👥

고객을 세그먼테이션하자 [프로젝트] - 배유나 (2)

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
LIMIT 10
```

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010. 12. 1 오전 8:26:00	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	2010. 12. 1 오전 8:26:00	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010. 12. 1 오전 8:26:00	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010. 12. 1 오전 8:26:00	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010. 12. 1 오전 8:26:00	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010. 12. 1 오전 8:26:00	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010. 12. 1 오전 8:26:00	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	2010. 12. 1 오전 8:28:00	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	2010. 12. 1 오전 8:28:00	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010. 12. 1 오전 8:34:00	1.69	13047	United Kingdom

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행   f0_
1    541909
```

데이터 수 세기

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

```
SELECT column_name
FROM `cobalt-sector-479904-d9.modulabs_project_3.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'data';
-- 컬럼 총 8개 확인
SELECT
  COUNT(InvoiceNo) AS InvoiceNo_count,
  COUNT(StockCode) AS StockCode_count,
  COUNT(Description) AS Description_count,
  COUNT(Quantity) AS Quantity_count,
  COUNT(InvoiceDate) AS InvoiceDate_count,
  COUNT(UnitPrice) AS UnitPrice_count,
  COUNT(CustomerID) AS CustomerID_count,
  COUNT(Country) AS Country_count
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

InvoiceNo_count	StockCode_count	Description_count	Quantity_count	InvoiceDate_count	UnitPrice_count	CustomerID_count	Country_count
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**

```
-- InvoiceNo
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- StockCode
SELECT
  'StockCode' AS column_name,
  ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- Description
SELECT
  'Description' AS column_name,
  ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- Quantity
SELECT
  'Quantity' AS column_name,
  ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- InvoiceDate
SELECT
  'InvoiceDate' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- UnitPrice
SELECT
  'UnitPrice' AS column_name,
  ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
```

```
-- CustomerID
SELECT
  'CustomerID' AS column_name,
  ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

UNION ALL
-- Country
SELECT
  'Country' AS column_name,
  ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

column_name	missing_percentage
InvoiceNo	0
StockCode	0
Description	0.27
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	24.93
Country	0

결측치 처리 전략

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

```
SELECT Description
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE StockCode = '85123A';

--중복처리

SELECT DISTINCT Description
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE StockCode = '85123A';
```

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

결측치 처리

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

```
--결측치 삭제를 위해 삭제 필요한 컬럼 속성 확인
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE CustomerID IS NULL
  AND Description IS NULL;

--CustomerID와 Description 의 NULL이 있는 결측치 행 삭제
DELETE FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
```

```
WHERE CustomerID IS NULL
  AND Description IS NULL;
```

이 문으로 data의 행 1,454개가 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**

```
WITH grouped AS (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS cnt
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
)
SELECT COUNT(*) AS num_duplicate_groups
FROM grouped
WHERE cnt > 1;
```

```
행   num_duplicate_groups
1    4879
```

중복값 처리

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

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.data` AS
SELECT DISTINCT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

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 `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행   unique_invoice_count
1    24446
```

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

```
SELECT InvoiceNo
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
LIMIT 100;
```

```
행   InvoiceNo
1    536544
2    536544
3    536544
4    536544
5    536544
6    536544
7    536544
8    536544
9    536544
10   536544
11   536544
12   536544
13   536544
14   536544
15   536544
16   536544
17   536544
18   536544
19   536544
20   536544
21   536544
22   536544
23   536544
24   536544
25   536544
26   536544
27   536544
28   536544
29   536544
30   536544
31   536544
32   536544
33   536544
34   536544
35   536544
36   536544
37   536544
38   536544
39   536544
40   536544
41   536544
```

```
42   536544
43   536544
44   536544
45   536544
46   536544
47   536544
48   536544
49   536544
50   536544
51   536544
52   536544
53   536544
54   536544
55   536544
56   536544
57   536544
58   536544
59   536544
60   536544
61   536544
62   536544
63   536544
64   536544
65   536544
66   536544
67   536544
68   536544
69   536544
70   536544
71   536544
72   536544
73   536544
74   536544
75   536544
76   536544
77   536544
78   536544
79   536544
80   536544
81   536544
82   536544
83   536544
84   536544
85   536544
86   536544
87   536544
88   536544
89   536544
90   536544
91   536544
92   536544
93   536544
94   536544
95   536544
96   536544
97   536544
98   536544
99   536544
100  536544
```

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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE `cobalt-sector-479904-d9.modulabs_project_3.data`
LIMIT 100;
```

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
C537251	21890	S/6 WOODEN SKITTLES IN COTTON BAG	-2	2010. 12. 6 오전 10:45:00	2.95		United Kingdom
C537251	84347	ROTATING SILVER ANGELS T-LIGHT HLDR	-9	2010. 12. 6 오전 10:45:00	2.55		United Kingdom
C537251	22940	FELTCRAFT CHRISTMAS FAIRY	-5	2010. 12. 6 오전 10:45:00	4.25		United Kingdom
C537251	22454	MEASURING TAPE BABUSHKA RED	-8	2010. 12. 6 오전 10:45:00	2.95		United Kingdom
C537251	22141	CHRISTMAS CRAFT TREE TOP ANGEL	-8	2010. 12. 6 오전 10:45:00	2.1		United Kingdom
C537251	22748	POPPY'S PLAYHOUSE KITCHEN	-4	2010. 12. 6 오전 10:45:00	2.1		United Kingdom
C537251	22429	ENAMEL MEASURING JUG CREAM	-2	2010. 12. 6 오전 10:45:00	4.25		United Kingdom
C537251	22564	ALPHABET STENCIL CRAFT	-5	2010. 12. 6 오전 10:45:00	1.25		United Kingdom
C537251	22536	MAGIC DRAWING SLATE PURDEY	-11	2010. 12. 6 오전 10:45:00	0.42		United Kingdom
C537251	21891	TRADITIONAL WOODEN SKIPPING ROPE	-3	2010. 12. 6 오전 10:45:00	1.25		United Kingdom
C537251	22943	CHRISTMAS LIGHTS 10 VINTAGE BAUBLES	-6	2010. 12. 6 오전 10:45:00	4.95		United Kingdom
C537251	22327	ROUND SNACK BOXES SET OF 4 SKULLS	-4	2010. 12. 6 오전 10:45:00	2.95		United Kingdom
C537251	22747	POPPY'S PLAYHOUSE BATHROOM	-6	2010. 12. 6 오전 10:45:00	2.1		United Kingdom
C537251	22911	PAPER CHAIN KIT LONDON	-2	2010. 12. 6 오전 10:45:00	2.95		United Kingdom
C537251	21915	RED HARMONICA IN BOX	-4	2010. 12. 6 오전 10:45:00	1.25		United Kingdom
C537251	22620	4 TRADITIONAL SPINNING TOPS	-8	2010. 12. 6 오전 10:45:00	1.25		United Kingdom
C537251	21826	EIGHT PIECE DINOSAUR SET	-4	2010. 12. 6 오전 10:45:00	1.25		United Kingdom
C537251	22418	10 COLOUR SPACEBOY	-7	2010. 12. 6 오전 10:45:00	0.85		United Kingdom

		PEN					
C537251	22328	ROUND SNACK BOXES SET OF 4 FRUITS	-2	2010. 12. 6 오전 10:45:00	2.95		United Kingdom
C537251	22466	FAIRY TALE COTTAGE NIGHTLIGHT	-3	2010. 12. 6 오전 10:45:00	1.95		United Kingdom
C537572	BANK CHARGES	Bank Charges	-1	2010. 12. 7 오후 12:00:00	95.38		United Kingdom
C537581	S	SAMPLES	-1	2010. 12. 7 오후 12:03:00	52		United Kingdom
C537581	S	SAMPLES	-1	2010. 12. 7 오후 12:03:00	12.95		United Kingdom
C537600	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 12:41:00	1		United Kingdom
C537610	M	Manual	-1	2010. 12. 7 오후 1:23:00	631.31		United Kingdom
C537613	M	Manual	-1	2010. 12. 7 오후 1:28:00	313.78		United Kingdom
C537630	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 3:04:00	13541.33		United Kingdom
C537644	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 3:34:00	13474.79		United Kingdom
C537647	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 3:41:00	5519.25		United Kingdom
C537651	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 3:49:00	13541.33		United Kingdom
C537652	AMAZONFEE	AMAZON FEE	-1	2010. 12. 7 오후 3:51:00	6706.71		United Kingdom
C538189	M	Manual	-1	2010. 12. 10 오전 10:35:00	133.08		United Kingdom
C538680	BANK CHARGES	Bank Charges	-1	2010. 12. 13 오후 5:10:00	966.92		United Kingdom
C538681	M	Manual	-1	2010. 12. 13 오후 5:12:00	316.3		United Kingdom
C538682	M	Manual	-1	2010. 12. 13 오후 5:14:00	1130.9		United Kingdom
C538686	22467	GUMBALL COAT RACK	-1	2010. 12. 14 오전 9:49:00	2.55		United Kingdom
C539756	22720	SET OF 3 CAKE TINS PANTRY DESIGN	-1	2010. 12. 21 오후 4:31:00	4.95		United Kingdom
C539948	21888	BINGO SET	-4	2010. 12. 23 오전 11:48:00	3.75		EIRE
C540117	AMAZONFEE	AMAZON FEE	-1	2011. 1. 5 오전 9:55:00	16888.02		United Kingdom
C540118	AMAZONFEE	AMAZON FEE	-1	2011. 1. 5 오전 9:57:00	16453.71		United Kingdom
C540155	72802B	OCEAN SCENT CANDLE IN JEWELLED BOX	-54	2011. 1. 5 오전 11:31:00	3.81		Bahrain
C540266	POST	POSTAGE	-1	2011. 1. 6 오전 11:05:00	35.09		United Kingdom
C540266	M	Manual	-1	2011. 1. 6 오전 11:05:00	458.29		United Kingdom
C540559	21888	BINGO SET	-4	2011. 1. 10 오전 10:07:00	3.75		EIRE
C540854	22461	SAVOY ART DECO CLOCK	-1	2011. 1. 12 오전 9:54:00	12.75		United Kingdom
C541492	84870B	BLUE GEISHA GIRL	-1	2011. 1. 18 오후 2:24:00	3.75		United Kingdom

C541492	85040A	S/4 PINK FLOWER CANDLES IN BOWL	-1	2011. 1. 18 오후 2:24:00	1.65		United Kingdom
C541492	37500	TEA TIME TEAPOT IN GIFT BOX	-1	2011. 1. 18 오후 2:24:00	9.95		United Kingdom
C541492	85169D	PINK LOVE BIRD CANDLE	-1	2011. 1. 18 오후 2:24:00	1.25		United Kingdom
C541492	85174	S/4 CACTI CANDLES	-1	2011. 1. 18 오후 2:24:00	4.95		United Kingdom
C541492	85039A	SET/4 RED MINI ROSE CANDLE IN BOWL	-1	2011. 1. 18 오후 2:24:00	1.65		United Kingdom
C541492	22511	RETROSPOT BABUSHKA DOORSTOP	-1	2011. 1. 18 오후 2:24:00	3.75		United Kingdom
C541492	20931	BLUE POT PLANT CANDLE	-1	2011. 1. 18 오후 2:24:00	3.75		United Kingdom
C541650	M	Manual	-1	2011. 1. 20 오전 11:44:00	544.4		United Kingdom
C541651	M	Manual	-1	2011. 1. 20 오전 11:48:00	1283.8		United Kingdom
C541653	BANK CHARGES	Bank Charges	-1	2011. 1. 20 오전 11:50:00	1050.15		United Kingdom
C542540	DOT	DOTCOM POSTAGE	-1	2011. 1. 28 오후 2:20:00	3.29		United Kingdom
C542540	POST	POSTAGE	-1	2011. 1. 28 오후 2:20:00	4.41		United Kingdom
C542540	21658	GLASS BEURRE DISH	-1	2011. 1. 28 오후 2:20:00	8.29		United Kingdom
C542860	22580	ADVENT CALENDAR GINGHAM SACK	-1	2011. 2. 1 오전 11:43:00	5.95		United Kingdom
C543185	22333	RETROSPOT PARTY BAG + STICKER SET	-14	2011. 2. 4 오전 11:22:00	1.65		United Kingdom
C543185	22332	SKULLS PARTY BAG + STICKER SET	-14	2011. 2. 4 오전 11:22:00	1.65		United Kingdom
C543384	48173C	DOORMAT BLACK FLOCK	-1	2011. 2. 7 오후 4:24:00	7.95		United Kingdom
C544047	M	Manual	-1	2011. 2. 15 오후 12:36:00	1435.79		United Kingdom
C544049	BANK CHARGES	Bank Charges	-1	2011. 2. 15 오후 12:39:00	566.37		United Kingdom
C544054	M	Manual	-1	2011. 2. 15 오후 12:41:00	869.55		United Kingdom
C544575	BANK CHARGES	Bank Charges	-1	2011. 2. 21 오후 1:58:00	134.76		United Kingdom
C544576	BANK CHARGES	Bank Charges	-1	2011. 2. 21 오후 2:01:00	149.16		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	10		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	9.99		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	5.74		United Kingdom

C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	7.69		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	9.74		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	5.44		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	7		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	5.79		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	20.98		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	6.7		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	19.5		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	20.55		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	8.74		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	29.99		United Kingdom
C544580	S	SAMPLES	-1	2011. 2. 21 오후 2:25:00	11.08		United Kingdom
C544581	S	SAMPLES	-1	2011. 2. 21 오후 2:32:00	55		United Kingdom
C544581	S	SAMPLES	-1	2011. 2. 21 오후 2:32:00	92		United Kingdom
C544581	S	SAMPLES	-1	2011. 2. 21 오후 2:32:00	12.94		United Kingdom
C544581	S	SAMPLES	-1	2011. 2. 21 오후 2:32:00	107.99		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	95		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	39.5		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	4.59		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	31.98		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	2.8		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	37.49		United Kingdom
C544583	S	SAMPLES	-1	2011. 2. 21 오후 2:48:00	128.56		United Kingdom
C544584	BANK CHARGES	Bank Charges	-1	2011. 2. 21 오후 2:52:00	109.84		United Kingdom
C544587	AMAZONFEE	AMAZON FEE	-1	2011. 2. 21 오후 3:07:00	5575.28		United Kingdom
C544589	AMAZONFEE	AMAZON FEE	-1	2011. 2. 21 오후 3:11:00	5258.77		United Kingdom
C544671	S	SAMPLES	-1	2011. 2. 22 오후 3:45:00	21.9		United Kingdom
C545506	22907	PACK OF 20 NAPKINS PANTRY DESIGN	-12	2011. 3. 3 오전 11:42:00	0.85		EIRE
C545884	M	Manual	-1	2011. 3. 7 오후 3:49:00	537.83		United Kingdom

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

```
SELECT
  ROUND(
    COUNTIF(InvoiceNo LIKE 'C%') / COUNT(*) * 100,
    1
  ) AS canceled_percentage
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행    canceled_percentage
1     1.7
```

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행    unique_stockcode_count
1     3958
```

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

```
SELECT
  StockCode,
  COUNT(*) AS sell_cnt
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

```
행    StockCode    sell_cnt
1     85123A   2301
2     22423    2192
3     85099B   2156
4     47566    1720
5     20725    1626
6     84879    1489
7     22197    1468
8     22720    1465
9     21212    1367
10    22383    1328
```

- **`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,
  -- 숫자를 제거하고 문자만 남긴 값
  REGEXP_REPLACE(StockCode, r'[0-9]', '') AS only_letters,
  -- 문자만 남긴 값의 길이
  LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS letter_length,

  -- StockCode 안에 숫자가 몇 개 있는지 계산
  LENGTH(StockCode)
    - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS digit_count

FROM `cobalt-sector-479904-d9.modulabs_project_3.data`

-- 숫자가 0개 또는 1개인 StockCode만 필터링
WHERE
  (LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))) <= 1

ORDER BY
  StockCode
LIMIT 200;
```

StockCode	only_letters	letter_length	digit_count
AMAZONFEE	AMAZONFEE	9	0
B	B	1	0
BANK CHARGES	BANK CHARGES	12	0
C2	C	1	1
CRUK	CRUK	4	0
D	D	1	0
DCGSSBOY	DCGSSBOY	8	0
DCGSSGIRL	DCGSSGIRL	9	0
DOT	DOT	3	0
M	M	1	0
PADS	PADS	4	0
POST	POST	4	0
S	S	1	0
m	m	1	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)

```
WITH stock_digit AS (
  SELECT
    StockCode,
    LENGTH(StockCode)
      - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
)
SELECT
  ROUND(
    COUNTIF(number_count <= 1) / COUNT(*) * 100
  , 2) AS pct_rows_with_0_1_digits
FROM stock_digit;
```

```
행    pct_rows_with_0_1_digits
1     0.55
```

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
SELECT DISTINCT StockCode
FROM (
  SELECT
    StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
)
WHERE number_count <= 1;
```

이 문으로 data의 행 2,928개가 삭제되었습니다.

Description 살펴보기

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

```
SELECT
  Description,
  COUNT(*) AS description_cnt
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

Description	description_cnt
WHITE HANGING HEART T-LIGHT HOLDER	2357
REGENCY CAKESTAND 3 TIER	2189
JUMBO BAG RED RETROSPOT	2156
PARTY BUNTING	1720
LUNCH BAG RED RETROSPOT	1625
ASSORTED COLOUR BIRD ORNAMENT	1488
SET OF 3 CAKE TINS PANTRY DESIGN	1465
PACK OF 72 RETROSPOT CAKE CASES	1367
LUNCH BAG BLACK SKULL.	1323
NATURAL SLATE HEART CHALKBOARD	1272
JUMBO BAG PINK POLKADOT	1245
HEART OF WICKER SMALL	1230
JAM MAKING SET WITH JARS	1221
JUMBO STORAGE BAG SUKI	1211
PAPER CHAIN KIT 50'S CHRISTMAS	1194
JUMBO SHOPPER VINTAGE RED PAISLEY	1192
LUNCH BAG CARS BLUE	1185
JAM MAKING SET PRINTED	1177
LUNCH BAG SPACEBOY DESIGN	1177
RECIPE BOX PANTRY YELLOW DESIGN	1173
SPOTTY BUNTING	1168
ROSES REGENCY TEACUP AND SAUCER	1128
WOODEN PICTURE FRAME WHITE FINISH	1124
LUNCH BAG PINK POLKADOT	1121
LUNCH BAG SUKI DESIGN	1121
SET OF 4 PANTRY JELLY MOULDS	1099
ALARM CLOCK BAKELIKE RED	1092
GREEN REGENCY TEACUP AND SAUCER	1066

VICTORIAN GLASS HANGING T-LIGHT	1066
LUNCH BAG APPLE DESIGN	1066

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.data` AS
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE Description NOT IN (
  'POSTAGE',
  'AMAZONFEE',
  'BANK CHARGES',
  'DOT',
  'CRUK',
  'DCGSSGIRL',
  'DCGSSBOY'
)
AND Description IS NOT NULL
AND Description != '?';
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행  Description
1   BATHROOM METAL SIGN
2   VINTAGE GLASS COFFEE CADDY
3   CARD CHRISTMAS VILLAGE
4   PACK OF 12 SPACEBOY TISSUES
5   SWEET PUDDING STICKER SHEET
6   BOYS ALPHABET IRON ON PATCHES
7   VINTAGE BILLBOARD TEA MUG
8   HEADS AND TAILS SPORTING FUN
9   PIZZA PLATE IN BOX
10  UNION STRIPE WITH FRINGE  HAMMOCK
11  ADVENT CALENDAR GINGHAM SACK
12  LOVEBIRD HANGING DECORATION WHITE
13  ENAMEL WATERING CAN CREAM
14  COFFEE MUG DOG + BALL DESIGN
15  DECORATIVE CATS BATHROOM BOTTLE
16  VINTAGE HEADS AND TAILS CARD GAME
17  WATERING CAN PINK BUNNY
18  FAIRY CAKE NOTEBOOK A5 SIZE
19  CHERRY BLOSSOM  DECORATIVE FLASK
20  ANTIQUE ALL GLASS CANDLESTICK
21  SCOTTIES CHILDRENS APRON
22  AIRLINE BAG VINTAGE JET SET BROWN
23  CHILDS BREAKFAST SET DOLLY GIRL
24  GENTLEMAN SHIRT REPAIR KIT
25  12 PENCIL SMALL TUBE WOODLAND
26  MINI JIGSAW BUNNIES
27  MODERN FLORAL STATIONERY SET
28  SCOTTIE DOGS BABY BIB
29  GROW YOUR OWN PLANT IN A CAN
```

```
30    CERAMIC STRAWBERRY CAKE MONEY BANK
31    LARGE PURPLE BABUSHKA NOTEBOOK
32    BROCADE RING PURSE
33    GUMBALL MONOCHROME COAT RACK
34    SWEETHEART CERAMIC TRINKET BOX
35    HOT WATER BOTTLE BABUSHKA
36    MINI JIGSAW DOLLY GIRL
37    FRENCH LATTICE CUSHION COVER
38    BLUE PATCH PURSE PINK HEART
39    DOVE DECORATION PAINTED ZINC
40    SET OF 12  VINTAGE POSTCARD SET
41    LADLE LOVE HEART PINK
42    CHILDREN'S SPACEBOY MUG
43    CARDHOLDER GINGHAM STAR
44    CHARLIE LOLA BLUE HOT WATER BOTTLE
45    ANT COPPER RED BOUDICCA BRACELET
46    WATERING CAN GARDEN MARKER
47    PAPERWEIGHT KINGS CHOICE
48    LADLE LOVE HEART RED
49    RED RETROSPOT CUP
50    IVY HEART WREATH
```

`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 `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행    min_price       max_price       avg_price
1     0.0   649.5     3.2875045094811179
```

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

```
SELECT
  COUNT(*) AS cnt_quantity,          -- 단가 0원 거래 개수
  MIN(Quantity) AS min_quantity,      -- 구매 수량 최솟값
  MAX(Quantity) AS max_quantity,      -- 구매 수량 최댓값
  AVG(Quantity) AS avg_quantity       -- 구매 수량 평균
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE UnitPrice = 0;
```

```
행    cnt_quantity   min_quantity   max_quantity  avg_quantity
1     1000       -9600      12540     -119.37600000000037
```

- **`UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.data` AS
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
WHERE UnitPrice != 0;
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

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

```
SELECT
  column_name,
  data_type
FROM `cobalt-sector-479904-d9.modulabs_project_3.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'data'
  AND column_name = 'InvoiceDate';
```

```
행    column_name data_type
1     InvoiceDate    DATE
```

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
  MAX(InvoiceDate) AS most_recent_date
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`;
```

```
행    most_recent_date
1     2011-12-09
```

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

```
SELECT
  CustomerID,
  MAX(InvoiceDate) AS most_recent_purchase_date
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY CustomerID
ORDER BY most_recent_purchase_date DESC
LIMIT 10;
```

```
행    CustomerID    most_recent_purchase_date
1     18102      2011-12-09
2     null  2011-12-09
3     13113      2011-12-09
4     16558      2011-12-09
5     13777      2011-12-09
6     12433      2011-12-09
7     13069      2011-12-09
8     15311      2011-12-09
9     17581      2011-12-09
10    12748      2011-12-09
```

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

```
WITH dates AS (
  SELECT
    CustomerID,
    MAX(InvoiceDate) AS last_purchase_date,
    (SELECT MAX(InvoiceDate) FROM `cobalt-sector-479904-d9.modulabs_project_3.data`) AS most_recent_date
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY CustomerID
)

SELECT
  CustomerID,
  last_purchase_date,
  most_recent_date,
  DATE_DIFF(most_recent_date, last_purchase_date, DAY) AS date_difference
FROM dates
ORDER BY date_difference DESC
LIMIT 20;
```

```
행  CustomerID  last_purchase_date  most_recent_date  date_difference
1   13065  2010-12-01  2011-12-09  373
2   18074  2010-12-01  2011-12-09  373
3   14237  2010-12-01  2011-12-09  373
4   14729  2010-12-01  2011-12-09  373
5   17908  2010-12-01  2011-12-09  373
6   16583  2010-12-01  2011-12-09  373
7   15165  2010-12-01  2011-12-09  373
8   13747  2010-12-01  2011-12-09  373
9   15350  2010-12-01  2011-12-09  373
10  16274  2010-12-01  2011-12-09  373
11  17643  2010-12-01  2011-12-09  373
12  18011  2010-12-01  2011-12-09  373
13  14142  2010-12-01  2011-12-09  373
14  12791  2010-12-01  2011-12-09  373
15  17968  2010-12-01  2011-12-09  373
16  17925  2010-12-02  2011-12-09  372
17  13958  2010-12-02  2011-12-09  372
18  16754  2010-12-02  2011-12-09  372
19  15922  2010-12-02  2011-12-09  372
20  15923  2010-12-02  2011-12-09  372
```

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

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.user_r` AS
WITH base AS (
  -- 유저별 기본 지표 계산
  SELECT
    CustomerID,
    MAX(InvoiceDate) AS last_purchase_date,              -- 마지막 구매일
    COUNT(DISTINCT InvoiceNo) AS frequency,              -- 구매 건수(F)
    SUM(Quantity * UnitPrice) AS monetary                -- 총 매출(M)
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY CustomerID
),
most_recent AS (
  -- 전체 데이터 중 가장 최근 날짜 1개
  SELECT
    MAX(InvoiceDate) AS most_recent_date
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
)

SELECT
  b.CustomerID,
  b.last_purchase_date,
  m.most_recent_date,
  DATE_DIFF(m.most_recent_date, b.last_purchase_date, DAY) AS recency,  -- 날짜 차이(R)
  b.frequency,
  b.monetary
FROM base b
CROSS JOIN most_recent m;
```

CustomerID	last_purchase_date	most_recent_date	recency	frequency	monetary
12748	2011. 12. 9	2011. 12. 9	0	217	29819.99
16705	2011. 12. 9	2011. 12. 9	0	29	13946.13
17581	2011. 12. 9	2011. 12. 9	0	31	10716.31
18102	2011. 12. 9	2011. 12. 9	0	60	259657.3
16626	2011. 12. 9	2011. 12. 9	0	20	4379.65
13069	2011. 12. 9	2011. 12. 9	0	27	3713.14
17389	2011. 12. 9	2011. 12. 9	0	42	31317.48
14397	2011. 12. 9	2011. 12. 9	0	23	2556.68
13113	2011. 12. 9	2011. 12. 9	0	39	10523.65
15311	2011. 12. 9	2011. 12. 9	0	118	59284.19

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY CustomerID
ORDER BY unique_invoice_count DESC;
```

CustomerID	unique_invoice_count
NULL	1410
14911	242
12748	217
17841	169
14606	125
13089	118
15311	118
12971	88
13408	75
14646	73

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS total_quantity
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY CustomerID
ORDER BY total_quantity DESC;
```

CustomerID	total_quantity
	417233
14646	196556

12415	76946
14911	76823
17450	69021
18102	64124
17511	63014
13694	61904
14298	58021
14156	56896

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

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.user_rf` AS
WITH rf AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,   -- 거래 건수
    SUM(Quantity) AS total_quantity                    -- 구매한 총 수량
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY CustomerID
)

SELECT *
FROM rf;

--확인
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_rf`
LIMIT 10;
```

CustomerID	total_transactions	total_quantity
15316	1	100
16757	1	81
16030	1	283
17893	1	149
16545	1	78
15942	1	232
13075	1	485
15466	1	246
16380	1	1428
12847	1	873

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS total_spending
FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
GROUP BY CustomerID
```

```
ORDER BY total_spending DESC
LIMIT 10;
```

CustomerID	total_spending
	1506232.1
14646	278778
18102	259657.3
17450	189575.5
14911	128768.2
12415	123638.2
14156	113685.8
17511	88138.2
16684	65920.1
13694	62961.5

- **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 `cobalt-sector-479904-d9.modulabs_project_3.user_rfm` AS
SELECT
  u.CustomerID,
  u.total_transactions AS purchase_cnt,          -- 거래 건수
  u.total_quantity,
  -- 고객별 총 지출액 / 거래 건수 = 평균 거래 금액
  ROUND(SUM(d.Quantity * d.UnitPrice) / u.total_transactions, 1) AS avg_transaction_amount
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_rf` AS u
LEFT JOIN `cobalt-sector-479904-d9.modulabs_project_3.data` AS d
  ON u.CustomerID = d.CustomerID
GROUP BY
  u.CustomerID,
  u.total_transactions,
  u.total_quantity;

--확인
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_rfm`
LIMIT 10;
```

CustomerID	purchase_cnt	total_quantity	avg_transaction_amount
12659	1	104	73.7
17070	1	131	304.2
13095	1	144	74.4
16204	1	218	384.2
12448	1	240	365.4
13170	1	141	108.8
12509	1	54	158.5
15783	1	212	246.3
15723	1	73	190.7
14537	1	120	355.6

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_rfm`
LIMIT 10;
```

CustomerID	purchase_cnt	total_quantity	avg_transaction_amount
14459	1	760	1837.9
16601	1	130	241.5
13979	1	639	869.9
14975	1	380	279.9
15657	1	24	30
13753	1	512	741.3
17475	1	178	194.4
17443	1	504	534.2
15442	1	120	594
17165	1	100	158.7

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 project_name.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

--확인
SELECT *
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_data`
LIMIT 10;
```

CustomerID	purchase_cnt	total_quantity	avg_transaction_amount	unique_items
15195	1	1404	3861	1
14090	1	72	76.3	1
13270	1	200	590	1
15940	1	4	35.8	1
16323	1	50	207.5	1
18068	1	6	101.7	1
17443	1	504	534.2	1

15753	1	144	79.2	1
16138	1	-1	-8	1
13829	1	-12	-102	1

2. 평균 구매 주기

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

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

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM project_name.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

CustomerID	purchase_cnt	total_quantity	avg_transaction_amount	unique_items	avg_purchase_interval
15668	1	72	76.3	1	
13017	1	48	204	1	
13185	1	12	71.4	1	
14705	1	100	179	1	
15316	1	100	165	1	
16579	1	-12	-30.6	1	
15524	1	4	440	1	
16061	1	-1	-29.9	1	
17752	1	192	80.6	1	
13747	1	8	79.6	1	

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 `cobalt-sector-479904-d9.modulabs_project_3.user_data` AS
WITH cancel_stats AS (
  -- 1) 고객별 취소 빈도와 비율 계산
  SELECT
    CustomerID,
    COUNTIF(STARTS_WITH(InvoiceNo, 'C')) AS cancel_frequency,         -- 취소 횟수
    COUNT(*) AS total_transactions,                                   -- 전체 거래 수
    SAFE_DIVIDE(
      COUNTIF(STARTS_WITH(InvoiceNo, 'C')),
      COUNT(*)
    ) AS cancel_rate_raw                                              -- 비율(0~1)
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY CustomerID
)

-- 2) 기존 user_data에 결합해서 저장
SELECT
  u.*,
  IFNULL(c.cancel_frequency, 0) AS cancel_frequency,                  -- 취소 빈도
  IFNULL(ROUND(c.cancel_rate_raw, 2), 0.0) AS cancel_rate             -- 취소 비율(소수 둘째 자리)
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_data` AS u
LEFT JOIN cancel_stats AS c
  ON u.CustomerID = c.CustomerID;

  --확인
  SELECT
  CustomerID,
  cancel_frequency,
  cancel_rate
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_data`
LIMIT 10;
```

CustomerID	cancel_frequency	cancel_rate
12603	0	0
17331	0	0
16881	0	0
16526	0	0
12430	0	0
15149	0	0
15097	0	0
12651	0	0
13532	0	0
12733	0	0

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

```
CREATE OR REPLACE TABLE `cobalt-sector-479904-d9.modulabs_project_3.user_data` AS
WITH spending AS (
  SELECT
    CustomerID,
    SUM(Quantity * UnitPrice) AS total_spending
  FROM `cobalt-sector-479904-d9.modulabs_project_3.data`
  GROUP BY CustomerID
)

SELECT
```

```
  u.*,
  ROUND(s.total_spending, 1) AS total_spending
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_data` AS u
LEFT JOIN spending AS s
  ON u.CustomerID = s.CustomerID;

--확인
SELECT
  CustomerID,
  total_spending,
  purchase_cnt,
  unique_items,
  cancel_rate
FROM `cobalt-sector-479904-d9.modulabs_project_3.user_data`
ORDER BY total_spending DESC
LIMIT 10;
```

CustomerID	total_spending	purchase_cnt	unique_items	cancel_rate
14646	278778	73	699	0
18102	259657.3	60	150	0
17450	189575.5	49	124	0.01
14911	128768.2	242	1791	0.04
12415	123638.2	24	443	0.08
14156	113685.8	64	714	0.01
17511	88138.2	45	465	0.1
16684	65920.1	30	119	0.01
13694	62961.5	57	367	0.02
16029	60369.9	66	43	0.08

회고

[회고 내용을 작성해주세요]

Keep : 일단 끝까지 완주했다 ㅜㅜ

Problem : gpt에 의지를 많이 했다

Try : 다시 혼자힘으로 해봐야겠다