고객을 세그먼테이션하자 [프로젝트] 윤가람

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
SELECT *
FROM `stone-nuance-228223.modulabs_project.data`
LIMIT 10;
```

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_rows
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	total_rows
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 COUNT_UnitPrice,
  COUNT(CustomerID) AS COUNT_CustomerID,
  COUNT(Country) AS COUNT_Country
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	COUNT_InvoiceNo	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_Country
1	541909	541909	540455	541909	541909	541909	406829	541909

11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

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

```
SELECT
   'Description' AS column_name,
   ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `stone-nuance-228223.modulabs_project.data`

UNION ALL

SELECT
   'CustomerID' AS column_name,
   ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM `stone-nuance-228223.modulabs_project.data`
```

행	column_name	missing_percentage
1	CustomerID	24.93
2	Description	0.27

결측치 처리 전략

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

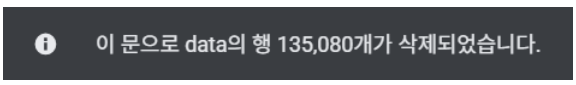
```
SELECT DISTINCT Description
FROM `stone-nuance-228223.modulabs_project.data`
WHERE StockCode = '85123A'
```

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**

```
DELETE FROM `stone-nuance-228223.modulabs_project.data`
WHERE CustomerID IS NULL
  OR Description 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**

```
SELECT *
FROM `stone-nuance-228223.modulabs_project.data`
GROUP BY
  InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
HAVING COUNT(*) > 1;
```

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

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	571034	23494	VINTAGE DOILY DELUXE SEWING KIT	3	2011-10-13 12:47:00 UTC	5.95	12359	Cyprus
2	571034	23245	SET OF 3 REGENCY CAKE TINS	4	2011-10-13 12:47:00 UTC	4.95	12359	Cyprus
3	571034	23239	SET OF 4 KNICK KNACK TINS POPPIES	6	2011-10-13 12:47:00 UTC	4.15	12359	Cyprus
4	538826	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	1	2010-12-14 12:58:00 UTC	3.75	12370	Cyprus
5	577228	22270	HAPPY EASTER HANGING DECORATION	1	2011-11-18 12:07:00 UTC	3.75	12391	Cyprus

페이지당 결과 수: 50 1 – 50 (전체 4837행)

중복값 처리

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

```
CREATE OR REPLACE TABLE `stone-nuance-228223.modulabs_project.data` AS
SELECT DISTINCT *
FROM `stone-nuance-228223.modulabs_project.data`;
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

행	row_count
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

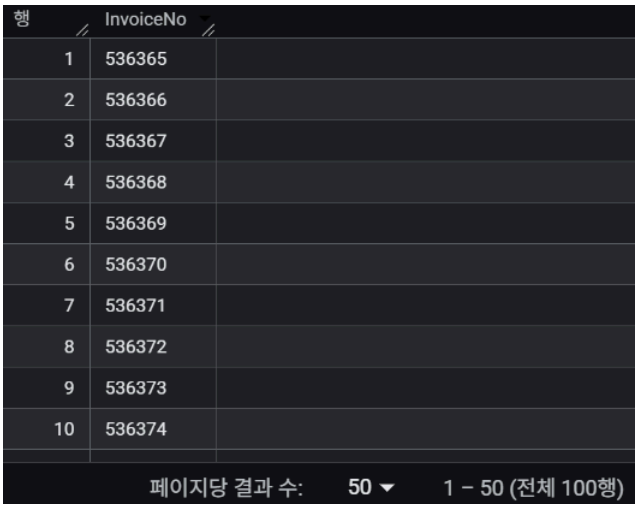
- **고유(unique)한** `InvoiceNo` **의 개수를 출력하기**

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	unique_invoice_c...
1	22190

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

```
SELECT DISTINCT InvoiceNo
FROM `stone-nuance-228223.modulabs_project.data`
ORDER BY InvoiceNo
LIMIT 100;
```

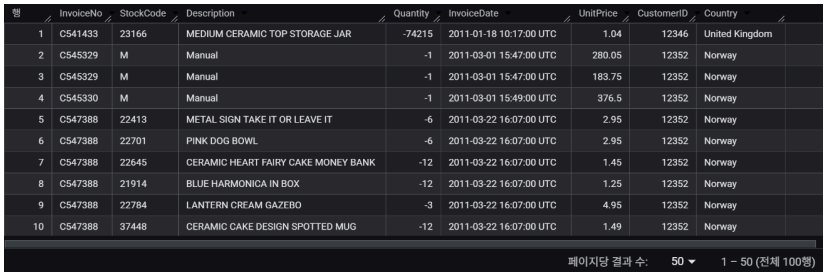


행	InvoiceNo
1	536365
2	536366
3	536367
4	536368
5	536369
6	536370
7	536371
8	536372
9	536373
10	536374

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT *
FROM `stone-nuance-228223.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

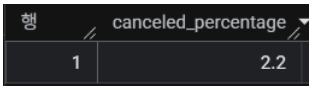


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22413	METAL SIGN TAKE IT OR LEAVE IT	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
6	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
7	C547388	22645	CERAMIC HEART FAIRY CAKE MONEY BANK	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
8	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
9	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
10	C547388	37448	CERAMIC CAKE DESIGN SPOTTED MUG	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway

페이지당 결과 수: 50 1 – 50 (전체 100행)

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*) * 100, 1) AS canceled_percentage
FROM `stone-nuance-228223.modulabs_project.data`;
```



행	canceled_percentage
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode`의 개수를 출력하기**

```
SELECT COUNT(DISTINCT StockCode) AS unique_stockcode_count
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	unique_stockcode_count
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 `stone-nuance-228223.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

행	StockCode	sell_cnt
1	85123A	2065
2	22423	1894
3	85099B	1659
4	47566	1409
5	84879	1405
6	20725	1346
7	22720	1224
8	POST	1196
9	22197	1110
10	23203	1108

- **`StockCode`의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값**들에는 어떤 코드들이 들어가 있는지 출력하기

```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `stone-nuance-228223.modulabs_project.data`
)
WHERE number_count = 0 or number_count = 1;
```

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

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

```
SELECT
  ROUND(100 * SUM(CASE WHEN LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) IN (0, 1)
                THEN 1 ELSE 0 END) / COUNT(*), 2) AS percentage
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	percentage
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM `stone-nuance-228223.modulabs_project.data`
WHERE StockCode IN (
  SELECT StockCode
  FROM (
    SELECT StockCode,
          LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
    FROM `stone-nuance-228223.modulabs_project.data`
  )
  WHERE number_count = 0 OR number_count = 1
);
```

이 문으로 data의 행 1,915개가 삭제되었습니다.

`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM `stone-nuance-228223.modulabs_project.data`
GROUP BY Description
```

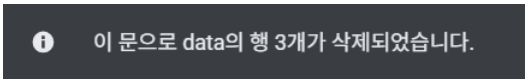
```
ORDER BY description_cnt DESC
LIMIT 30;
```

행	Description	description_cnt
1	WHITE HANGING HEART T-LIGHT HOLDER	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORNAMENT	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224

페이지당 결과 수: 50 1 – 30 (전체 30행)

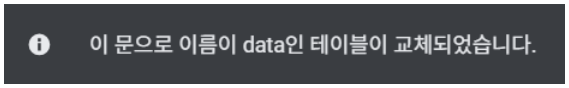
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM `stone-nuance-228223.modulabs_project.data`
WHERE
Description = 'Next Image' or Description = 'High Resolution Image';
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE `stone-nuance-228223.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `stone-nuance-228223.modulabs_project.data`;
```



UnitPrice 살펴보기

- **UnitPrice의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
  MIN(UnitPrice) AS min_price,
  MAX(UnitPrice) AS max_price,
  ROUND(AVG(UnitPrice), 2) AS avg_price
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.91

- **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,
  ROUND(AVG(Quantity), 2) AS avg_quantity
FROM `stone-nuance-228223.modulabs_project.data`
WHERE UnitPrice = 0;
```

행	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.52

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

```
CREATE OR REPLACE TABLE `stone-nuance-228223.modulabs_project.data` AS
SELECT *
FROM `stone-nuance-228223.modulabs_project.data`
WHERE UnitPrice <> 0;
```

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

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

```
SELECT *, DATE(InvoiceDate) AS InvoiceDay
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description	InvoiceDay
1	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR	2011-01-18
2	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR	2011-01-18
3	537626	84997B	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	RED 3 PIECE RETROSPOT CUTLERY SET	2010-12-07
4	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	RED DRAWER KNOB ACRYLIC EDWARDIAN	2010-12-07
5	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PLAYING CARDS	2010-12-07

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
MAX(InvoiceDate) OVER() AS most_recent_date,
DATE(InvoiceDate) AS InvoiceDay,
*
FROM `stone-nuance-228223.modulabs_project.data`;
```

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-12-09 12:50:00 UTC	2011-06-10	556415	23008	6	2011-06-10 12:19:00 UTC	14.95	12409	Switzerland	DOLLY GIRL BABY GIFT SET
2	2011-12-09 12:50:00 UTC	2011-02-15	543989	22841	12	2011-02-15 09:52:00 UTC	6.95	12415	Australia	ROUND CAKE TIN VINTAGE GREEN
3	2011-12-09 12:50:00 UTC	2011-08-18	563614	23345	200	2011-08-18 08:51:00 UTC	1.08	12415	Australia	DOLLY GIRL BEAKER
4	2011-12-09 12:50:00 UTC	2011-01-12	540972	22236	12	2011-01-12 14:13:00 UTC	10.95	12437	France	CAKE STAND 3 TIER MAGIC GARDEN
5	2011-12-09 12:50:00 UTC	2011-07-31	561898	82486	2	2011-07-31 15:25:00 UTC	8.95	12456	Switzerland	WOOD S/3 CABINET ANT WHITE FINISH

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM `stone-nuance-228223.modulabs_project.data`
GROUP BY CustomerID;
```

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

- **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 stone-nuance-228223.modulabs_project.data
  GROUP BY CustomerID
);
```

행	CustomerID	recency
1	12354	232
2	12388	15
3	12452	16
4	12633	58
5	12688	113
6	12830	37
7	12849	31

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

```
CREATE OR REPLACE TABLE stone-nuance-228223.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 stone-nuance-228223.modulabs_project.data
  GROUP BY CustomerID
);
```

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

행	CustomerID	recency
1	12662	0
2	16705	0
3	13426	0
4	12518	0
5	13777	0
6	12748	0
7	15804	0
8	12526	0
9	14422	0
10	17754	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM stone-nuance-228223.modulabs_project.data
GROUP BY CustomerID;
```

행	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM stone-nuance-228223.modulabs_project.data
GROUP BY CustomerID;
```

행	CustomerID	item_cnt
1	12346	0
2	12347	2458
3	12348	2332
4	12349	630
5	12350	196

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

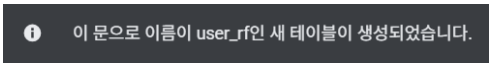
```
CREATE OR REPLACE TABLE stone-nuance-228223.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM stone-nuance-228223.modulabs_project.data
  GROUP BY CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM stone-nuance-228223.modulabs_project.data
  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 stone-nuance-228223.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```



행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	15520	1	314	1
3	13298	1	96	1
4	13436	1	76	1
5	14569	1	79	1
6	15195	1	1404	2
7	15471	1	256	2
8	14204	1	72	2
9	15318	1	642	3
10	15992	1	17	3

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

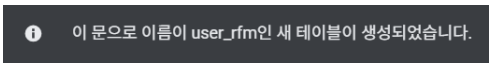
```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice * Quantity), 1) AS user_total
FROM
  stone-nuance-228223.modulabs_project.data
GROUP BY
  CustomerID
ORDER BY
  user_total DESC;
```

행	CustomerID	user_total
1	14646	278778.0
2	18102	259657.3
3	17450	189575.5
4	14911	128768.2
5	12415	123638.2
6	14156	113685.8
7	17511	88138.2
8	16684	65920.1

- **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 stone-nuance-228223.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 stone-nuance-228223.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID,
    SUM(UnitPrice * Quantity) AS user_total
  FROM stone-nuance-228223.modulabs_project.data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.55	794.5
2	13298	1	96	1	360.0	360.0
3	14569	1	79	1	227.39	227.4
4	15520	1	314	1	343.5	343.5
5	13436	1	76	1	196.8900000...	196.9
6	15195	1	1404	2	3861.0	3861.0
7	15471	1	256	2	454.48	454.5
8	14204	1	72	2	150.61	150.6
9	12478	1	233	3	545.9900000...	546.0
10	16528	1	171	3	244.4099999...	244.4

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT
  CustomerID,
  purchase_cnt,
  item_cnt,
  recency,
  user_total,
  user_average
FROM
  stone-nuance-228223.modulabs_project.user_rfm
ORDER BY
  user_average DESC;
```

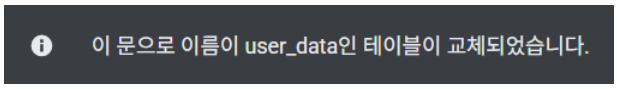
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	15098	4	61	182	39619.5	9904.9
2	12357	1	2708	33	6207.6699999999992	6207.7
3	15749	4	9014	235	21535.9	5384.0
4	12415	24	76946	24	123638.17999999988	5151.6
5	12688	1	3028	113	4873.8100000000013	4873.8
6	12590	2	4278	190	9338.380000000001	4669.2
7	12752	1	2262	81	4366.7800000000007	4366.8
8	18102	60	64124	0	259657.30000000008	4327.6
9	18251	1	7824	87	4314.7199999999993	4314.7

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 stone-nuance-228223.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM stone-nuance-228223.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM stone-nuance-228223.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



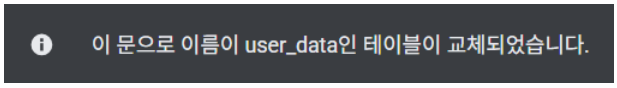
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	17948	1	144	147	358.5600000...	358.6	1
2	17956	1	1	249	12.75	12.8	1
3	12814	1	48	101	85.92	85.9	1
4	13185	1	12	267	71.4	71.4	1
5	16881	1	600	66	432.0	432.0	1
6	16093	1	20	106	17.0	17.0	1
7	16995	1	-1	372	-1.25	-1.3	1
8	17923	1	50	282	207.5000000...	207.5	1
9	14119	1	-2	354	-19.9	-19.9	1
10	17752	1	192	359	80.64	80.6	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 stone-nuance-228223.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
      stone-nuance-228223.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM stone-nuance-228223.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	15700	3	328	172	664.0299999...	221.3	85	0.01
2	15154	2	85	74	224.7300000...	112.4	66	0.01
3	12607	2	0	58	-7.28306304...	0.0	101	0.01
4	17509	11	3508	57	6069.240000...	551.7	75	0.01
5	14096	17	16336	4	53258.42999...	3132.8	1118	0.02
6	13521	3	715	1	1070.469999...	356.8	312	0.02
7	14893	2	799	9	1237.850000...	618.9	72	0.03
8	12508	2	271	26	397.88	198.9	57	0.03
9	15427	3	967	33	1430.539999...	476.8	233	0.04
10	17841	169	22613	1	39861.49000...	235.9	1330	0.04

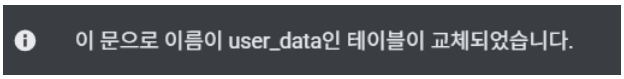
3. 구매 취소 경향성

- **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 stone-nuance-228223.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNTIF(STARTS_WITH(InvoiceNo, 'C')) AS cancel_frequency
  FROM stone-nuance-228223.modulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)
```

```
SELECT u.*, t.* EXCEPT(CustomerID), ROUND(COALESCE(t.cancel_frequency, 0) / t.total_transactions, 2) AS cancel_rate
FROM stone-nuance-228223.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	15562	1	39	351	134.55	134.6	1	0.0	1	0	0.0
2	14424	1	48	17	322.08	322.1	1	0.0	1	0	0.0
3	15316	1	100	326	165.0	165.0	1	0.0	1	0	0.0
4	16061	1	-1	269	-29.95	-29.9	1	0.0	1	1	1.0
5	17443	1	504	219	534.24	534.2	1	0.0	1	0	0.0
6	16344	1	18	158	101.100...	101.1	1	0.0	2	0	0.0
7	15510	1	2	330	250.0	250.0	1	0.0	1	0	0.0
8	14679	1	-1	371	-2.55	-2.5	1	0.0	1	1	1.0
9	13302	1	5	155	63.75	63.8	1	0.0	1	0	0.0
10	18133	1	1350	212	931.499...	931.5	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
  stone-nuance-228223.modulabs_project.user_data
ORDER BY
  CustomerID;
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	12346	2	0	325	0.0	0.0	1	0.0	2	1	0.5
2	12347	7	2458	2	4309.9...	615.7	103	2.0	182	0	0.0
3	12348	4	2332	75	1437.2...	359.3	21	10.85	27	0	0.0
4	12349	1	630	18	1457.5...	1457.6	72	0.0	72	0	0.0
5	12350	1	196	310	294.40...	294.4	16	0.0	16	0	0.0
6	12352	8	463	36	1265.41	158.2	57	3.11	84	7	0.08
7	12353	1	20	204	89.0	89.0	4	0.0	4	0	0.0
8	12354	1	530	232	1079.4	1079.4	58	0.0	58	0	0.0
9	12355	1	240	214	459.40...	459.4	13	0.0	13	0	0.0
10	12356	3	1573	22	2487.4...	829.1	52	5.3	58	0	0.0

페이지당 결과 수: 50 1 – 50 (전체 4362행)

회고

Keep : user_data를 이용하여 고객들을 분류하고 그에 맞는 판매 전략을 세울 수 있다. 고객 세그멘테이션을 하기 위해서는 구매 최신성(Recency), 구매 빈도(Frequency), 총 구매 금액, 거래당 평균 거래금액(Monetary), 구매하는 제품의 다양성, 평균 구매 주기, 구매 취소율 등 고객에 대한 행동 패턴을 확인할 수 있는 여러 지표들을 활용해본다.

Problem : 다양한 속성 정보들을 기반으로 유저들을 세분화하는 데 RFM 분석 프레임워크만으로는 한계가 있다.

Try : 유저 그룹의 심층 분석을 위해 클러스터링 알고리즘을 활용해본다.