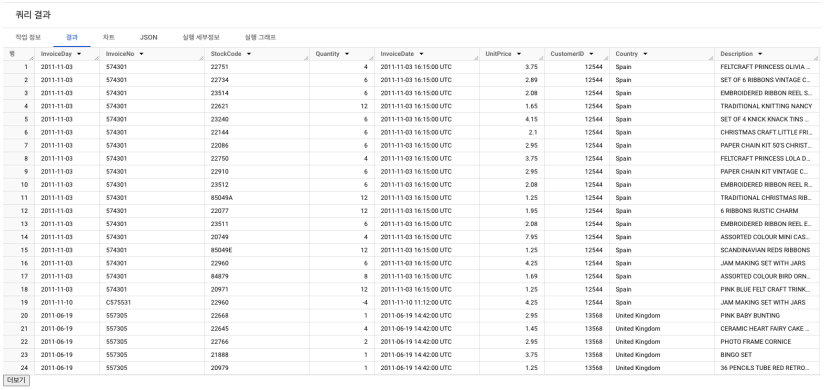
11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```

  - 전체 399573행

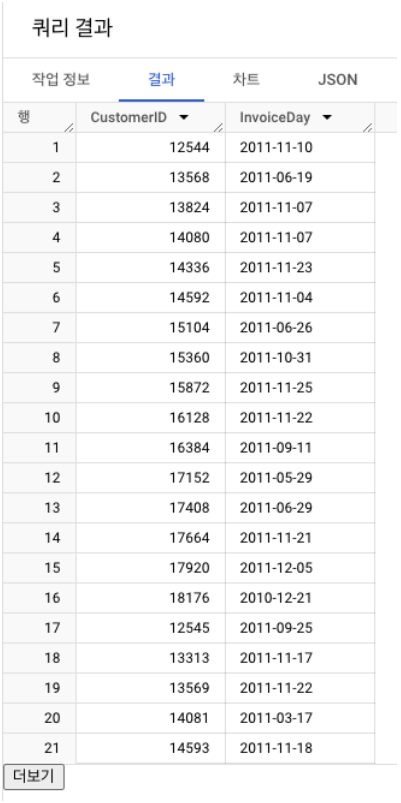


- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
   (SELECT MAX(InvoiceDate) FROM upheld-pursuit-439402-b3.modulabs_project.data) AS most_recent_date
   DATE(InvoiceDate) AS InvoiceDay,
   *
FROM upheld-pursuit-439402-b3.modulabs_project.data;
```

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY CustomerID;
```



쿼리 결과

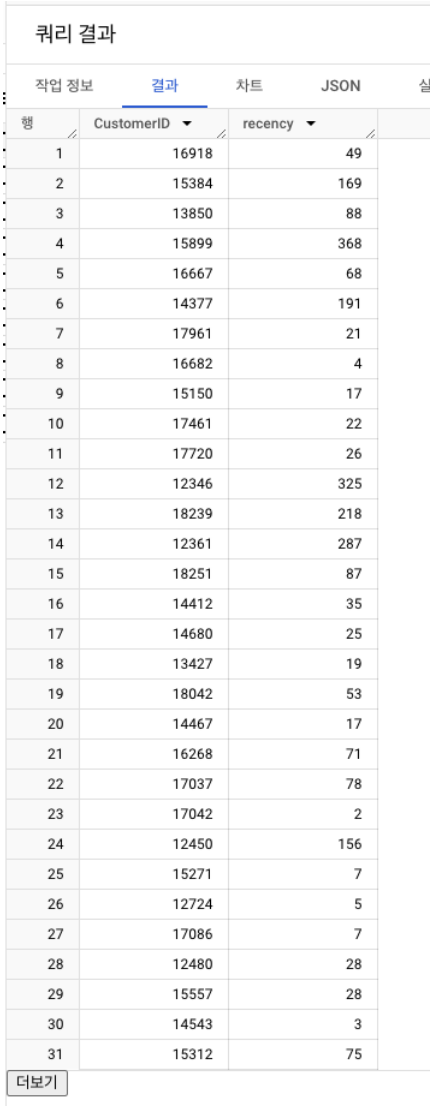
작업 정보 | 결과 | 차트 | JSON

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
10	16128	2011-11-22
11	16384	2011-09-11
12	17152	2011-05-29
13	17408	2011-06-29
14	17664	2011-11-21
15	17920	2011-12-05
16	18176	2010-12-21
17	12545	2011-09-25
18	13313	2011-11-17
19	13569	2011-11-22
20	14081	2011-03-17
21	14593	2011-11-18

더보기

- **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
);
```



쿼리 결과

작업 정보 | 결과 | 차트 | JSON | 실

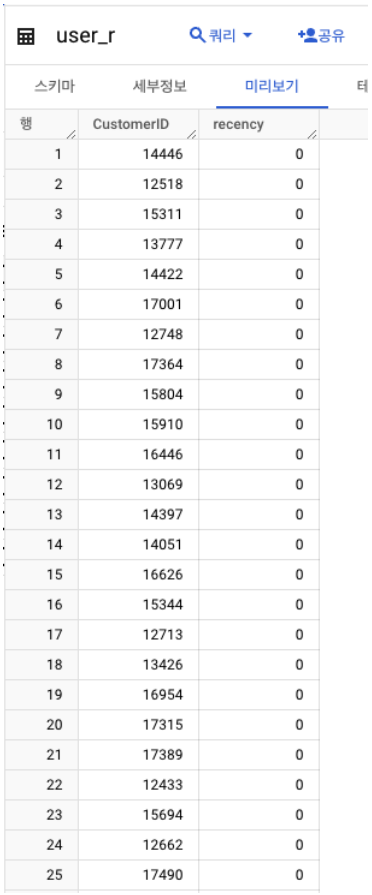
행	CustomerID	recency
1	16918	49
2	15384	169
3	13850	88
4	15899	368
5	16667	68
6	14377	191
7	17961	21
8	16682	4
9	15150	17
10	17461	22
11	17720	26
12	12346	325
13	18239	218
14	12361	287
15	18251	87
16	14412	35
17	14680	25
18	13427	19
19	18042	53
20	14467	17
21	16268	71
22	17037	78
23	17042	2
24	12450	156
25	15271	7
26	12724	5
27	17086	7
28	12480	28
29	15557	28
30	14543	3
31	15312	75

더보기

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

```
CREATE OR REPLACE TABLE upheld-pursuit-439402-b3.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 upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
);
```



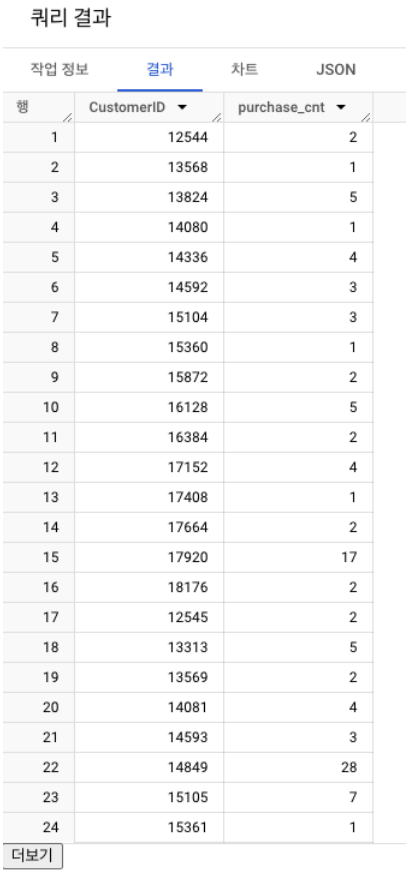
행	CustomerID	recency
1	14446	0
2	12518	0
3	15311	0
4	13777	0
5	14422	0
6	17001	0
7	12748	0
8	17364	0
9	15804	0
10	15910	0
11	16446	0
12	13069	0
13	14397	0
14	14051	0
15	16626	0
16	15344	0
17	12713	0
18	13426	0
19	16954	0
20	17315	0
21	17389	0
22	12433	0
23	15694	0
24	12662	0
25	17490	0

26	12985	0
27	16705	0
28	17581	0
29	16558	0
30	12423	0
31	17428	0
32	12680	0
33	17754	0
34	12526	0
35	13113	0
36	18102	0
37	14441	0
38	16692	256
39	13649	256
40	18010	256
41	13368	256
42	15032	256
43	12792	256
44	15083	256
45	13098	1
46	13510	1
47	13436	1
48	16401	1
49	13263	1
50	17526	1

Frequency

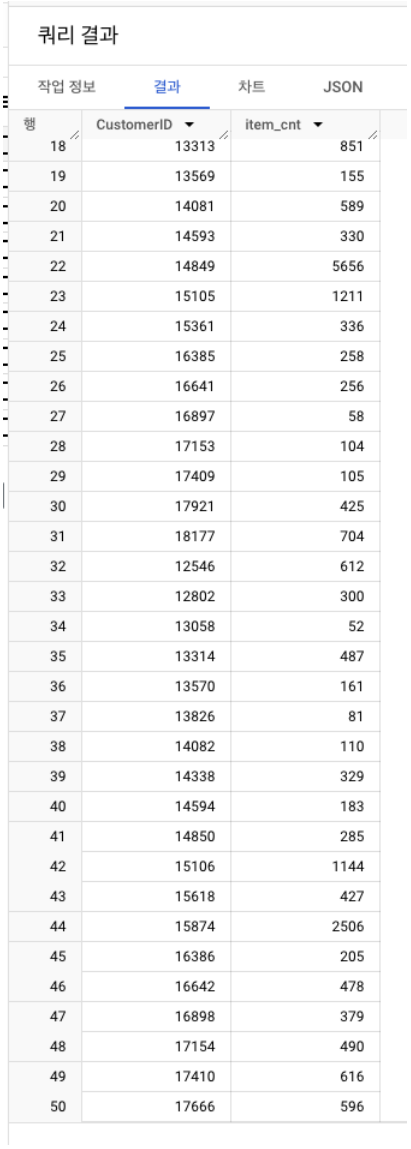
- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY CustomerID;
```



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY CustomerID;
```



행	CustomerID	item_cnt
18	13313	851
19	13569	155
20	14081	589
21	14593	330
22	14849	5656
23	15105	1211
24	15361	336
25	16385	258
26	16641	256
27	16897	58
28	17153	104
29	17409	105
30	17921	425
31	18177	704
32	12546	612
33	12802	300
34	13058	52
35	13314	487
36	13570	161
37	13826	81
38	14082	110
39	14338	329
40	14594	183
41	14850	285
42	15106	1144
43	15618	427
44	15874	2506
45	16386	205
46	16642	478
47	16898	379
48	17154	490
49	17410	616
50	17666	596

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

```
CREATE OR REPLACE TABLE upheld-pursuit-439402-b3.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
)
```

```sql
-- 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 upheld-pursuit-439402-b3.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

user_rf

쿼리 | 공유 | 복사 | 스냅샷

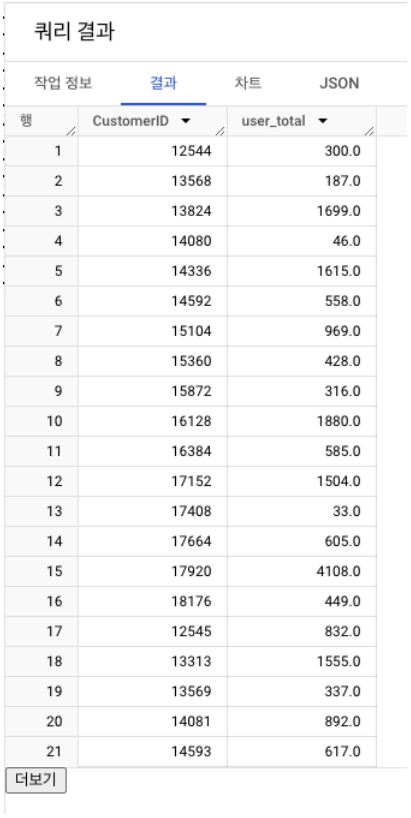
스키마 | 세부정보 | 미리보기 | 테이블 탐색기 | 미리보기 | 통계

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	18010	1	60	256
3	12792	1	215	256
4	15083	1	38	256
5	14569	1	79	1
6	13298	1	96	1
7	15520	1	314	1
8	13436	1	76	1
9	14476	1	110	257
10	13357	1	321	257
11	14204	1	72	2
12	15195	1	1404	2
13	15471	1	256	2
14	12442	1	181	3
15	14578	1	240	3
16	17914	1	457	3
17	16528	1	171	3
18	12650	1	250	3
19	15992	1	17	3
20	15318	1	642	3
21	16569	1	93	3
22	12478	1	233	3
23	14536	1	39	259
24	16597	1	184	4
25	18015	1	157	4
26	13790	1	748	4
27	14219	1	78	4
28	17383	1	148	4
29	15097	1	170	4
30	12367	1	172	4
31	14348	1	183	260
32	15178	1	30	260
33	16063	1	262	260
34	16022	1	255	260
35	16769	1	80	260
36	13512	1	143	260
37	13341	1	301	260

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
FROM upheld-pursuit-439402-b3.modulabs_project.data
GROUP BY CustomerID;
```



쿼리 결과

작업 정보 | 결과 | 차트 | JSON

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
10	16128	1880.0
11	16384	585.0
12	17152	1504.0
13	17408	33.0
14	17664	605.0
15	17920	4108.0
16	18176	449.0
17	12545	832.0
18	13313	1555.0
19	13569	337.0
20	14081	892.0
21	14593	617.0

더보기

- **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 upheld-pursuit-439402-b3.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, 0) AS user_average
FROM upheld-pursuit-439402-b3.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 0) AS user_total
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
```

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

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM upheld-pursuit-439402-b3.modulabs_project.user_rfm;
```

쿼리 결과

작업 정보 | 결과 | 차트 | 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	12792	1	215	256	345.0	345.0
3	18010	1	60	256	175.0	175.0
4	15083	1	38	256	88.0	88.0
5	14569	1	79	1	227.0	227.0
6	15520	1	314	1	344.0	344.0
7	13436	1	76	1	197.0	197.0
8	13298	1	96	1	360.0	360.0
9	14476	1	110	257	193.0	193.0
10	13357	1	321	257	609.0	609.0
11	15195	1	1404	2	3861.0	3861.0
12	15471	1	256	2	454.0	454.0
13	14204	1	72	2	151.0	151.0
14	15992	1	17	3	42.0	42.0
15	17914	1	457	3	329.0	329.0
16	16569	1	93	3	124.0	124.0
17	14578	1	240	3	169.0	169.0
18	12650	1	250	3	242.0	242.0
19	12442	1	181	3	144.0	144.0
20	16528	1	171	3	244.0	244.0
21	12478	1	233	3	546.0	546.0

더보기

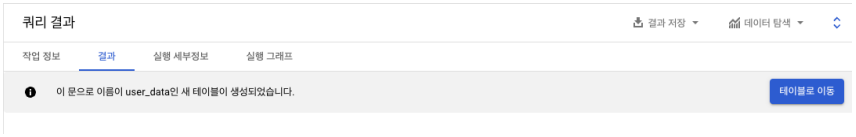
11-8. 추가 Feature 추출

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 upheld-pursuit-439402-b3.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
```

```
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM upheld-pursuit-439402-b3.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



user_data 쿼리 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
1	18233	1	4	325	440.0	440.0	1
2	16257	1	1	176	22.0	22.0	1
3	13366	1	144	50	56.0	56.0	1
4	14119	1	-2	354	-20.0	-20.0	1
5	12603	1	56	21	613.0	613.0	1
6	15524	1	4	24	440.0	440.0	1
7	13841	1	100	252	85.0	85.0	1
8	17291	1	72	308	551.0	551.0	1
9	17956	1	1	249	13.0	13.0	1
10	16148	1	72	296	76.0	76.0	1
11	13391	1	4	203	60.0	60.0	1
12	15488	1	72	92	76.0	76.0	1
13	17443	1	504	219	534.0	534.0	1
14	13120	1	12	238	31.0	31.0	1
15	17923	1	50	282	208.0	208.0	1
16	15657	1	24	22	30.0	30.0	1
17	16061	1	-1	269	-30.0	-30.0	1
18	13099	1	288	99	207.0	207.0	1
19	15753	1	144	304	79.0	79.0	1
20	18113	1	72	368	76.0	76.0	1
21	15195	1	1404	2	3861.0	3861.0	1
22	14705	1	100	198	179.0	179.0	1
23	14576	1	12	372	35.0	35.0	1
24	12814	1	48	101	86.0	86.0	1
25	16144	1	16	246	175.0	175.0	1
26	13307	1	4	120	15.0	15.0	1
27	17763	1	12	263	15.0	15.0	1
28	13185	1	12	267	71.0	71.0	1
29	16953	1	10	30	21.0	21.0	1
30	16737	1	288	53	418.0	418.0	1
31	17925	1	72	372	244.0	244.0	1
32	15389	1	400	172	500.0	500.0	1
33	14679	1	-1	371	-3.0	-3.0	1
34	15562	1	39	351	135.0	135.0	1
35	17752	1	192	359	81.0	81.0	1
36	14351	1	12	164	51.0	51.0	1
37	16138	1	-1	368	-8.0	-8.0	1

38	17331	1	16	123	175.0	175.0	1
39	16765	1	4	294	34.0	34.0	1
40	15940	1	4	311	36.0	36.0	1
41	16738	1	3	297	4.0	4.0	1
42	13188	1	24	11	100.0	100.0	1
43	15316	1	100	326	165.0	165.0	1
44	17986	1	10	56	21.0	21.0	1
45	15313	1	25	110	52.0	52.0	1
46	18133	1	1350	212	931.0	931.0	1
47	13829	1	-12	359	-102.0	-102.0	1
48	16428	1	-1	81	-3.0	-3.0	1
49	15668	1	72	217	76.0	76.0	1
50	16323	1	50	196	208.0	208.0	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 upheld-pursuit-439402-b3.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
      upheld-pursuit-439402-b3.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM upheld-pursuit-439402-b3.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

user_data 쿼리 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	14432	6	2013	9	2248.0	375.0	256	0.2
2	12428	11	3477	25	6366.0	579.0	256	0.87
3	13268	14	3525	17	3106.0	222.0	256	0.56
4	14119	1	-2	354	-20.0	-20.0	1	0.0
5	16953	1	10	30	21.0	21.0	1	0.0
6	13302	1	5	155	64.0	64.0	1	0.0
7	15657	1	24	22	30.0	30.0	1	0.0
8	15195	1	1404	2	3861.0	3861.0	1	0.0
9	13135	1	4300	196	3096.0	3096.0	1	0.0
10	13307	1	4	120	15.0	15.0	1	0.0
11	17925	1	72	372	244.0	244.0	1	0.0
12	17986	1	10	56	21.0	21.0	1	0.0
13	17948	1	144	147	359.0	359.0	1	0.0
14	16881	1	600	66	432.0	432.0	1	0.0
15	15488	1	72	92	76.0	76.0	1	0.0
16	16737	1	288	53	418.0	418.0	1	0.0
17	15070	1	36	372	106.0	106.0	1	0.0
18	17347	1	216	86	229.0	229.0	1	0.0
19	14090	1	72	324	76.0	76.0	1	0.0
20	13829	1	-12	359	-102.0	-102.0	1	0.0
21	17715	1	384	200	326.0	326.0	1	0.0
22	14424	1	48	17	322.0	322.0	1	0.0
23	17443	1	504	219	534.0	534.0	1	0.0
24	16144	1	16	246	175.0	175.0	1	0.0
25	13747	1	8	373	80.0	80.0	1	0.0
26	16093	1	20	106	17.0	17.0	1	0.0
27	18141	1	-12	360	-35.0	-35.0	1	0.0
28	16257	1	1	176	22.0	22.0	1	0.0
29	15940	1	4	311	36.0	36.0	1	0.0
30	16428	1	-1	81	-3.0	-3.0	1	0.0
31	16148	1	72	296	76.0	76.0	1	0.0
32	18113	1	72	368	76.0	76.0	1	0.0
33	15562	1	39	351	135.0	135.0	1	0.0
34	15118	1	1440	134	245.0	245.0	1	0.0
35	18184	1	60	15	50.0	50.0	1	0.0
36	12791	1	96	373	178.0	178.0	1	0.0
37	17307	1	-144	365	-153.0	-153.0	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 upheld-pursuit-439402-b3.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNTIF(LEFT(InvoiceNo, 1) = 'C' AND Quantity < 0) AS cancel_frequency
  FROM upheld-pursuit-439402-b3.modulabs_project.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID),
ROUND(t.cancel_frequency / t.total_transactions ,2) AS cancel_rate
FROM upheld-pursuit-439402-b3.modulabs_project.user_data AS u
```

```
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

user_data

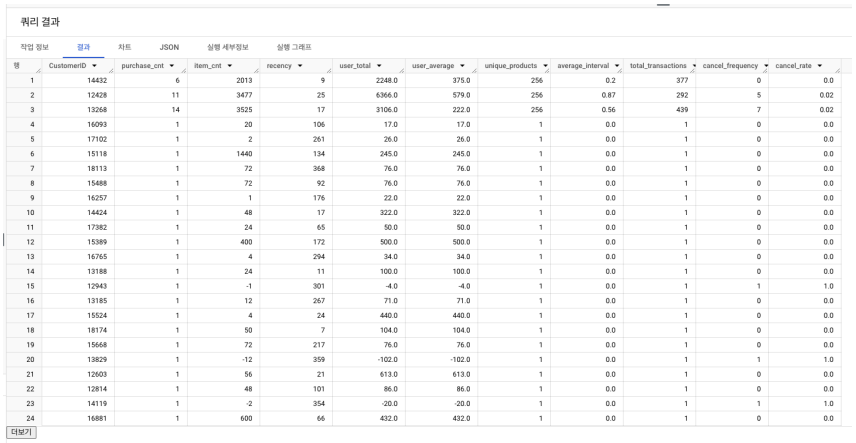
쿼리 | 공유 | 복사 | 스냅샷 | 삭제 | 내보내기

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactio	cancel_frequenc	cancel_rate
1	14432	6	2013	9	2248.0	375.0	256	0.2	377	0	0.0
2	12428	11	3477	25	6366.0	579.0	256	0.87	292	5	0.02
3	13268	14	3525	17	3106.0	222.0	256	0.56	439	7	0.02
4	16093	1	20	106	17.0	17.0	1	0.0	1	0	0.0
5	17102	1	2	261	26.0	26.0	1	0.0	1	0	0.0
6	15118	1	1440	134	245.0	245.0	1	0.0	1	0	0.0
7	18113	1	72	368	76.0	76.0	1	0.0	1	0	0.0
8	15488	1	72	92	76.0	76.0	1	0.0	1	0	0.0
9	16257	1	1	176	22.0	22.0	1	0.0	1	0	0.0
10	14424	1	48	17	322.0	322.0	1	0.0	1	0	0.0
11	17382	1	24	65	50.0	50.0	1	0.0	1	0	0.0
12	15389	1	400	172	500.0	500.0	1	0.0	1	0	0.0
13	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
14	13188	1	24	11	100.0	100.0	1	0.0	1	0	0.0
15	12943	1	-1	301	-4.0	-4.0	1	0.0	1	1	1.0
16	13185	1	12	267	71.0	71.0	1	0.0	1	0	0.0
17	15524	1	4	24	440.0	440.0	1	0.0	1	0	0.0
18	18174	1	50	7	104.0	104.0	1	0.0	1	0	0.0
19	15668	1	72	217	76.0	76.0	1	0.0	1	0	0.0
20	13829	1	-12	359	-102.0	-102.0	1	0.0	1	1	1.0
21	12603	1	56	21	613.0	613.0	1	0.0	1	0	0.0
22	12814	1	48	101	86.0	86.0	1	0.0	1	0	0.0
23	14119	1	-2	354	-20.0	-20.0	1	0.0	1	1	1.0
24	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0
25	18141	1	-12	360	-35.0	-35.0	1	0.0	1	1	1.0
26	18233	1	4	325	440.0	440.0	1	0.0	1	0	0.0
27	17923	1	50	282	208.0	208.0	1	0.0	1	0	0.0
28	16144	1	16	246	175.0	175.0	1	0.0	1	0	0.0
29	15313	1	25	110	52.0	52.0	1	0.0	1	0	0.0
30	17948	1	144	147	359.0	359.0	1	0.0	1	0	0.0
31	17986	1	10	56	21.0	21.0	1	0.0	1	0	0.0
32	13120	1	12	238	31.0	31.0	1	0.0	1	0	0.0
33	16148	1	72	296	76.0	76.0	1	0.0	1	0	0.0
34	15316	1	100	326	165.0	165.0	1	0.0	1	0	0.0
35	13841	1	100	252	85.0	85.0	1	0.0	1	0	0.0
36	16078	1	16	283	79.0	79.0	1	0.0	1	0	0.0
37	17443	1	504	219	534.0	534.0	1	0.0	1	0	0.0

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

```
SELECT *
FROM upheld-pursuit-439402-b3.modulabs_project.user_data;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	14432	6	2013	9	2248.0	375.0	256	0.2	377	0	0.0
2	12428	11	3477	25	6366.0	579.0	256	0.87	292	5	0.02
3	13268	14	3525	17	3106.0	222.0	256	0.56	439	7	0.02
4	16093	1	20	106	17.0	17.0	1	0.0	1	0	0.0
5	17102	1	2	261	26.0	26.0	1	0.0	1	0	0.0
6	15118	1	1440	134	245.0	245.0	1	0.0	1	0	0.0
7	18113	1	72	368	76.0	76.0	1	0.0	1	0	0.0
8	15488	1	72	92	76.0	76.0	1	0.0	1	0	0.0
9	16257	1	1	176	22.0	22.0	1	0.0	1	0	0.0
10	14424	1	48	17	322.0	322.0	1	0.0	1	0	0.0
11	17382	1	24	65	50.0	50.0	1	0.0	1	0	0.0
12	15389	1	400	172	500.0	500.0	1	0.0	1	0	0.0
13	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
14	13188	1	24	11	100.0	100.0	1	0.0	1	0	0.0
15	12943	1	-1	301	-4.0	-4.0	1	0.0	1	1	1.0
16	13185	1	12	267	71.0	71.0	1	0.0	1	0	0.0
17	15524	1	4	24	440.0	440.0	1	0.0	1	0	0.0
18	18174	1	50	7	104.0	104.0	1	0.0	1	0	0.0
19	15668	1	72	217	76.0	76.0	1	0.0	1	0	0.0
20	13829	1	-12	359	-102.0	-102.0	1	0.0	1	1	1.0
21	12603	1	56	21	613.0	613.0	1	0.0	1	0	0.0
22	12814	1	48	101	86.0	86.0	1	0.0	1	0	0.0
23	14119	1	-2	354	-20.0	-20.0	1	0.0	1	1	1.0
24	16881	1	600	66	432.0	432.0	1	0.0	1	0	0.0

- 컬럼이 많아 실제로 계산이 잘 되었는지 확인하기 위해 LMS 결과 예시의 특정 컬럼을 조회해 잘 나오는지 확인! (CustomerID가 13185)

```
SELECT *
FROM upheld-pursuit-439402-b3.modulabs_project.user_data
WHERE CustomerID = 13185;
```

쿼리 결과

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_rate
1	13185	1	12	267	71.0	71.0	1	0.0	1	0	0.0

회고

생각만큼 쉽지는 않았지만 기본으로 제공되는 쿼리문이 있어 내용을 채우는 방식으로 진행할 수 있어서 수학 문제를 푸는 것 같아 정말 재미있었습니다😀

먼저, 11-2에서는 CSV 내부에 있는 데이터가 무엇이 있는지, 데이터 타입은 무엇인지, 몇개의 컬럼과 행으로 이루어져 있는지에 대해서 먼저 파악하는 연습을 하면서 데이터를 어떻게 전처리해야할지 가이드를 세울 수 있었어요.

11-4에서는 본격적인 전치리 과정에 진입하면서 각 컬럼에 따라 누락 비율을 계산하는 작업을 진행했는데 특히, 결측치 비율 계산 시에 기존의 CASE WHEN을 사용할 경우 컬럼 1개당 1개의 SELECT문을 사용해야 하는 불편함이 있었습니다.
하나의 SELECT 내부에서 해결할 수 있지 않을까?하고 고민하려던 차에 ROUND를 이용해서 전체에서 column_value를 빼고 그 값을 `total * 100` 과 나누면 된다는 것을 예시를 통해 바로 알 수 있어 흥미롭고 유익했습니다!

11-5에서의 중복값 처리는 예상외로 간단하게 DISTINCT를 통해 바로 진행할 수 있다는 점이 신기했어요.

그 다음 11-6에서는 "오류값을 처리"하는 과정이 진행되었는데, 모든 컬럼이 공통적으로 유니크한 값을 찾아내고 각 컬럼의 특이점과 경향성을 분석해서 조건을 스스로 만들어갈 수 있는 부분이 정말 재미있게 느껴졌던 것 같습니다.

11-7은 RFM 스코어 계산 파트였고, 차례대로 Recency, Frequency, Monetary를 계산해 user_rfm 테이블에 저장하는 간단한 과정이어서 큰 무리없이 진행할 수 있었어요..!

그리고 마지막으로 11-8을 시작하여 이번 노드의 최종 목표이자 다음 노드를 위한 이전 작업인 "구매하는 제품의 다양성", "평균 구매 주기", "구매 취소 경향성" feature를 추출하는 작업을 진행했습니다.
어려운 내용은 없었으나, 마지막의 구매 취소 경향성 부분의 코드에서 모든 거래를 뜻하는 부분을 카운트할 때에 중복을 제거한 Customer ID를 기준으로 세니 잘못된 결과가 나와서 한참을 고민했었어요.
그런데 알고보니 아주 간단하게 COUNT()을 하면 되는 문제였다는 걸 뒤늦게 깨달았습니다 하하..
그래서 시간은 조금 더 소요되었지만! 결국 원하는 결과가 도출되었어요.

쿼리문을 주어진 조건에 따라 작성하는 것은 어렵지 않게 느껴졌지만, 아직은 용어들이 낯설게만 느껴지는 건 아직 어쩔 수 없는 부분인 것 같습니다.
이번 노드를 진행하면서 학부생 시절 공부했던 데이터 전처리 및 가공 방법에 대해서 다시 복기도 하고, 더 새롭고 딥한 내용들도 학습할 수 있어 정말 유익했던 것 같아요.
다만, 쿼리문을 조금 더 간결하게 쓰고 통합해서 쓰는 연습이 필요하다고 절실히 느꼈습니다😂
길어지면 저조차도 무엇을 의도하고 그 긴 쿼리문을 작성했는지 잊게되고, 어디에서 오류가 발생했는지도 찾기가 어렵다는 걸 이번 노드를 통해서 너무나 크게 깨달았어요.
앞으로도 더 열심히 공부해보겠습니다. 화이팅!