

고객을 세그멘테이션하자 [프로젝트]

5-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM modulabs_project.data
LIMIT 10;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536414	22139	null	56	2010-12-01 11:52:00 UTC	0.0	null	United Kingdom
2	536545	21134	null	1	2010-12-01 14:32:00 UTC	0.0	null	United Kingdom
3	536546	22145	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
4	536547	37509	null	1	2010-12-01 14:33:00 UTC	0.0	null	United Kingdom
5	536549	85226A	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
6	536550	85044	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
7	536552	20950	null	1	2010-12-01 14:34:00 UTC	0.0	null	United Kingdom
8	536553	37461	null	3	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
9	536554	84670	null	23	2010-12-01 14:35:00 UTC	0.0	null	United Kingdom
10	536589	21777	null	-10	2010-12-01 16:50:00 UTC	0.0	null	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
# [[YOUR QUERY]]
SELECT COUNT(*) AS count_Data
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	count_Data
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
```

[결과 이미지를 넣어주세요]

5-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

- 각 컬럼 별 누락된 값의 비율을 계산
 - 각 컬럼에 대해서 누락 값을 계산한 후, 계산된 누락 값을 UNION ALL을 통해 합치기

```
# [[YOUR QUERY]]
SELECT
  'InvoiceNo' AS column_name,
  ROUND(COUNTIF(InvoiceNo IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project.data

UNION ALL
```

```

SELECT
    'StockCode' AS column_name,
    ROUND(COUNTIF(StockCode IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project.data

UNION ALL

SELECT
    'Description' AS column_name,
    ROUND(COUNTIF(Description IS NULL) / COUNT(*) * 100, 2) AS missing_percentage
FROM modulabs_project.data

```

[결과 이미지를 넣어주세요]

행	column_name	missing_percentage
1	StockCode	0.0
2	InvoiceNo	0.0
3	Description	0.27

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM modulabs_project.data
WHERE StockCode = '85123A'
AND Description IS NOT NULL;

```

[결과 이미지를 넣어주세요]

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

결측치 처리

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

```

DELETE FROM 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;

```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 0개가 삭제되었습니다.

5-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```

WITH group_data AS (
  SELECT
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country,
    COUNT(*) AS row_count
  FROM modulabs_project.data
  GROUP BY
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
)
SELECT COUNT(*)
FROM group_data
WHERE row_count > 1;

```

[결과 이미지를 넣어주세요]

행	f0_
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 modulabs_project.data AS
SELECT DISTINCT *
FROM modulabs_project.data;

```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

5-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

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

```

SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data;

```

[결과 이미지를 넣어주세요]

행	f0_
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM modulabs_project.data
ORDER BY InvoiceNo
LIMIT 100;
```

[결과 이미지를 넣어주세요]

행	InvoiceNo
1	536365
2	536366
3	536367
4	536368
5	536369
6	536370
7	536371
8	536372
9	536373
10	536374
11	536375
12	536376
13	536377
14	536378
15	536380
16	536381
17	536382
18	536384
19	536385
20	536386
21	536387

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

```
SELECT *
FROM modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	지트	JSON	실행 세부정보	실행 그래프			
행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	CS75531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	CS58080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	CS58080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	CS54983	475908	PINK HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	CS54983	47590A	BLUE HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	CS39709	84978	HANDING HEART JAR FLIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
7	CS39709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
8	CS39709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
9	CS43820	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom
10	CS46858	21534	DAIRY MAID LARGE MILK JUG	-1	2011-03-17 14:24:00 UTC	4.95	14081	United Kingdom
11	CS46858	22839	3 TIER CAKE TIN GREEN AND ...	-1	2011-03-17 14:24:00 UTC	14.95	14081	United Kingdom
12	CS42263	22699	ROSES REGENCY TEACUP AN...	-1	2011-01-26 17:16:00 UTC	2.95	14849	United Kingdom
13	CS53534	21467	CHERRY CROCHET FOOD COV...	-1	2011-05-17 15:15:00 UTC	3.75	14849	United Kingdom
14	CS70996	23376	PACK OF 12 VINTAGE CHRIST...	-24	2011-10-13 12:02:00 UTC	0.39	14849	United Kingdom
15	CS70996	22909	SET OF 20 VINTAGE CHRISTM...	-12	2011-10-13 12:02:00 UTC	0.85	14849	United Kingdom
16	CS43818	22622	BOX OF VINTAGE ALPHABET B...	-2	2011-02-13 15:45:00 UTC	9.95	16897	United Kingdom

더 보기

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

```
SELECT
ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 1) AS canceled_perc
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	canceled_percentage
1	2.2

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	f0_
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 modulabs_project.data
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 문자열 내 숫자의 길이를 구해보기

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM project_name.modulabs_project.data
)

SELECT
  LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count,
  COUNT(*) AS stock_cnt
FROM UniqueStockCodes
GROUP BY number_count
ORDER BY stock_cnt DESC;
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON
id	number_count ▼	stock_cnt ▼	
1	5	3676	
2	0	7	
3	1	1	

- **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 modulabs_project.data
)
WHERE number_count <= 1;
```

[결과 이미지를 넣어주세요]

id	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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH char_count_data AS (
  SELECT
    StockCode,
    LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS char_length,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.data
)
SELECT
  ROUND(COUNTIF(number_count <= 1) * 100.0 / COUNT(*), 2) AS percentage
FROM char_count_data;
```

[결과 이미지를 넣어주세요]

id	percentage ▼
1	0.48

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM project_name.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM modulabs_project.data
```

```
WHERE StockCode IN ('BANK CHARGES', 'POST', 'D', 'M', 'CRUK')
      OR StockCode LIKE 'GIFT%'
      OR Quantity <= 0
      OR UnitPrice <= 0
    )
);
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 0개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

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

[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	FELTCRAFT PRINCESS CHARL...	408
2	HAND WARMER RED LOVE HE...	386
3	POTTERING IN THE SHED MET...	377
4	PACK OF 12 HEARTS DESIGN T...	356
5	SET/6 RED SPOTTY PAPER PL...	328
6	SET OF 60 VINTAGE LEAF CAK...	312
7	WASHROOM METAL SIGN	296
8	SET/6 RED SPOTTY PAPER CU...	292
9	CHILDRENS APRON APPLES D...	279
10	VINTAGE SNAKES & LADDERS	268
11	JAZZ HEARTS PURSE NOTEBO...	258
12	ROBOT BIRTHDAY CARD	258
13	PICTURE DOMINOES	249
14	SET OF 3 CAKE TINS SKETCHB...	247
15	SWALLOWS GREETING CARD	243
16	TRADITIONAL PICK UP STICKS...	241
17	FANCY FONTS BIRTHDAY WRAP	230
18	HEART IVORY TRELLIS SMALL	226
19	TRADITIONAL NAUGHTS & CR...	221

더보기

- 대소문자가 혼합된 Description이 있는지 확인하기

```
SELECT DISTINCT Description
FROM project_name.modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]');
```

[결과 이미지를 넣어주세요]

번호	Description
1	NUMBER TILE VINTAGE FONT ...
2	FLOWERS HANDBAG blue and ...
3	ESSENTIAL BALM 3.5g TIN IN ...
4	High Resolution Image
5	POLYESTER FILLER PAD 45x30...
6	POLYESTER FILLER PAD 30CM...
7	POLYESTER FILLER PAD 65CM...
8	POLYESTER FILLER PAD 60x40...
9	THE KING GIFT BAG 25x24x12...

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM modulabs_project.data
WHERE
  LOWER(Description) LIKE '%service%'
  OR LOWER(Description) LIKE '%repair%'
  OR LOWER(Description) LIKE '%postage%'
  OR LOWER(Description) LIKE '%delivery%'
  OR LOWER(Description) LIKE '%fee%'
  OR LOWER(Description) LIKE '%charge%';
```

[결과 이미지를 넣어주세요]

i 이 문으로 data의 행 497개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  AVG(UnitPrice) AS avg_price
FROM modulabs_project.data
```

[결과 이미지를 넣어주세요]

행	min_price	max_price	avg_price
1	0.04	165.0	2.137986720099...

- 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 modulabs_project.data
WHERE UnitPrice = 0;
```

[결과 이미지를 넣어주세요]

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	0	null	null	null

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

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT *
FROM modulabs_project.data
WHERE UnitPrice > 0;
```

[결과 이미지를 넣어주세요]

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

5-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	InvoiceDay	InvoiceNo	StockDate	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-31	54020	10002	120	2011-01-31 09:57:00 UTC	0.85	12731	France	INFLATABLE POLITICAL GLOBE
2	2011-01-06	540277	10002	60	2011-01-06 12:18:00 UTC	0.85	14238	United Kingdom	INFLATABLE POLITICAL GLOBE
3	2011-01-20	541698	10002	18	2011-01-20 19:16:00 UTC	0.85	14713	United Kingdom	INFLATABLE POLITICAL GLOBE
4	2011-04-01	548606	10002	120	2011-04-01 11:10:00 UTC	0.85	12731	France	INFLATABLE POLITICAL GLOBE
5	2011-04-15	550272	10002	62	2011-04-15 12:14:00 UTC	0.85	18079	United Kingdom	INFLATABLE POLITICAL GLOBE
6	2011-05-13	540194	10002	11	2011-05-13 13:36:00 UTC	0.85	15246	United Kingdom	INFLATABLE POLITICAL GLOBE
7	2011-01-30	542610	10002	14	2011-01-30 14:55:00 UTC	0.85	13148	United Kingdom	INFLATABLE POLITICAL GLOBE
8	2011-11-21	577801	10080	26	2011-11-21 17:04:00 UTC	0.39	17629	United Kingdom	GROOVY CACTUS INFLATABLE
9	2011-03-04	545637	10120	11	2011-03-04 12:16:00 UTC	0.21	15514	United Kingdom	DOGGY RUBBER
10	2011-08-22	564049	10120	30	2011-08-22 13:30:00 UTC	0.21	17585	United Kingdom	DOGGY RUBBER
11	2011-02-20	544460	10120	30	2011-02-20 14:01:00 UTC	0.21	16931	United Kingdom	DOGGY RUBBER
12	2011-08-11	560394	10125	40	2011-08-11 14:43:00 UTC	0.85	17644	United Kingdom	MINI FUNKY DESIGN TAPES
13	2011-10-02	560223	10125	7	2011-10-02 13:49:00 UTC	0.85	16283	United Kingdom	MINI FUNKY DESIGN TAPES
14	2011-07-18	560398	10125	100	2011-07-18 14:10:00 UTC	0.85	12731	France	MINI FUNKY DESIGN TAPES
15	2011-01-06	540263	10125	40	2011-01-06 10:03:00 UTC	0.42	18062	United Kingdom	MINI FUNKY DESIGN TAPES
16	2011-09-30	569097	10125	150	2011-09-30 12:06:00 UTC	0.85	12731	France	MINI FUNKY DESIGN TAPES
17	2011-05-11	552851	10125	100	2011-05-11 15:07:00 UTC	0.85	12731	France	MINI FUNKY DESIGN TAPES

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  (SELECT MAX(InvoiceDate) FROM modulabs_project.data) AS most_recent_date,
  DATE(InvoiceDate) AS InvoiceDay,
```

★

```
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

#	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09 12:31:00 UTC	2011-01-31	542629	10002	120	2011-01-31 09:57:00 UTC	0.85	12731	France
2	2011-12-09 12:31:00 UTC	2011-01-06	540277	10002	40	2011-01-06 12:16:00 UTC	0.85	14258	United Kingdom
3	2011-12-09 12:31:00 UTC	2011-01-20	541088	10002	18	2011-01-20 19:16:00 UTC	0.85	14713	United Kingdom
4	2011-12-09 12:31:00 UTC	2011-04-01	548066	10002	120	2011-04-01 11:10:00 UTC	0.85	12731	France
5	2011-12-09 12:31:00 UTC	2011-04-15	550272	10002	62	2011-04-15 12:14:00 UTC	0.85	18079	United Kingdom
6	2011-12-09 12:31:00 UTC	2011-01-13	541084	10002	11	2011-01-13 13:36:00 UTC	0.85	13246	United Kingdom
7	2011-12-09 12:31:00 UTC	2011-01-30	542610	10002	14	2011-01-30 14:05:00 UTC	0.85	13148	United Kingdom
8	2011-12-09 12:31:00 UTC	2011-11-21	577801	10080	26	2011-11-21 17:04:00 UTC	0.39	17629	United Kingdom
9	2011-12-09 12:31:00 UTC	2011-08-04	545037	10120	11	2011-08-04 12:16:00 UTC	0.21	15514	United Kingdom
10	2011-12-09 12:31:00 UTC	2011-08-22	564549	10120	30	2011-08-22 13:30:00 UTC	0.21	17585	United Kingdom
11	2011-12-09 12:31:00 UTC	2011-02-20	544460	10120	30	2011-02-20 14:01:00 UTC	0.21	16931	United Kingdom
12	2011-12-09 12:31:00 UTC	2011-08-11	563054	10125	40	2011-08-11 14:53:00 UTC	0.85	17644	United Kingdom
13	2011-12-09 12:31:00 UTC	2011-10-02	569223	10125	7	2011-10-02 13:49:00 UTC	0.85	16283	United Kingdom
14	2011-12-09 12:31:00 UTC	2011-07-18	560398	10125	100	2011-07-18 14:10:00 UTC	0.85	12731	France
15	2011-12-09 12:31:00 UTC	2011-01-06	540263	10125	40	2011-01-06 10:05:00 UTC	0.42	18062	United Kingdom
16	2011-12-09 12:31:00 UTC	2011-09-30	560907	10125	150	2011-09-30 12:06:00 UTC	0.85	12731	France
17	2011-12-09 12:31:00 UTC	2011-05-11	523551	10125	100	2011-05-11 15:07:00 UTC	0.85	12731	France
18	2011-12-09 12:31:00 UTC	2011-03-31	548409	10125	40	2011-03-31 10:27:00 UTC	0.85	12731	France
19	2011-12-09 12:31:00 UTC	2011-11-16	575672	10125	60	2011-11-16 11:59:00 UTC	0.85	12731	France
20	2011-12-09 12:31:00 UTC	2010-12-15	538991	10125	60	2010-12-15 11:53:00 UTC	0.42	17511	United Kingdom

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	InvoiceDay
1	12731	2011-11-16
2	14258	2011-11-30
3	14713	2011-11-30
4	18079	2011-10-25
5	13246	2011-11-21
6	13148	2011-11-10
7	17629	2011-12-04
8	15514	2011-11-24
9	17585	2011-08-22
10	16931	2011-12-04
11	17644	2011-12-08
12	16283	2011-12-04
13	18062	2011-01-06

- 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 modulabs_project.data
    GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

일련	CustomerID	recency
1	15298	2
2	12770	210
3	15984	4
4	12916	138
5	17052	28
6	15042	199
7	13493	275
8	16000	2
9	15185	70
10	14697	218
11	17062	313
12	17786	85
13	18109	14
14	16750	44
15	13171	269

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_r AS
SELECT
  CustomerID,
  MAX(Date(InvoiceDate)) AS last_purchase_date,
  COUNT(DISTINCT InvoiceNo) AS frequency,
  SUM(Quantity * UnitPrice) AS monetary,
  EXTRACT(DAY FROM (SELECT MAX(Date(InvoiceDate)) FROM midyear-module-447402-s8.modulabs_project.data)
FROM midyear-module-447402-s8.modulabs_project.data
WHERE Quantity > 0 AND UnitPrice > 0
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 project_name.modulabs_project.data
GROUP BY CustomerID
ORDER BY purchase_cnt DESC;
```

[결과 이미지를 넣어주세요]

일련	CustomerID	purchase_cnt
1	14911	150
2	12748	148
3	17841	118
4	14606	89
5	15311	83
6	13089	52
7	14646	46
8	14527	45

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
FROM modulabs_project.data
WHERE Quantity > 0
GROUP BY CustomerID
ORDER BY item_cnt DESC;
```

[결과 이미지를 넣어주세요]

행	CustomerID	item_cnt
1	14298	22314
2	14646	20710
3	13694	18115
4	14911	13543
5	17511	13273
6	18102	11164
7	12415	8116
8	16308	8000
9	17450	7326
10	14156	6394

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

```
CREATE OR REPLACE TABLE `modulabs_project.user_rf` AS

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

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
    SELECT
        CustomerID,
        SUM(Quantity) AS item_cnt
    FROM `modulabs_project.data`
    WHERE Quantity > 0
    GROUP BY CustomerID
)

-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
    pc.CustomerID,
    pc.purchase_cnt,
    ic.item_cnt,
    ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
    ON pc.CustomerID = ic.CustomerID
JOIN project_name.modulabs_project.user_r AS ur
    ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity)) AS user_total
FROM modulabs_project.data
WHERE Quantity > 0 AND UnitPrice > 0
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12731	1640.0
2	14258	1507.0
3	14713	288.0
4	18079	792.0
5	13246	171.0
6	13148	625.0
7	17629	283.0
8	15514	163.0
9	17585	264.0
10	16931	643.0
11	17644	207.0
12	16283	212.0
13	18062	48.0
14	17511	9458.0
15	12864	17.0

- 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 `midyear-module-447402-s8.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, 1) AS user_average
FROM `midyear-module-447402-s8.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM `midyear-module-447402-s8.modulabs_project.data`
  WHERE Quantity > 0 AND UnitPrice > 0
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *  
FROM `midyear-module-447402-s8.modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	92	0	88.1	88.1
2	18010	1	29	256	50.0	50.0
3	13368	1	4	256	15.0	15.0
4	12792	1	48	256	70.0	70.0
5	15083	1	2	256	25.5	25.5
6	14569	1	6	1	10.9	10.9
7	17144	1	54	1	66.8	66.8
8	15520	1	24	1	34.4	34.4
9	13436	1	35	1	42.8	42.8
10	16777	1	4	257	11.4	11.4
11	13357	1	48	257	75.3	75.3
12	14476	1	16	257	33.6	33.6
13	14204	1	25	2	50.6	50.6
14	14087	1	112	2	54.9	54.9
15	16000	1	620	2	855.6	855.6
16	15471	1	64	2	72.5	72.5
17	16569	1	50	3	21.0	21.0

더보기

5-8. 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;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.


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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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;
```

[결과 이미지를 넣어주세요]

 이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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 `midyear-module-447402-s8.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END) AS cancel_frequency
  FROM `midyear-module-447402-s8.modulabs_project.data`
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(SAFE_DIVIDE(t.cancel_frequency, t.total_transactions), 2) AS
FROM `midyear-module-447402-s8.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

i 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
    u.CustomerID,
    u.recency,
    u.purchase_cnt AS frequency,
    u.user_total AS monetary,
    u.user_average,
    u.unique_products,
    u.average_interval,
    u.total_transactions,
    u.cancel_frequency,
    u.cancel_rate,
    ROUND(AVG(d.UnitPrice), 2) AS avg_unit_price,
    MAX(d.UnitPrice) AS max_unit_price,
    MIN(d.UnitPrice) AS min_unit_price,
    COUNT(DISTINCT d.StockCode) AS distinct_products,
    COUNT(DISTINCT EXTRACT(DAYOFWEEK FROM d.InvoiceDate)) AS active_days,
    MAX(d.InvoiceDate) AS last_purchase_date,
    MIN(d.InvoiceDate) AS first_purchase_date,
    DATE_DIFF(MAX(d.InvoiceDate), MIN(d.InvoiceDate), DAY) AS customer_lifetime_days
FROM `midyear-module-447402-s8.modulabs_project.user_data` AS u
LEFT JOIN `midyear-module-447402-s8.modulabs_project.data` AS d
ON u.CustomerID = d.CustomerID
GROUP BY
    u.CustomerID, u.recency, u.purchase_cnt, u.user_total, u.user_average,
    u.unique_products, u.average_interval, u.total_transactions,
    u.cancel_frequency, u.cancel_rate
ORDER BY u.user_total DESC
```

[결과 이미지를 넣어주세요]

rank	CustomerID	recency	frequency	monetary	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate	avg_unit_price	max_unit_price
1	18102	1	31	63422.8	2045.9	30	4.58	79	0	0.0	6.4	52.77
2	14646	2	46	20843.5	453.1	144	1.36	247	0	0.0	1.85	110.0
3	17450	8	24	17171.8	715.5	22	6.98	49	0	0.0	2.35	8.0
4	14298	30	29	14551.5	501.8	306	0.65	499	0	0.0	1.24	16.95
5	14911	1	150	13207.0	88.0	477	0.38	811	0	0.0	2.12	39.95
6	13694	3	31	10035.2	323.7	113	2.47	145	0	0.0	1.13	7.95
7	17511	2	27	9457.5	350.3	105	2.2	163	0	0.0	1.28	16.95
8	12415	65	8	8479.3	1059.9	65	3.54	77	0	0.0	1.8	14.95
9	14096	4	17	7587.6	446.3	204	0.08	1130	0	0.0	3.2	24.96
10	14156	9	32	7216.8	225.5	118	2.48	141	0	0.0	2.37	29.95
11	15749	235	2	6043.0	3022.5	1	97.0	2	0	0.0	2.53	2.05
12	12742	0	148	5993.3	40.5	549	0.3	1045	0	0.0	2.06	29.95
13	17941	1	118	5168.7	43.8	348	0.24	1326	0	0.0	1.96	12.75
14	14088	10	9	4995.4	555.0	66	3.44	91	0	0.0	6.77	110.0
15	13089	4	52	4029.4	77.5	124	1.57	219	0	0.0	1.67	14.95
16	13081	11	9	3302.0	366.9	112	1.8	198	0	0.0	1.25	8.5
17	16684	4	11	3269.8	297.3	13	17.35	21	0	0.0	1.96	10.95

전보기