

찬규

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM continual-nomad-479202-p8.modulabs_project.data  
LIMIT 10
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGH...	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

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM continual-nomad-479202-p8.modulabs_project.data
```

행	fo_
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 continual-nomad-479202-p8.test.data
```

행	COUNT_InvoiceNO	COUNT_StockCode	COUNT_Descripti...	COUNT_Quantity	COUNT_InvoiceD...	COUNT_UnitPrice	COUNT_Custome...	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 ALL**을 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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.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 continual-nomad-479202-p8.test.data
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 continual-nomad-479202-p8.test.data

-- 다른 방법
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
    SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data UNION ALL
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM continual-nomad-479202-p8.test.data
) AS column_data;
```

[다른 방법으로 했습니다]

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

결측치 처리 전략

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

```
SELECT Description
FROM continual-nomad-479202-p8.test.data
WHERE StockCode = '85123A'
GROUP BY Description
```

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

결측치 처리

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

```
DELETE FROM continual-nomad-479202-p8.test.data
WHERE CustomerID IS NULL OR Description IS NULL
```

쿼리 결과			
작업 정보	결과	실행 세부정보	실행 그래프
<p>이 문으로 data의 행 135,080개가 삭제되었습니다.</p>			

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
WITH temp AS(
SELECT *, COUNT(*) AS cnt_row
FROM continual-nomad-479202-p8.test.data
GROUP BY InvoiceNO, StockCode, Description, QUantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING cnt_row > 1
)
-- 중복개수 확인
SELECT COUNT(*) FROM temp
```

행	fo_
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 continual-nomad-479202-p8.test.data AS
SELECT DISTINCT *
FROM continual-nomad-479202-p8.test.data
```

```

83  -- 중복값 처리
84  CREATE OR REPLACE TABLE continual-nomad-479202-p8.test.data AS
85  SELECT DISTINCT *
86  FROM continual-nomad-479202-p8.test.data
87
88  -- 중복값 처리 후 남은 데이터 행 개수 파악
89  SELECT COUNT(*)
90  FROM continual-nomad-479202-p8.test.data
91
92  -- 데이터 전처리(3) : 오류값 처리

```

쿼리 완료됨

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 [결과](#) 실행 세부정보 실행 그래프

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

중복값 처리 후 남은 데이터 행 개수 파악

행	fo_
1	401604

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNO)
FROM continual-nomad-479202-p8.test.data
```

행	fo_
1	22190

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

```
SELECT DISTINCT InvoiceNO
FROM continual-nomad-479202-p8.test.data
LIMIT 100
```

행	InvoiceNO
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180

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

```
SELECT *
FROM continual-nomad-479202-p8.test.data
WHERE InvoiceNO LIKE "C%"
LIMIT 100
```

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	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
6	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
7	C547388	84050	PINK HEART SHAPE EGG FRYN...	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
8	C547388	37448	CERAMIC CAKE DESIGN SPOTT...	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway

- 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_ratio
FROM continual-nomad-479202-p8.test.data
```

행	canceled_ratio
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM continual-nomad-479202-p8.test.data
```

행	fo_
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 continual-nomad-479202-p8.test.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

쿼리 결과			
작업 정보		결과	시작화
		JSON	실행 세부정보
		실행 그래프	
행	StockCode	sell_cnt	
1	85123A	2065	
2	22423	1694	
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 continual-nomad-479202-p8.test.data
)
WHERE number_count = 1 OR number_count = 0
```

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(*) / (SELECT COUNT(*) FROM continual-nomad-479202-p8.test.data) * 100, 2) AS ratio
FROM continual-nomad-479202-p8.test.data
GROUP BY StockCode
HAVING StockCode IN ('POST','D','C2','M','BANK CHARGES','PADS','DOT','CRUK')
```

행	StockCode	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

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM continual-nomad-479202-p8.test.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM continual-nomad-479202-p8.test.data
  WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')
);
```

104 -- 제품과 관련되지 않은 거래 기록을 제거

```
65 DELETE FROM continual-nomad-479202-p8.test.data
166 WHERE StockCode IN (
167   SELECT DISTINCT StockCode
168   FROM continual-nomad-479202-p8.test.data
169   WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')
170 );
171
172 -- Description 살펴보기
173 -- 고유한 Description 별 출현 빈도를 계산하고 상위 30개 출력
```

쿼리 완료됨

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

이 문으로 data의 행 1,915개가 삭제되었습니다.

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기

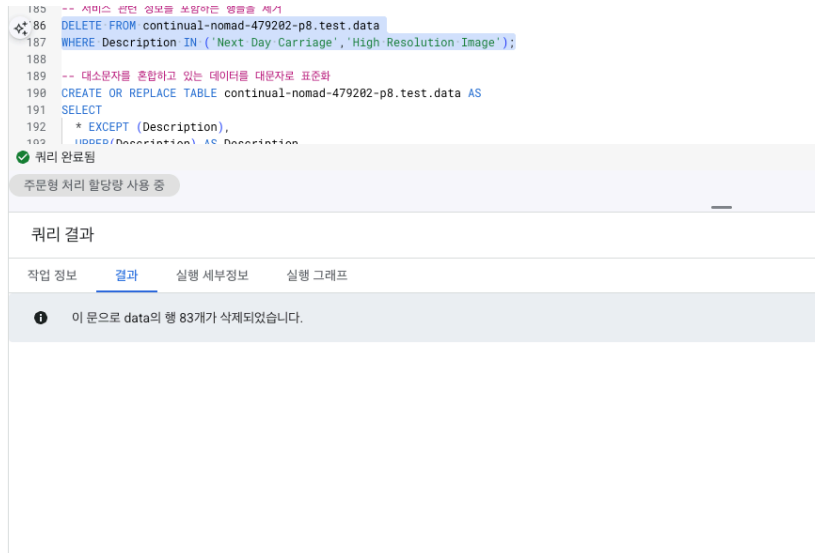
```
SELECT Description, COUNT(*) AS description_cnt
FROM continual-nomad-479202-p8.test.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30
```

쿼리 결과		
작업 정보	결과	실행 세부정보
Description	description_cnt	
1 WHITE HANGING HEART TAILG...	2058	
2 REGENCY CAKESTAND 3 TIER	1804	
3 JUMBO BAG RED RETROSPOT	1659	
4 PARTY BUNTING	1499	
5 ASSORTED COLOUR BIRD ORN...	1495	
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	

페이지당 결과 수: 50 ▼ 1 ~ 30 (전체 30행)

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE FROM continual-nomad-479202-p8.test.data
WHERE Description IN ('Next Day Carriage','High Resolution Image');
```



- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE continual-nomad-479202-p8.test.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM continual-nomad-479202-p8.test.data
```



UnitPrice 살펴보기

- UnitPrice의 최솟값, 최댓값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price,
       MAX(UnitPrice) AS max_price,
```

```
AVG(UnitPrice) AS avg_price
FROM continual-nomad-479202-p8.test.data
```

행	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

- 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 continual-nomad-479202-p8.test.data
WHERE UnitPrice = 0
```

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

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
DELETE FROM continual-nomad-479202-p8.test.data
WHERE UnitPrice = 0
```

```
210 -- 단가가 0원인 데이터 제거
211 DELETE FROM continual-nomad-479202-p8.test.data
212 WHERE UnitPrice = 0
213 -- 혹은
214 CREATE OR REPLACE TABLE continual-nomad-479202-p8.test.data AS
215 SELECT *
216 FROM continual-nomad-479202-p8.test.data
217 WHERE UnitPrice > 0
218
219 -- RFM 스코어
220 -- Recency
221 -- InvoiceDate 컬럼을 연월일 자료형으로 변경
```

Syntax error: Expected end of input but got keyword SELECT at [7:1]

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 33개가 삭제되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM continual-nomad-479202-p8.test.data
```

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOF
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOF
3	2010-12-07	537626	22729	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELII
4	2010-12-07	537626	84997B	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	RED 3 PIECE RETROSPI
5	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	CLEAR DRAWER KNOB
6	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	PINK 3 PIECE POLKADC
7	2010-12-07	537626	22725	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELII
8	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLACK GRAND BAROQI
9	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB A

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT MAX(InvoiceDate) AS most_recent_date
FROM `continual-nomad-479202-p8.test.data`;
```

행	most_recent_date
1	2011-12-09 12:50:00 UTC

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT
  CustomerID,
  MAX(InvoiceDate) AS InvoiceDay
FROM continual-nomad-479202-p8.test.data
GROUP BY CCustomerID
ORDER BY CustomerID
```

행	CustomerID	InvoiceDay
1	12346	2011-01-18 10:17:00 UTC
2	12347	2011-12-07 15:52:00 UTC
3	12348	2011-09-25 13:13:00 UTC
4	12349	2011-11-21 09:51:00 UTC
5	12350	2011-02-02 16:01:00 UTC
6	12352	2011-11-03 14:37:00 UTC
7	12353	2011-05-19 17:47:00 UTC
8	12354	2011-04-21 13:11:00 UTC
9	12355	2011-05-09 13:49:00 UTC
10	12356	2011-11-17 08:40:00 UTC

- 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(InvoiceDate) AS InvoiceDay
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
);
```

행	CustomerID	recency
1	12377	315
2	12423	0
3	12426	194
4	12646	4
5	12950	2
6	13385	329
7	13575	23
8	13626	252
9	13654	43
10	13744	95

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

```
CREATE OR REPLACE TABLE continual-nomad-479202-p8.test.user_r AS
SELECT CustomerID,
EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
SELECT CustomerID, MAX(InvoiceDate) AS InvoiceDay
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
);
```

```
248 -- 지금까지의 결과를 user_r이라는 이름의 테이블로 저장
249 CREATE OR REPLACE TABLE continual-nomad-479202-p8.test.user_r AS
250 SELECT CustomerID,
251 EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
252 FROM (
253 SELECT CustomerID, MAX(InvoiceDate) AS InvoiceDay
254 FROM continual-nomad-479202-p8.test.data
255 GROUP BY CustomerID
256 );
257
258 -- Frequency
259 -- 전체 거래 건수 계산
...
```

쿼리 완료됨

주문항 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
CustomerID,
COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
ORDER BY purchase_cnt DESC
```

행	CustomerID	purchase_cnt
1	14911	242
2	12748	217
3	17841	169
4	14606	125
5	15311	118
6	13089	118
7	12971	88
8	13408	75
9	14646	73
10	16029	66

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID,
       SUM(Quantity) AS item_cnt
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
```

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

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

```
CREATE OR REPLACE TABLE `continual-nomad-479202-p8.modulabs.user_rf` AS
```

```
-- [WITH 절 시작]
```

```
WITH purchase_cnt AS (
```

```
-- (1) 전체 거래 건수 (F: Frequency)
```

```
SELECT
```

```
  CustomerID,
```

```
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
```

```
FROM `continual-nomad-479202-p8.test.data`
```

```
GROUP BY CustomerID
```

```
-- 여기 ORDER BY 제거! (불필요)
```

```
),
```

```
item_cnt AS (
```

```
-- (2) 총 수량 (M: Monetary 관련)
```

```
SELECT
```

```
  CustomerID,
```

```
  SUM(Quantity) AS item_cnt
```

```
FROM `continual-nomad-479202-p8.test.data`
```

```
GROUP BY CustomerID
```

```
)
```

```
-- [최종 SELECT] 3개 테이블(pc, ic, ur) 조인
```

```
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 `continual-nomad-479202-p8.test.user_r` AS ur
ON pc.CustomerID = ur.CustomerID;
```

쿼리 결과	
작업 정보	결과
실행 세부정보	실행 그래프
<div> <div></div> <div>이 문으로 이름이 user_rf인 테이블이 교체되었습니다.</div> </div>	

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT CustomerID, ROUND(SUM(UnitPrice * Quantity),1) AS user_total
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
```

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.6
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7

- 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 continual-nomad-479202-p8.test.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 continual-nomad-479202-p8.modulabs.user_rf AS rf
LEFT JOIN (
  --고객 별 총 지출액
  SELECT CustomerID, ROUND(SUM(UnitPrice * Quantity),1) AS user_total
FROM continual-nomad-479202-p8.test.data
GROUP BY CustomerID
) AS ut
ON rf.CustomerID = ut.CustomerID;
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
SELECT *
FROM continual-nomad-479202-p8.test.user_rfm
```

행	CustomerId	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.5	794.5
2	14569	1	79	1	227.4	227.4
3	13298	1	96	1	360.0	360.0
4	15520	1	314	1	343.5	343.5
5	13436	1	76	1	196.9	196.9
6	14204	1	72	2	150.6	150.6
7	15471	1	256	2	454.5	454.5
8	15195	1	1404	2	3861.0	3861.0
9	12650	1	250	3	242.4	242.4
10	15992	1	17	3	42.0	42.0
11	14578	1	240	3	168.6	168.6

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 continual-nomad-479202-p8.test.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM continual-nomad-479202-p8.test.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM continual-nomad-479202-p8.test.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



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 continual-nomad-479202-p8.test.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
      continual-nomad-479202-p8.test.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM continual-nomad-479202-p8.test.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

✓ 쿼리 완료됨

주문형 처리 할당량 사용 중

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 continual-nomad-479202-p8.test.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT (*) AS total_transactions,
    SUM(CASE WHEN InvoiceNO LIKE 'C%' THEN 1 ELSE 0 END) AS cancel_frequency
  FROM continual-nomad-479202-p8.test.data
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), ROUND(t.cancel_frequency / t.total_transactions,2) AS cancel_rate
FROM `continual-nomad-479202-p8.test.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON t.CustomerID = u.CustomerID;
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

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 *
FROM continual-nomad-479202-p8.test.user_data
```

행	Customerid	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	13667	1	149	149	299.0	299.0	82	0.0	83	0	0.0
2	15870	1	333	32	351.4	351.4	68	0.0	74	0	0.0
3	16054	1	380	145	783.9	783.9	70	0.0	70	0	0.0
4	16270	1	688	353	1141.2	1141.2	54	0.0	54	0	0.0
5	12349	1	630	18	1457.6	1457.6	72	0.0	72	0	0.0
6	15000	1	218	47	490.5	490.5	76	0.0	77	0	0.0
7	16504	1	510	25	465.9	465.9	76	0.0	83	0	0.0
8	12391	1	293	21	439.7	439.7	92	0.0	93	0	0.0
9	17914	1	457	3	329.4	329.4	72	0.0	75	0	0.0
10	15832	1	465	254	836.8	836.8	54	0.0	54	0	0.0
11	13339	1	326	200	860.1	860.1	54	0.0	54	0	0.0
12	16796	1	220	266	614.0	614.0	64	0.0	66	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : SQL에서도 정규표현식을 이용해 문자열 필터링이 가능하다는 것을 알게 되었다.

Problem : 특정 코드 값을 가진 데이터 수의 비율을 구하는 쿼리문에서 SELECT 절을 컬럼의 수만큼 적는게 너무 번거로웠다. 그래서 전체 컬럼(*)에 대해 조건을 걸고, 그룹화하여 HAVING으로 특정 코드를 조건으로 적어주니 훨씬 간편했다.

Try : UNION ALL을 하는 과정에서 최종 결과가 가로로 쭉 늘어져있는 형태여서 시각적으로 좋지 못했다. 그래서 서브쿼리를 통해 2개의 열로만 깔끔하게 나오는 결과를 얻을 수 있었다.

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT...	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