

고객을 세그멘테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM enhanced-ward-466622-g5.1st_project.Data1  
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT count(*) AS total_row_number  
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	total_row_number
1	541909

데이터 수 세기

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

```
SELECT count(InvoiceNo) AS COUNT_InvoiceNo,  
count(StockCode) AS COUNT_StockCode,  
count(Description) AS COUNT_Description,  
count(Quantity) AS COUNT_Quantity,  
count(InvoiceDate) AS COUNT_InvoiceDate,  
count(UnitPrice) AS UnitPrice,  
count(CustomerID) AS COUNT_CustomerID,  
count(Country) AS COUNT_Country  
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	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

```

SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1

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 enhanced-ward-466622-g5.1st_project.Data1;

```

[결과 이미지를 넣어주세요]

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

결측치 처리 전략

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

```
SELECT DISTINCT Description
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE StockCode = '85123A';
```

[결과 이미지를 넣어주세요]

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

[결과 이미지를 넣어주세요]

```
DELETE FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE Description IS NULL OR CustomerID IS NULL;
```

i 이 문으로 Data1의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
WITH DC AS (
  SELECT
    InvoiceNo, StockCode, Description, Quantity,
    InvoiceDate, UnitPrice, CustomerID, Country,
    COUNT(*) AS row_count
  FROM enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY
    InvoiceNo, StockCode, Description, Quantity,
    InvoiceDate, UnitPrice, CustomerID, Country
)

SELECT COUNT(*) AS '중복 행 갯수'
```

```
FROM DC
WHERE row_count > 1;
```

[결과 이미지를 넣어주세요]

행	중복 행 갯수
1	4837

중복값 처리

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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.Data1 AS
SELECT DISTINCT *
FROM enhanced-ward-466622-g5.1st_project.Data1;

SELECT count(*) AS new_total_row_number
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 Data1인 테이블이 교체되었습니다.

행	new_total_row_number
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```
SELECT count(DISTINCT InvoiceNo) AS unique_COUNT_INvoiceNo
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	unique_COUNT_INvoiceNo
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM enhanced-ward-466622-g5.1st_project.Data1
LIMIT 100;
```

[결과 이미지를 넣어주세요]

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

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

```
SELECT *
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE InvoiceNO LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORA...	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	84050	PINK HEART SHAPE EGG FRYN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
6	C547388	25645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
7	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	27448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
9	C547388	22784	LANTERN CREAM CAZERO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
10	C547388	22413	METAL SIGN TAKE IT OR LEAVE...	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
11	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
12	C548955	22666	RECOPE BOX PANTRY YELLOW ...	-2	2011-04-13 13:38:00 UTC	2.95	12359	Cyprus
13	C548955	22839	3 TIER CAKE TIN GREEN AND C...	-2	2011-04-13 13:38:00 UTC	14.95	12359	Cyprus

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

```
SELECT ROUND((SUM(CASE WHEN InvoiceNO LIKE 'C%' THEN 1 ELSE 0 END)*100)/COUNT(*),1) AS CancelRatio
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	CancelRatio
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT count(DISTINCT StockCode) AS unique_COUNT_StockCode
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	unique_COUNT_StockCode
1	3684

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 **StockCode** 별 등장 빈도를 출력하기

- 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

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 값들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM enhanced-ward-466622-g5.1st_project.Data1
)
WHERE number_count <=1;
```

[결과 이미지를 넣어주세요]

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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
SELECT StockCode, ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM enhanced-ward-466622-g5.1st_project.Data1), 2)
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1
GROUP BY StockCode;

SELECT ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM enhanced-ward-466622-g5.1st_project.Data1), 2) AS total_co
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1;
```

[결과 이미지를 넣어주세요]

행	StockCode	code_ratio
1	POST	0.3
2	M	0.11
3	C2	0.03
4	D	0.02
5	BANK CHARGES	0.0
6	PADS	0.0
7	DOT	0.0
8	CRUK	0.0

행	total_code_ratio
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM enhanced-ward-466622-g5.1st_project.Data1
    WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1
  )
);
```

[결과 이미지를 넣어주세요]

i 이 문으로 Data1의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

```
SELECT Description, COUNT(*) AS description_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY Description
ORDER BY 2 DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]

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 D...	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 WHI...	900
21	REX CASH+CARRY JUMBO SHO...	900
22	JUMBO BAG PINK POLKADOT	897
23	LUNCH BAG APPLE DESIGN	890
24	SET OF 4 PANTRY JELLY MOUL...	890
25	BAKING SET 9 PIECE RETROSP...	885
26	JAM MAKING SET PRINTED	883
27	RECIPE BOX PANTRY YELLOW ...	883
28	LUNCH BAG WOODLAND	850
29	ROSES REGENCY TEACUP AND ...	844
30	VICTORIAN GLASS HANGING T...	843

• 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE Description IN ('High Resolution Image','Next Day Carriage');
```

[결과 이미지를 넣어주세요]

i 이 문으로 Data1의 행 83개가 삭제되었습니다.

• 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.Data1 AS
SELECT
* EXCEPT (Description),
UPPER(Description) AS Description
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 Data1인 테이블이 교체되었습니다.

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 enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.0	649.5	2.907457172951...

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

```
SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS
FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE UnitPrice = 0 ;
```

[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
DELETE FROM enhanced-ward-466622-g5.1st_project.Data1
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

i 이 문으로 Data1의 행 33개가 삭제되었습니다.

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	Invoice	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-01-18	541431	23166	MEDIUM CERAMIC TOP STORA...	74218	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	2011-01-18	CS41433	23166	MEDIUM CERAMIC TOP STORA...	-74218	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	2010-10-07	537626	849978	RED 3 PIECE RETROFOT OUTL...	6	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
4	2010-10-07	537626	22772	PINK DRAWER KNOB ACRYLIC...	12	2010-10-07 14:57:00 UTC	1.25	12347	Iceland
5	2010-10-07	537626	21731	RED TOADSTOOL LED NIGHT L...	12	2010-10-07 14:57:00 UTC	1.65	12347	Iceland
6	2010-10-07	537626	85116	BLACK CANDELABRA TIGHT...	12	2010-10-07 14:57:00 UTC	2.1	12347	Iceland
7	2010-10-07	537626	22726	ALARM CLOCK BAKELIKE GREEN	4	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
8	2010-10-07	537626	22725	ALARM CLOCK BAKELIKE CHO...	4	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
9	2010-10-07	537626	22727	ALARM CLOCK BAKELIKE RED	4	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
10	2010-10-07	537626	22497	SET OF 2 TINS VINTAGE BATHR...	4	2010-10-07 14:57:00 UTC	4.25	12347	Iceland
11	2010-10-07	537626	22805	BLUE DRAWER KNOB ACRYLIC...	12	2010-10-07 14:57:00 UTC	1.25	12347	Iceland
12	2010-10-07	537626	851178	BLACK GRAND BAROQUE PROT...	36	2010-10-07 14:57:00 UTC	1.25	12347	Iceland
13	2010-10-07	537626	22312	FOUR HOOK WHITE LOVESBROS	6	2010-10-07 14:57:00 UTC	2.1	12347	Iceland
14	2010-10-07	537626	21984	BOOM BOX SPEAKER BOYS	6	2010-10-07 14:57:00 UTC	5.95	12347	Iceland
15	2010-10-07	537626	22728	ALARM CLOCK BAKELIKE PINK	4	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
16	2010-10-07	537626	20782	CAMOUFLAGE KAS MUFF HEA...	6	2010-10-07 14:57:00 UTC	5.49	12347	Iceland
17	2010-10-07	537626	21171	BATHROOM METAL SIGN	12	2010-10-07 14:57:00 UTC	1.45	12347	Iceland
18	2010-10-07	537626	849970	PINK 3 PIECE POLKADOT OUTL...	6	2010-10-07 14:57:00 UTC	3.75	12347	Iceland
19	2010-10-07	537626	22492	MINI PAINT SET VINTAGE	36	2010-10-07 14:57:00 UTC	0.65	12347	Iceland
20	2010-10-07	537626	22774	RED DRAWER KNOB ACRYLIC E...	12	2010-10-07 14:57:00 UTC	1.25	12347	Iceland
21	2010-10-07	537626	852039	SET 16 ECOLOURS STAINING...	3	2010-10-07 14:57:00 UTC	4.95	12347	Iceland
22	2010-10-07	537626	22375	AIRLINE BAG VINTAGE JET SET	4	2010-10-07 14:57:00 UTC	4.25	12347	Iceland
23	2010-10-07	537626	84508A	3D DOG PICTURE PLAYING CAR...	24	2010-10-07 14:57:00 UTC	2.95	12347	Iceland

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
    MAX(DATE(InvoiceDate)) OVER () AS most_recent_date,
    DATE(InvoiceDate) AS InvoiceDay, *
FROM enhanced-ward-466622-g5.1st_project.Data1;
```

[결과 이미지를 넣어주세요]

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09	2011-01-12	540946	82484	WOOD BLACK BOARD ANT WHL...	12	2011-01-12 12:43:00 UTC	5.55	12359	Cyprus
2	2011-12-09	2011-11-17	577125	23584	RABBIT NIGHT LIGHT	24	2011-11-17 19:22:00 UTC	1.79	12406	Denmark
3	2011-12-09	2011-01-06	540287	22279	RECYCLING BAG RETROPOUT	100	2011-01-06 11:12:00 UTC	1.85	12415	Australia
4	2011-12-09	2011-02-15	543989	21058	PARTY INVITES WOODLAND	144	2011-02-15 09:52:00 UTC	5.72	12415	Australia
5	2011-12-09	2011-03-03	545475	21217	RED RETROPOUT ROUND CAKE...	120	2011-03-03 10:59:00 UTC	6.95	12415	Australia
6	2011-12-09	2011-03-01	545226	21579	WOODLAND DESIGN COTTON...	6	2011-03-01 09:33:00 UTC	2.25	12428	Finland
7	2011-12-09	2011-06-05	555572	18057	ESSENTIAL BALM 3.5g TIN IN E...	24	2011-06-05 15:37:00 UTC	0.18	12449	Belgium
8	2011-12-09	2011-10-11	570640	23610	CIRCUS PANACHE BABY GIFT SET	1	2011-10-11 12:39:00 UTC	16.95	12474	Germany
9	2011-12-09	2011-11-17	576910	23746	TRIPLE WINK ANTIQUS KIPPY...	8	2011-11-17 09:51:00 UTC	3.29	12477	Germany
10	2011-12-09	2011-01-07	540469	842518	GREETING CARD STICKY GORD...	12	2011-01-07 14:04:00 UTC	0.19	12484	Spain
11	2011-12-09	2011-09-23	567959	22945	CHRISTMAS METAL TAGS ASS...	144	2011-09-23 09:13:00 UTC	0.72	12516	Germany
12	2011-12-09	2011-08-08	562655	21829	INDOSAUR KEYRINGS ASSORTED	72	2011-08-08 10:59:00 UTC	0.21	12530	Germany
13	2011-12-09	2011-09-21	567653	21625	VINTAGE LINCOLN JACK APRON	3	2011-09-21 14:33:00 UTC	6.95	12550	Spain
14	2011-12-09	2011-12-01	579627	23446	HOME SWEET HOME KEY HOL...	2	2011-12-01 09:20:00 UTC	6.25	12572	Germany
15	2011-12-09	2011-11-04	574201	21452	RED EGG SPOON	3	2011-11-04 15:15:00 UTC	0.12	12577	France
16	2011-12-09	2011-11-30	5291785	22845	VINTAGE CREAM CAT FOOD CO...	3	2011-11-30 15:29:00 UTC	6.35	12584	Italy
17	2011-12-09	2011-10-30	573262	23486	ANTIQUE HEART GIFT UNIT	1	2011-10-30 13:00:00 UTC	16.65	12597	Spain
18	2011-12-09	2011-06-30	558628	22253	JIGSAW RABBIT AND BIRCHOL...	12	2011-06-30 17:59:00 UTC	0.59	12626	Germany
19	2011-12-09	2011-07-08	559962	21576	LETS GO SHOPPING COTTON T...	6	2011-07-08 08:43:00 UTC	2.25	12635	Germany
20	2011-12-09	2010-12-07	557064	22947	BREAD BIN DAMES STYLE VIO...	4	2010-12-07 12:28:00 UTC	14.95	12647	Germany
21	2011-12-09	2011-10-28	573153	23584	RABBIT NIGHT LIGHT	912	2011-10-28 07:39:00 UTC	1.79	12678	France
22	2011-12-09	2011-12-06	580793	23163	REGENCY SUGAR TONGS	8	2011-12-06 10:00:00 UTC	2.49	12682	France
23	2011-12-09	2011-08-26	564670	22435	SET OF 9 HEART SHAPED BALL...	90	2011-08-26 15:44:00 UTC	1.06	12683	France

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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
18	12364	2011-12-02
19	12365	2011-02-21
20	12367	2011-12-05
21	12370	2011-10-19
22	12371	2011-10-11

- 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 enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	12373	311
2	12588	39
3	13038	80
4	13135	196
5	13499	17
6	13503	71
7	13865	58
8	13924	1
9	13984	26
10	14261	50
11	14411	310
12	14462	63
13	14560	7
14	14698	1
15	14747	247
16	15004	147
17	15158	45
18	15444	9
19	15577	151
20	15611	9
21	15900	23
22	16050	173
23	16402	264
24	16468	74

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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency,
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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
19	12365	1
20	12367	1
21	12370	4
22	12371	1
23	12372	3

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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
18	12364	1499
19	12365	173
20	12367	172
21	12370	2349
22	12371	582
23	12372	788

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

```

CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.user_rf AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  COUNT(Quantity) AS item_cnt
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID
)
-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recentry
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN enhanced-ward-466622-g5.1st_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;

```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15520	1	314	1
3	13298	1	96	1
4	14569	1	79	1
5	13436	1	76	1
6	15195	1	1404	2
7	15471	1	256	2
8	14204	1	72	2
9	12478	1	233	3
10	12650	1	250	3
11	17914	1	457	3
12	14578	1	240	3
13	16569	1	93	3
14	15318	1	642	3
15	16528	1	171	3
16	12442	1	181	3
17	15992	1	17	3
18	12367	1	172	4
19	14219	1	78	4
20	15097	1	170	4
21	17383	1	148	4
22	16597	1	184	4
23	13790	1	748	4

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice*Quantity)) AS user_total
FROM enhanced-ward-466622-g5.1st_project.Data1
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

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
19	12365	321.0
20	12367	151.0
21	12370	3422.0
22	12371	1528.0
23	12372	1196.0
24	12373	325.0
25	12374	623.0

- 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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_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,1) AS user_average
FROM enhanced-ward-466622-g5.1st_project.user_rf AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(UnitPrice*Quantity)) AS user_total
  FROM enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

```
SELECT *
FROM enhanced-ward-466622-g5.1st_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	15195	1	1404	2	3861.0	3861.0
7	14204	1	72	2	151.0	151.0
8	15471	1	256	2	454.0	454.0
9	15992	1	17	3	42.0	42.0
10	14578	1	240	3	169.0	169.0
11	15318	1	642	3	313.0	313.0
12	12442	1	181	3	144.0	144.0
13	16569	1	93	3	124.0	124.0
14	16528	1	171	3	244.0	244.0
15	12478	1	233	3	546.0	546.0
16	12650	1	250	3	242.0	242.0
17	17914	1	457	3	329.0	329.0
18	17383	1	148	4	193.0	193.0
19	14219	1	78	4	90.0	90.0
20	16597	1	184	4	90.0	90.0
21	13790	1	748	4	349.0	349.0
22	12367	1	172	4	151.0	151.0
23	15097	1	170	4	248.0	248.0
24	18015	1	157	4	120.0	120.0
25	16988	1	43	5	126.0	126.0

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 enhanced-ward-466622-g5.1st_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM enhanced-ward-466622-g5.1st_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;

SELECT *
FROM enhanced-ward-466622-g5.1st_project.user_data
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	12713	1	505	0	795.0	795.0	37
2	15520	1	314	1	344.0	344.0	18
3	13436	1	76	1	197.0	197.0	12
4	13298	1	96	1	360.0	360.0	2
5	14569	1	79	1	227.0	227.0	10
6	14204	1	72	2	151.0	151.0	36
7	15195	1	1404	2	3861.0	3861.0	1
8	15471	1	256	2	454.0	454.0	67
9	15318	1	642	3	313.0	313.0	33
10	16569	1	93	3	124.0	124.0	5
11	16528	1	171	3	244.0	244.0	17
12	12650	1	250	3	242.0	242.0	19
13	12478	1	233	3	546.0	546.0	35
14	14578	1	240	3	169.0	169.0	24
15	12442	1	181	3	144.0	144.0	11
16	15992	1	17	3	42.0	42.0	3
17	17914	1	457	3	329.0	329.0	72
18	15097	1	170	4	248.0	248.0	25
19	18015	1	157	4	120.0	120.0	46
20	12367	1	172	4	151.0	151.0	10
21	14219	1	78	4	90.0	90.0	7
22	16597	1	184	4	90.0	90.0	7
23	13790	1	748	4	349.0	349.0	45

2. 평균 구매 주기

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

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_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
      enhanced-ward-466622-g5.1st_project.Data1
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM enhanced-ward-466622-g5.1st_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	16990	1	100	218	179.0	179.0	1	0.0
2	16061	1	-1	269	-30.0	-30.0	1	0.0
3	15118	1	1440	134	245.0	245.0	1	0.0
4	13135	1	4300	196	3096.0	3096.0	1	0.0
5	18133	1	1350	212	931.0	931.0	1	0.0
6	14090	1	72	324	76.0	76.0	1	0.0
7	13366	1	144	50	56.0	56.0	1	0.0
8	16257	1	1	176	22.0	22.0	1	0.0
9	16148	1	72	296	76.0	76.0	1	0.0
10	15753	1	144	304	79.0	79.0	1	0.0
11	14679	1	-1	371	-3.0	-3.0	1	0.0
12	13747	1	8	373	80.0	80.0	1	0.0
13	12943	1	-1	301	-4.0	-4.0	1	0.0
14	17307	1	-144	365	-153.0	-153.0	1	0.0
15	16428	1	-1	81	-3.0	-3.0	1	0.0

3. 구매 취소 경향성


- 고객의 취소 패턴 파악하기
 - 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
 - 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
 - 취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data`에 통합하기
(취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE enhanced-ward-466622-g5.1st_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS total_transactions,
    COUNT(DISTINCT CASE WHEN InvoiceNo LIKE 'C%' THEN InvoiceNo END) AS cancel_frequency
  FROM enhanced-ward-466622-g5.1st_project.Data1
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(
  CASE
    WHEN t.total_transactions > 0 THEN (t.cancel_frequency * 100.0 / t.total_transactions)
    ELSE 0
  END, 2
) AS cancel_rate
FROM enhanced-ward-466622-g5.1st_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```
# SELECT *
FROM enhanced-ward-466622-g5.1st_project.user_data;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13747	1	8	373	80.0	80.0	1	0.0	1	0	0.0
2	17763	1	12	263	15.0	15.0	1	0.0	1	0	0.0
3	13703	1	10	318	100.0	100.0	1	0.0	1	0	0.0
4	13228	1	200	81	358.0	358.0	2	0.0	1	0	0.0
5	14642	1	76	58	96.0	96.0	2	0.0	1	0	0.0
6	16505	1	576	31	778.0	778.0	4	0.0	1	0	0.0
7	18019	1	7	51	38.0	38.0	5	0.0	1	0	0.0
8	18042	1	33	53	165.0	165.0	7	0.0	1	0	0.0
9	14271	1	88	225	112.0	112.0	10	0.0	1	0	0.0
10	16097	1	87	240	185.0	185.0	10	0.0	1	0	0.0
11	14935	1	935	297	1785.0	1785.0	10	0.0	1	0	0.0
12	17496	1	79	358	271.0	271.0	10	0.0	1	0	0.0
13	15667	1	240	39	301.0	301.0	13	0.0	1	0	0.0
14	17824	1	408	51	298.0	298.0	13	0.0	1	0	0.0
15	15783	1	212	10	246.0	246.0	14	0.0	1	0	0.0
16	17415	1	114	79	508.0	508.0	14	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : SQL을 사용해 데이터를 특정 기준에 따라서 분류하고, 분류된 데이터를 제거하거나 연산을 통해 새로운 정보를 습득하는 능력을 유지할 것

Problem : 데이터 값의 특이점이나 null 값을 분석해서 잘못된 데이터를 분류했지만, 아직도 불량 데이터가 남아 있음.

Try : 이를 보완하기 위해서 데이터 수집이나 전처리 과정에서 먼저 불량 데이터를 거르고, 휴리스틱 분석을 통해 불량 데이터를 색출하는 방법도 공부해야 할 것이다.