PDA

고객을 세그먼테이션하자 [프로젝트]

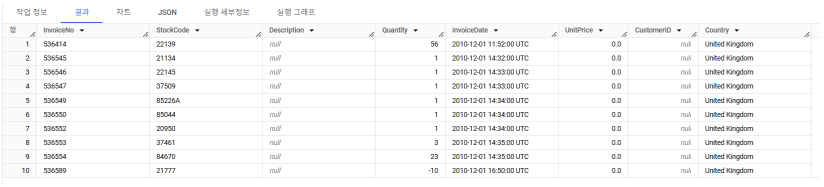
5-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]
SELECT * FROM `boreal-furnace-447401-f9.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(*) FROM `boreal-furnace-447401-f9.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]

작업 정보 결과

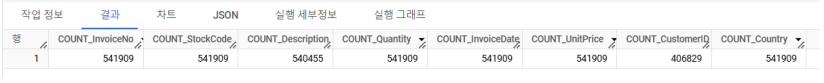
행	f0_
1	541909

데이터 수 세기

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

```
# [[YOUR QUERY]]
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
  `boreal-furnace-447401-f9.modulabs_project.data`;
```

[결과 이미지를 넣어주세요]



작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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

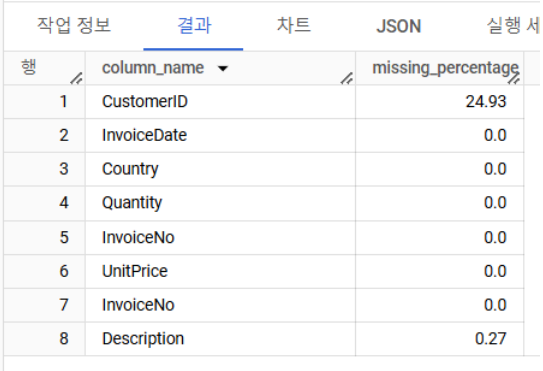
5-4. 데이터 전처리 방법(1): 결측치 제거

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(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
    AS missing_percentage
FROM  `boreal-furnace-447401-f9.modulabs_project.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  `boreal-furnace-447401-f9.modulabs_project.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  `boreal-furnace-447401-f9.modulabs_project.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  `boreal-furnace-447401-f9.modulabs_project.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  `boreal-furnace-447401-f9.modulabs_project.data` UNION ALL
SELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2)
    AS missing_percentage
FROM  `boreal-furnace-447401-f9.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  `boreal-furnace-447401-f9.modulabs_project.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  `boreal-furnace-447401-f9.modulabs_project.data`
;
```

[결과 이미지를 넣어주세요]



작업 정보 | 결과 | 차트 | JSON | 실행 서

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

결측치 처리 전략

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

```
select
  distinct(Description)
from
  modulabs_project.data
where
  stockcode ='85123A';
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트

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

결측치 처리

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

```
delete from
  modulabs_project.data
where
  Description is null
  or
  CustomerID is null;
```

[결과 이미지를 넣어주세요]

→ 일관성이 없고 결측치가

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 data의 행 135,080개가 삭제되었습니다.

5-5. 데이터 전처리(2): 중복값 처리

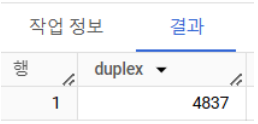
중복값 확인

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

```
SELECT *
FROM modulabs_project.data
# [[YOUR QUERY]]
select
    count(*) as duplex
from
  (
  select
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
```

```
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  from
    `boreal-furnace-447401-f9.modulabs_project.data`
  group by
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  having
    count(*) > 1
  )
```

[결과 이미지를 넣어주세요]

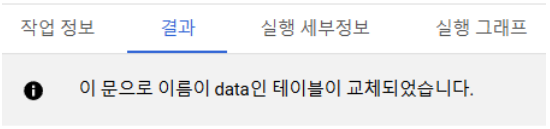


중복값 처리

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

```
# [[YOUR QUERY]];
create or replace table modulabs_project.data as
  select distinct
    InvoiceNo,
    StockCode,
    Description,
    Quantity,
    InvoiceDate,
    UnitPrice,
    CustomerID,
    Country
  from
    modulabs_project.data;
```

[결과 이미지를 넣어주세요]



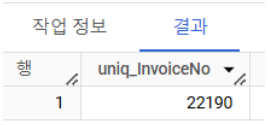
5-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

```
# [[YOUR QUERY]]
select
  count(*) as uniq_InvoiceNo
from(
  select
    InvoiceNo,
    count(distinct(InvoiceNo))
  from
    modulabs_project.data
  group by
    InvoiceNo
);
```

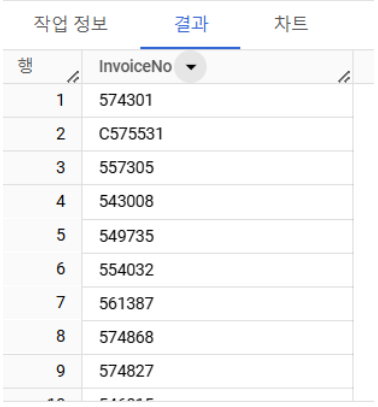
[결과 이미지를 넣어주세요]



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

```
# [[YOUR QUERY]]
  select
    InvoiceNo
  from
    modulabs_project.data
  group by
    InvoiceNo
  limit 100;
```

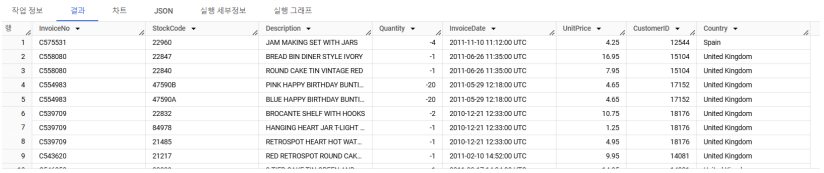
[결과 이미지를 넣어주세요]



- **`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]

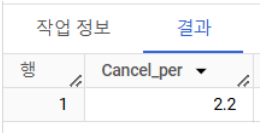


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
2	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
3	C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
4	C554983	47590B	PINK HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
6	C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
7	C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
8	C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
9	C543620	21217	RED RETROSPOT ROUND CAK...	-1	2011-02-10 14:52:00 UTC	9.95	14081	United Kingdom

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

```
SELECT
  round(
  (SUM(CASE WHEN InvoiceNo like 'C%' THEN 1 ELSE 0 END) / count(InvoiceNo)) * 100, 1
  )
   as Cancel_per
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



작업 정보 | 결과

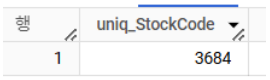
행	Cancel_per
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

```
# [[YOUR QUERY]]
select
  count(*) as uniq_StockCode
from(
  select
    StockCode,
    count(distinct(StockCode))
  from
    modulabs_project.data
  group by
    StockCode
);
```

[결과 이미지를 넣어주세요]



행	uniq_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 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 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]

행	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 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 modulabs_project.data
)
WHERE number_count in (0,1);
```

[결과 이미지를 넣어주세요]

행	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 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
# [[YOUR QUERY]]
SELECT
    round(
        (SUM
            (CASE
                WHEN
                    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))
                     IN (0, 1)
                THEN 1
                ELSE 0
             END) / count(StockCode)
        ) * 100, 2
    )
    as another_code
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	another_code
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', ''))
    AS number_count
      FROM modulabs_project.data)
      where number_count in (0,1)
);
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 실행 세부정보 | 실행 :

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 modulabs_project.data
group by Description
limit 30;
```

[결과 이미지를 넣어주세요]

행	Description	description_cnt
1	PAPER CHAIN KIT VINTAGE CHRISTMAS	718
2	CHRISTMAS CRAFT LITTLE FRIENDS	433
3	FELTCRAFT PRINCESS OLIVIA DOLL	254
4	TRADITIONAL KNITTING NANCY	523
5	ASSORTED COLOUR MINI CASES	372
6	ASSORTED COLOUR BIRD ORNAMENT	1405
7	6 RIBBONS RUSTIC CHARM	747
8	SET OF 6 RIBBONS VINTAGE CHRISTM...	343
9	SCANDINAVIAN REDS RIBBONS	343
10	PINK BLUE FELT CRAFT TRINKET BOX	465
11	EMBROIDERED RIBBON REEL ROSIE	49
12	EMBROIDERED RIBBON REEL EMILY	87

더보기

- **대소문자가 혼합된 Description이 있는지 확인하기**

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

[결과 이미지를 넣어주세요]

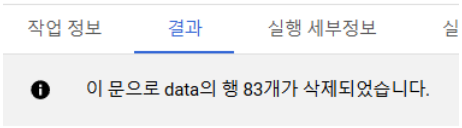
행	Description
1	BAG 125g SWIRLY MARBLES
2	3 TRADITIONAl BISCUIT CUTT...
3	BAG 250g SWIRLY MARBLES
4	ESSENTIAL BALM 3.5g TIN IN ...
5	FOLK ART GREETING CARD,pa...
6	BAG 500g SWIRLY MARBLES
7	POLYESTER FILLER PAD 45x45...
8	POLYESTER FILLER PAD 40x40...
9	Next Day Carriage
10	FRENCH BLUE METAL DOOR SI...
11	High Resolution Image

더보기

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM modulabs_project.data
WHERE REGEXP_CONTAINS(Description, r'[a-z]')
and Description in ('Next Day Carriage','High Resolution Image');
```

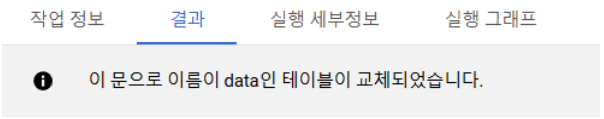
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  case
      when REGEXP_CONTAINS(Description, r'[a-z]')
      then upper(Description)
      else Description
  end AS Description
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

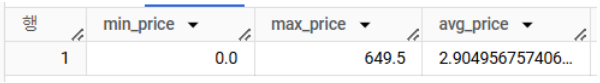


`UnitPrice` 살펴보기

- `UnitPrice`**의 최솟값, 최댓값, 평균을 구하기**

```
SELECT
   min(UnitPrice) AS min_price,
   max(UnitPrice) AS max_price,
   avg(UnitPrice) AS avg_price
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]



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(Quantity) 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;
```

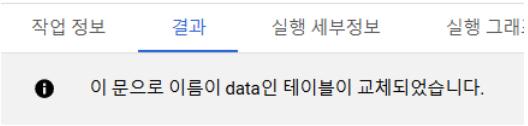
[결과 이미지를 넣어주세요]

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

- `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]



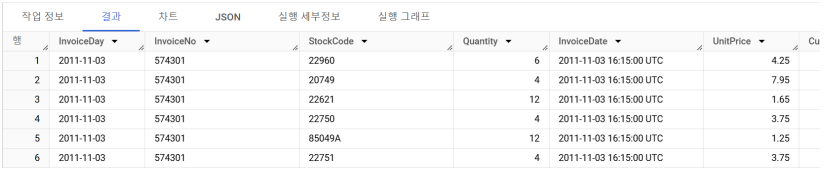
5-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

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

[결과 이미지를 넣어주세요]



행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	Cu
1	2011-11-03	574301	22960	6	2011-11-03 16:15:00 UTC	4.25	
2	2011-11-03	574301	20749	4	2011-11-03 16:15:00 UTC	7.95	
3	2011-11-03	574301	22621	12	2011-11-03 16:15:00 UTC	1.65	
4	2011-11-03	574301	22750	4	2011-11-03 16:15:00 UTC	3.75	
5	2011-11-03	574301	85049A	12	2011-11-03 16:15:00 UTC	1.25	
6	2011-11-03	574301	22751	4	2011-11-03 16:15:00 UTC	3.75	

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  max(InvoiceDate) AS most_recent_date,
  date(max(InvoiceDate)) AS InvoiceDay
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

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

- 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	12544	2011-11-10
2	13568	2011-06-19
3	13824	2011-11-07
4	14080	2011-11-07
5	14336	2011-11-23

- **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]

행	CustomerID	recency
1	16641	105
2	14083	4
3	13588	15
4	12821	214
5	17431	313

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

```
CREATE OR REPLACE TABLE modulabs_project.user_r AS
  with maxInvoiceDay as(
    SELECT
        CustomerID,
        MAX(DATE(InvoiceDate)) AS InvoiceDay
      FROM modulabs_project.data
      GROUP BY CustomerID
  ),
  cntInvoiceDay as(
    SELECT
        CustomerID,
    EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
    from maxInvoiceDay
  )
  select *
  from cntInvoiceDay;
```

[결과 이미지를 넣어주세요]

행	CustomerID	recency
1	18102	0
2	17389	0
3	13426	0
4	13777	0
5	13069	0
6	15694	0
7	17581	0
8	17428	0
9	15344	0
10	14422	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  count(distinct InvoiceNo) AS purchase_cnt
FROM modulabs_project.data
group by CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12544	2
2	13568	1
3	13824	5
4	14080	1
5	14336	4

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  sum(Quantity) AS item_cnt
FROM modulabs_project.data
group by CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	item_cnt
1	12544	130
2	13568	66
3	13824	768
4	14080	48
5	14336	1759

- **전체 거래 건수 계산와 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
  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 modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	18010	1	60	256
3	12792	1	215	256
4	15083	1	38	256
5	15520	1	314	1
6	13298	1	96	1
7	13436	1	76	1
8	14569	1	79	1
9	14476	1	110	257
10	13357	1	321	257
11	14204	1	72	2

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
 round(sum(UnitPrice * quantity),0) AS user_total
FROM modulabs_project.data
group by CustomerID;
```

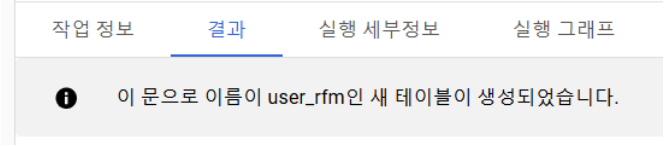
[결과 이미지를 넣어주세요]

행	CustomerID	user_total
1	12544	300.0
2	13568	187.0
3	13824	1699.0
4	14080	46.0
5	14336	1615.0
6	14592	558.0

- **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 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,0) AS user_average
FROM modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
  CustomerID,
  round(sum(UnitPrice * quantity),0) AS user_total
  FROM modulabs_project.data
  group by CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
select * FROM modulabs_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	18010	1	60	256	175.0	175.0
3	15083	1	38	256	88.0	88.0
4	12792	1	215	256	345.0	345.0
5	13436	1	76	1	197.0	197.0
6	14569	1	79	1	227.0	227.0

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

[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	16990	1	100	218	179.0	179.0	1
2	13302	1	5	155	64.0	64.0	1
3	15316	1	100	326	165.0	165.0	1
4	12791	1	96	373	178.0	178.0	1
5	14119	1	-2	354	-20.0	-20.0	1
6	18113	1	72	368	76.0	76.0	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 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
      modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM modulabs_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	14432	6	2013	9	2248.0	375.0	256	0.2
2	12428	11	3477	25	6366.0	579.0	256	0.87
3	13268	14	3525	17	3106.0	222.0	256	0.56
4	16995	1	-1	372	-1.0	-1.0	1	0.0
5	18233	1	4	325	440.0	440.0	1	0.0
6	14576	1	12	372	35.0	35.0	1	0.0

3. 구매 취소 경향성

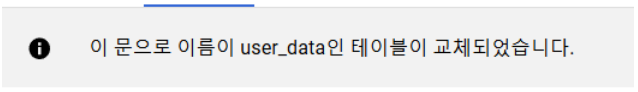
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(customerid) AS total_transactions,
    sum(case when InvoiceNo like 'c%' then 1 else 0 end) AS cancel_frequency
  FROM modulabs_project.data
  group by CustomerID
)

SELECT u.*, t.*
EXCEPT(CustomerID), round(cancel_frequency / total_transactions, 2)  AS cancel_rate
FROM `modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]



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

```
select * from modulabs_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	18113	1	72	368	76.0	76.0	1	0.0	1	0	0.0
3	16738	1	3	297	4.0	4.0	1	0.0	1	0	0.0
4	15668	1	72	217	76.0	76.0	1	0.0	1	0	0.0
5	17443	1	504	219	534.0	534.0	1	0.0	1	0	0.0
6	15657	1	24	22	30.0	30.0	1	0.0	1	0	0.0
7	15389	1	400	172	500.0	500.0	1	0.0	1	0	0.0

회고

두근두근 첫 메인 퀘스트! 정규 학습 시간 내로 끝내는 것으로 기획된 콘텐츠로 보였으나...

종일 매달려 있다가 익일 새벽에 겨우 제출이라니😅

좋았던 점

- **확실한 복습**

  → 공부한 내용을 '어떻게' 사용해야 하는지를 고민하는 시간이었다.

  그간 학습이 자율적으로 진행된 만큼 성취도를 파악하기엔 다소 어려웠는데, 덕분에 구체적인 사용법을 궁리할 수 있었다.

- **동기 부여**

  → 퀘스트라는 타이틀이 붙은 만큼 기한 내로 어떻게든 끝내고 제출하겠다는 의욕이 샘솟았다. 일찍 끝마친 다른 그루들을 보며 자극도 받고, 차후에는 보다 수월한 진행을 꿈꾸며 학습 플랜을 보다 구체적으로 세우게 되었다.

아쉬운 점

- **소화 불량**

  → 오늘치 노드에서도 여러 중요한 개념을 다루고 있었지만 문제 푸느라 바쁜데다 맑은 정신이 아니라 그런지 머리에 제대로 입력하지 못했다.

  다시 정독하지 않으면 조만간 낯설게 느낄 것만 같아 대책 마련이 필요하다.

- **시간 관리 실패**

  → 이렇게 오래 매여있을 난이도는 아니라고 생각하는데, 숲을 보는 대신 나무만 보다가 훌쩍 시간이 가버렸다...

개선할 점

- **본질 파악**

  → 지문에서 요구하는 바를 명확히 이해해야 헛발질 하느라 자원을 소모하지 않는다.

  소숫점 *첫째 자리까지* 반올림과 *첫째 자리에서* 반올림은 다르건만, 처음에 제대로 이해해놓고서는 자정 넘어 착각한 줄 알고 야단법석 떠느라 흘려보낸 시간이 어언 두 시간...

  이 외에도 지엽적인 부분에 매달리느라 아까운 시간과 체력을 허비했다.

- **완급 조절**

  → 전날 노드를 완료하고 싶은 욕심에 줄곧 매달려있다가 새벽에 취침했는데 이틀 연속으로 수면이 부족한 상태다 보니 몽롱 그 자체🥴

  하루이틀만 듣는 강의가 아닌 만큼, 주중에는 다음날 학습을 위해서라도 과욕 부리지 말고 적정 시점에 끊을 줄도 알아야 한다.