

고객을 세그멘테이션하자 [프로젝트]

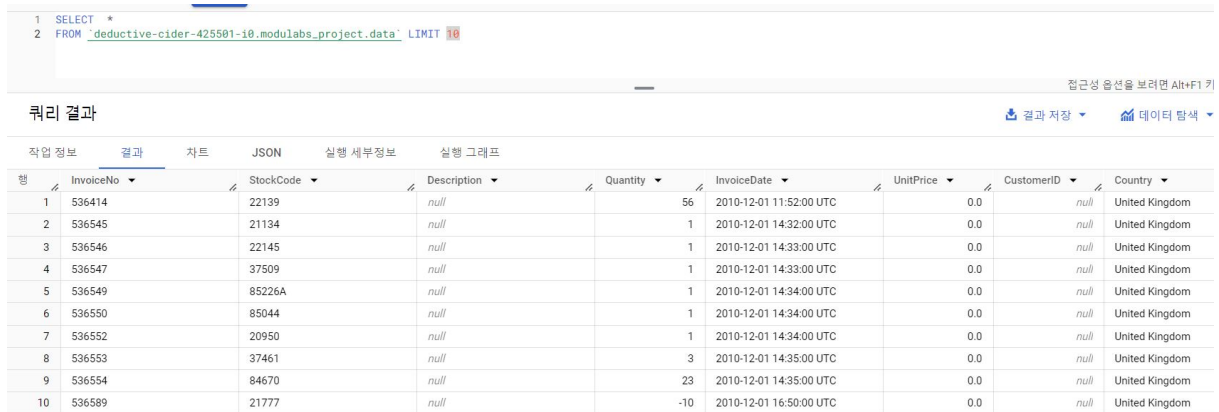
11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
# [[YOUR QUERY]]
SELECT *
FROM `deductive-cider-425501-i0.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 `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



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 `deductive-cider-425501-i0.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

1	SELECT
2	COUNT(InvoiceNo) AS COUNT_InvoiceNo,
3	COUNT(StockCode) AS COUNT_StockCode,
4	COUNT(Description) AS COUNT_Description,
5	COUNT(Quantity) AS COUNT_Quantity,
6	COUNT(InvoiceDate) AS COUNT_InvoiceDate,
7	COUNT(UnitPrice) AS COUNT_UnitPrice,
8	COUNT(CustomerID) AS COUNT_CustomerID,
9	COUNT(Country) AS COUNT_Country
10	FROM `deductive-cider-425501-i0.modulabs_project.data`

쿼리 결과									
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

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

```

# [[YOUR QUERY]]
SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM
  `deductive-cider-425501-i0.modulabs_project.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_percent
FROM
  `deductive-cider-425501-i0.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_perce
FROM
  `deductive-cider-425501-i0.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_percenta
FROM
  `deductive-cider-425501-i0.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_perce
FROM
`deductive-cider-425501-i0.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_percent
FROM
`deductive-cider-425501-i0.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_perce
FROM
`deductive-cider-425501-i0.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_percentag
FROM `deductive-cider-425501-i0.modulabs_project.data`

```

[결과 이미지를 넣어주세요]

```

1 SELECT
2   'InvoiceNo' AS column_name,
3   ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
4 FROM
5   `deductive-cider-425501-i0.modulabs_project.data`
6 UNION ALL
7 SELECT
8   'StockCode' AS column_name,
9   ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
10 FROM
11   `deductive-cider-425501-i0.modulabs_project.data`

```

쿼리 결과

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

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

결측치 처리 전략

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

```

SELECT
DISTINCT(Description)
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE
  StockCode = '85123A';

```

[결과 이미지를 넣어주세요]

```

1 SELECT
2   | DISTINCT(description)
3 FROM `deductive-cider-425501-i0.modulabs_project.data`
4
5 WHERE
6   | StockCode = '85123A'
7

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실
	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 `deductive-cider-425501-i0.modulabs_project.data`

WHERE
  Description IS NULL OR
  CustomerID IS NULL

```

[결과 이미지를 넣어주세요]

```

1 DELETE
2 FROM `deductive-cider-425501-i0.modulabs_project.data`
3
4 WHERE
5   | Description IS NULL OR
6   | CustomerID IS NULL

```

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

i 이 문으로 data의 행 135,080개가 삭제되었습니다.

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COUNT(*)
FROM `deductive-cider-425501-i0.modulabs_project.data`

GROUP BY
    InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
HAVING
    COUNT(*) > 1;

#50개만 보여서 답을 50으로 적었는데 하단에 4837행의 결과가 나타난 것을 알 수 있다.
```

[결과 이미지를 넣어주세요]

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
560937	22946	MINI JOSAW FURLEY	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom
560937	23154	SET OF 4 JAM JAR MAGNETS	1	2011-07-22 10:52:00 UTC	0.29	17920	United Kingdom
560937	21311	SET/4 BIRD MIRROR MAGNETS	1	2011-07-22 10:52:00 UTC	0.29	17920	United Kingdom
560937	22541	MINI JOSAW LEAP FROG	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom
560937	22539	MINI JOSAW DOLLY GIRL	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom

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
`deductive-cider-425501-i0.modulabs_project.data` AS
SELECT DISTINCT *
FROM

`deductive-cider-425501-i0.modulabs_project.data`;

SELECT
    COUNT(*) AS f0_
FROM
    (
        SELECT DISTINCT *
        FROM
            `deductive-cider-425501-i0.modulabs_project.data`
    );
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE
`deductive-cider-425501-i0.modulabs_project.data` AS
SELECT DISTINCT *
FROM

`deductive-cider-425501-i0.modulabs_project.data`;

SELECT
| COUNT(*) AS f0_
FROM
(
| SELECT DISTINCT *
| FROM
| `deductive-cider-425501-i0.modulabs_project.data`
);
```

쿼리 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
f0_ ▾	401604				

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT
COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM
`deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

```
SELECT
  COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
```

의 결과

정보	결과	차트	JSON	실행 세부정보
	unique_invoice_coun			
	22190			

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

```
# [[YOUR QUERY]]

SELECT
  DISTINCT InvoiceNo
FROM
  `deductive-cider-425501-i0.modulabs_project.data`

LIMIT 100;
```

[결과 이미지를 넣어주세요]

```
SELECT
  DISTINCT InvoiceNo
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
LIMIT 100;
```

결과

정보 결과 차트 JSON 실행 세부정보

InvoiceNo
574301
C575531
557305
543008
549735
554032
561387
574868
574827
546015

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

```
SELECT *
FROM project_name.modulabs_project.data
WHERE # [[YOUR QUERY]]
LIMIT 100;
```

[결과 이미지를 넣어주세요]

```
SELECT *
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
WHERE
  InvoiceNo LIKE 'C%'
LIMIT 100;
```

최근성 옵션을 보려면 Alt+F1 키를

결과

결과 저장 데이터 탐색

정보	결과	차트	JSON	실행 세부정보	실행 그래프		
InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
C575531	22960	JAM MAKING SET WITH JARS	-4	2011-11-10 11:12:00 UTC	4.25	12544	Spain
C558080	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
C554983	47590A	BLUE HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
C554983	47590B	PINK HAPPY BIRTHDAY BUNTL...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
C539709	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	18176	United Kingdom
C539709	84978	HANGING HEART JAR T-LIGHT ...	-1	2010-12-21 12:33:00 UTC	1.25	18176	United Kingdom
C539709	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	United Kingdom
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(*) * 100, 1
) AS canceled_percentage
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]


```
SELECT ROUND(
  SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) * 100, 1
) AS canceled_percentage
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
```

의 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
	canceled_percentage				
	2.2				

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
# [[YOUR QUERY]]
SELECT
  COUNT(DISTINCT StockCode)
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

```
SELECT
  COUNT(DISTINCT StockCode)
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
```

의 결과

정보	결과	차트	JSON	실행 세부정보	슬
f0_					
	3684				

- 어떤 제품이 가장 많이 판매되었는지 보기 위하여 StockCode 별 등장 빈도를 출력하기
 - 상위 10개의 제품들을 출력하기

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY
  StockCode
ORDER BY
```

```
sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

```
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

결과

정보 결과 차트 JSON 실행 세부정보 실행 그래프

StockCode ▼	sell_cnt ▼
85123A	2065
22423	1894
85099B	1659
47566	1409
84879	1405
20725	1346
22720	1224
POST	1196
22197	1110
22200	1100

비고

- **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 `deductive-cider-425501-i0.modulabs_project.data`
)
WHERE number_count BETWEEN 0 AND 1;
```

[결과 이미지를 넣어주세요]

```

13
14 SELECT DISTINCT StockCode, number_count
15 FROM (
16     SELECT StockCode,
17         LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
18     FROM `deductive-cider-425501-i0.modulabs_project.data`
19 )
20 WHERE number_count BETWEEN 0 AND 1;
21
22
23
24
25

```

← 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
#	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			

- **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
  ROUND(
    ((SELECT COUNT(*) FROM `deductive-cider-425501-i0.modulabs_project.data`
      WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) BETWEEN 0 AND 1)
    / (SELECT COUNT(*) FROM `deductive-cider-425501-i0.modulabs_project.data`)) * 100, 2) AS perce

```

[결과 이미지를 넣어주세요]

```

SELECT
  ROUND(
    ((SELECT COUNT(*) FROM `deductive-cider-425501-i0.modulabs_project.data`
      WHERE LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) BETWEEN 0 AND 1)
    / (SELECT COUNT(*) FROM `deductive-cider-425501-i0.modulabs_project.data`)) * 100, 2) AS percentage;

```

결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
percentage ▼					
0.48					

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM `deductive-cider-425501-i0.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 `deductive-cider-425501-i0.modulabs_project.data`
  )
  WHERE number_count BETWEEN 0 AND 1
);
```

[결과 이미지를 넣어주세요]

```
1 DELETE FROM `deductive-cider-425501-i0.modulabs_project.data`
2 WHERE StockCode IN (
3   SELECT DISTINCT StockCode
4   FROM (
5     SELECT StockCode,
6       LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
7     FROM `deductive-cider-425501-i0.modulabs_project.data`
8   )
9   WHERE number_count BETWEEN 0 AND 1
10 );
11
```

쿼리 결과

작업 정보 결과 실행 세부정보 실행 그래프

i 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 `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]

```
SELECT Description, COUNT(*) AS description_cnt
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
	Description ▾		description_cnt ▾		
	WHITE HANGING HEART T-LIG...		2058		
	REGENCY CAKESTAND 3 TIER		1894		
	JUMBO BAG RED RETROSPOT		1659		
	PARTY BUNTING		1409		
	ASSORTED COLOUR BIRD ORN...		1405		

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE
Description IN ('Next Day Carriage', 'High Resolution Image')
```

[결과 이미지를 넣어주세요]

```
DELETE
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE
Description IN ('Next Day Carriage', 'High Resolution Image')
```

결과

정보 결과 실행 세부정보 실행 그래프

이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.data` AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM deductive-cider-425501-i0.modulabs_project.data
```

의 결과

정보 **결과** 실행 세부정보 실행 그래프

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 `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]

```

1 SELECT
2   MIN(UnitPrice) AS min_price,
3   MAX(UnitPrice) AS max_price,
4   AVG(UnitPrice) AS avg_price
5 FROM `deductive-cider-425501-i0.modulabs_project.data`

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
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 `deductive-cider-425501-i0.modulabs_project.data`
WHERE UnitPrice = 0;

```

[결과 이미지를 넣어주세요]

```

SELECT
  COUNT(*) AS cnt_quantity,
  MIN(Quantity) AS min_quantity,
  MAX(Quantity) AS max_quantity,
  AVG(Quantity) AS avg_quantity
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE UnitPrice = 0;

```

리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
cnt_quantity	min_quantity	max_quantity	avg_quantity		
33	1	12540	420.5151515151...		

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```

CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.data` AS
SELECT *

```

```
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE UnitPrice != 0;
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.data` AS
SELECT *
FROM `deductive-cider-425501-i0.modulabs_project.data`
WHERE UnitPrice != 0;
```

결과

정보 결과 실행 세부정보 실행 그래프

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT
    DATE(InvoiceDate) AS InvoiceDay,
    *
FROM
    `deductive-cider-425501-i0.modulabs_project.dta`
```

[결과 이미지를 넣어주세요]

```
SELECT
    DATE(InvoiceDate) AS InvoiceDay,
    *
FROM
    `deductive-cider-425501-i0.modulabs_project.data`
```

결과

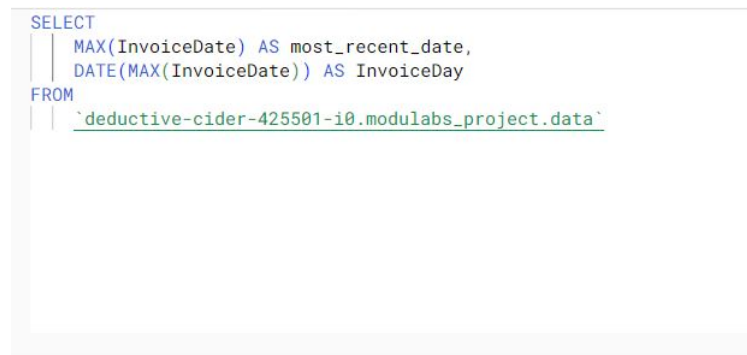
정보 결과 차트 JSON 실행 세부정보 실행 그래프

InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate
2011-11-03	574301	22750	4	2011-11-03 16:15:00 UTC
2011-11-03	574301	22734	6	2011-11-03 16:15:00 UTC
2011-11-03	574301	22144	6	2011-11-03 16:15:00 UTC
2011-11-03	574301	85049A	12	2011-11-03 16:15:00 UTC
2011-11-03	574301	20971	12	2011-11-03 16:15:00 UTC

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
    MAX(InvoiceDate) AS most_recent_date,
    DATE(MAX(InvoiceDate)) AS InvoiceDay
FROM
    `deductive-cider-425501-i0.modulabs_project.data`
```

[결과 이미지를 넣어주세요]



리 결과

검 정보	결과	차트	JSON	실행 세부정보	실행 그라
	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
    `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY
    CustomerID;
```

[결과 이미지를 넣어주세요]

```

SELECT
  CustomerID,
  MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY
  CustomerID;

```

결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
	CustomerID ▼	InvoiceDay ▼			
	12544	2011-11-10			
	13568	2011-06-19			
	13824	2011-11-07			
	14080	2011-11-07			
	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 `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY CustomerID
)

```

[결과 이미지를 넣어주세요]

```

SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY CustomerID
)

```

| 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
	CustomerID ▼	recency ▼			
	17155	17			
	16133	3			
	17928	45			
	16907	29			
	17676	4			

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

```

CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.user_r` AS
WITH user_last_purchase AS (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM
    `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY
    CustomerID
),
most_recent_date AS (
  SELECT
    MAX(DATE(InvoiceDate)) AS most_recent_date
  FROM
    `deductive-cider-425501-i0.modulabs_project.data`
)
SELECT
  ulp.CustomerID,
  EXTRACT(DAY FROM mrd.most_recent_date - ulp.InvoiceDay) AS recency
FROM
  user_last_purchase ulp,
  most_recent_date mrd;

```

[결과 이미지를 넣어주세요]

```

15 | FROM
16 |   `deductive-cider-425501-i0.modulabs_project.data`
17 | )
18 |
19 | SELECT
20 |   ulp.CustomerID,
21 |   EXTRACT(DAY FROM mrd.most_recent_date - ulp.InvoiceDay) AS recency
22 | FROM
23 |   user_last_purchase ulp,
24 |   most_recent_date mrd;
25 |
26 |
27 | SELECT *
28 | FROM `deductive-cider-425501-i0.modulabs_project.user_r`

```

← 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	CustomerID ▾	recency ▾			
1	16446	0			
2	17001	0			
3	12662	0			
4	15910	0			
5	12985	0			
6	12112	0			

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```

SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY
  CustomerID;

```

[결과 이미지를 넣어주세요]

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM
  `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY
  CustomerID;
```

결과

정보

결과

차트

JSON

실행 세부정보

실행 그래프

CustomerID ▾	purchase_cnt ▾	
12544	2	
13568	1	
13824	5	
14080	1	
14336	4	

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY CustomerID;
```

결과

정보

결과

차트

JSON

실행 세부정보

실행 그래프

<div>CustomerID</div>	<div>item_cnt</div>	
12544	130	
13568	66	
13824	768	
14080	48	
14336	1759	

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

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.user_rf` AS
-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM
    `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY
    CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM
    `deductive-cider-425501-i0.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
  deductive-cider-425501-i0.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

```
26 ic.item_cnt,
27 ur.recency
28 FROM
29 purchase_cnt AS pc
30 JOIN
31 item_cnt AS ic
32 ON pc.CustomerID = ic.CustomerID
33 JOIN
34 deductive-cider-425501-i0.modulabs_project.user_r AS ur
35 ON pc.CustomerID = ur.CustomerID;
36
37
38 SELECT *
39 FROM `deductive-cider-425501-i0.modulabs_project.user_rf`
```

← 쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
병	CustomerID	purchase_cnt	item_cnt	recency	
1	12713	1	505	0	
2	12792	1	215	256	
3	15083	1	38	256	
4	18010	1	60	256	
5	15520	1	314	1	

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

```
SELECT
  CustomerID,
  ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
FROM `deductive-cider-425501-i0.modulabs_project.data`
GROUP BY CustomerID;
```

리 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프
	CustomerID ▼	user_total ▼			
	12544	299.7			
	13568	187.0			
	13824	1698.9			
	14080	45.6			
	14336	1614.9			

- 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 `deductive-cider-425501-i0.modulabs_project.user_rfm` AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  CASE
    WHEN rf.purchase_cnt > 0 THEN (ut.user_total / rf.purchase_cnt)
    ELSE 0
  END AS user_average

FROM
  deductive-cider-425501-i0.modulabs_project.user_rf rf
LEFT JOIN (
  SELECT
    CustomerID,
    ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM
    `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY CustomerID
```

```
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

```
10 | ELSE 0
11 | END AS user_average
12 |
13 | FROM
14 | deductive-cider-425501-i0.modulabs_project.user_rf rf
15 | LEFT JOIN (
16 | SELECT
17 | CustomerID,
18 | ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
19 | FROM
20 | `deductive-cider-425501-i0.modulabs_project.data`
21 | GROUP BY CustomerID
22 | ) ut
23 | ON rf.CustomerID = ut.CustomerID;
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

❗ 이 문으로 이름이 user_rfm인 테이블이 교체되었습니다.

FLYING KITE

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```
# [[YOUR QUERY]];

SELECT *
FROM `deductive-cider-425501-i0.modulabs_project.user_rfm`
```

[결과 이미지를 넣어주세요]

```
1
2 SELECT *
3 FROM `deductive-cider-425501-i0.modulabs_project.user_rfm`
```

쿼리 결과

작업 정보 **결과** 차트 JSON 실행 세부정보 실행 그래프

	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	
1	12713	1	505	0	794.6	794.6	
2	18010	1	60	256	174.8	174.8	
3	15083	1	38	256	88.2	88.2	
4	12792	1	215	256	344.5	344.5	
5	13436	1	76	1	196.9	196.9	
6	13298	1	96	1	360.0	360.0	
7	15520	1	314	1	343.5	343.5	
8	14569	1	79	1	227.4	227.4	
9	13357	1	321	257	609.4	609.4	
10	14476	1	110	257	193.0	193.0	
11	14204	1	72	2	150.6	150.6	
12	15471	1	256	2	454.5	454.5	

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 `deductive-cider-425501-i0.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM deductive-cider-425501-i0.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM deductive-cider-425501-i0.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM deductive-cider-425501-i0.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM deductive-cider-425501-i0.modulabs_project.user_rfm AS ur
```

쿼리 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프			
CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products		
1/34	1	216	86	229.0	229.0	1		
14576	1	12	372	35.4	35.4	1		
18184	1	60	15	49.8	49.8	1		
13703	1	10	318	99.5	99.5	1		
15562	1	39	351	134.6	134.6	1		
13270	1	200	366	590.0	590.0	1		
16995	1	-1	372	-1.3	-1.3	1		
15940	1	4	311	35.8	35.8	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 `deductive-cider-425501-i0.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
      `deductive-cider-425501-i0.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
)
```

```

GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM deductive-cider-425501-i0.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

```

[결과 이미지를 넣어주세요]

```

FROM
  `deductive-cider-425501-i0.modulabs_project.data`
WHERE CustomerID IS NOT NULL
)
GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM deductive-cider-425501-i0.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;

SELECT *
FROM `deductive-cider-425501-i0.modulabs_project.user_data`

```

쿼리 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프			
CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	
14432	6	2013	9	2248.5	374.75	256	0.2	
12428	11	3477	25	6366.0	578.7272727272...	256	0.87	
13268	14	3525	17	3105.7	221.8357142857...	256	0.56	
17948	1	144	147	358.6	358.6	1	0.0	
13391	1	4	203	59.8	59.8	1	0.0	

페이지당 결과 수: 50

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 `deductive-cider-425501-i0.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(*) AS total_transactions,
    COUNTIF(Quantity < 0) AS cancel_frequency
  FROM `deductive-cider-425501-i0.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT
  u.*,
  t.total_transactions,
  t.cancel_frequency,
  ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM `deductive-cider-425501-i0.modulabs_project.data` AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;

```

[결과 이미지를 넣어주세요]

```
SELECT *
FROM `deductive-cider-425501-i0.modulabs_project.user_data`
```

접근성 옵션을 보려면

리 결과

결과 저장 데이터

정보	결과	차트	JSON	실행 세부정보	실행 그래프			
<div><div></div><div>Quantity</div><div></div></div>	<div><div></div><div>InvoiceDate</div><div></div></div>	<div><div></div><div>UnitPrice</div><div></div></div>	<div><div></div><div>CustomerID</div><div></div></div>	<div><div></div><div>Country</div><div></div></div>	<div><div></div><div>Description</div><div></div></div>	<div><div></div><div>total_transactions</div><div></div></div>	<div><div></div><div>cancel_frequency</div><div></div></div>	ca
4	2011-11-03 16:15:00 UTC	3.75	12544	Spain	FELTCRAFT PRINCESS LOLA D...	19	1	
6	2011-11-03 16:15:00 UTC	2.95	12544	Spain	PAPER CHAIN KIT 50'S CHRIS...	19	1	
4	2011-11-03 16:15:00 UTC	7.95	12544	Spain	ASSORTED COLOUR MINI CAS...	19	1	
6	2011-11-03 16:15:00 UTC	4.25	12544	Spain	JAM MAKING SET WITH JARS	19	1	
12	2011-11-03 16:15:00 UTC	1.25	12544	Spain	PINK RIUF FIT CRAFT TRINK...	19	1	

페이지당 결과 수: 50 (현재 399573행)

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

```
# [[YOUR QUERY]];
SELECT *
FROM `deductive-cider-425501-i0.modulabs_project.user_data`
```

[결과 이미지를 넣어주세요]

```
CREATE OR REPLACE TABLE `deductive-cider-425501-i0.modulabs_project.user_data` AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM deductive-cider-425501-i0.modulabs_project.data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM deductive-cider-425501-i0.modulabs_project.user_data AS ur,
unique_products AS up
```

쿼리 결과

정보	결과	차트	JSON	실행 세부정보	실행 그래프		
CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	
1/34/	1	216	86	229.0	229.0	1	
14576	1	12	372	35.4	35.4	1	
18184	1	60	15	49.8	49.8	1	
13703	1	10	318	99.5	99.5	1	
15562	1	39	351	134.6	134.6	1	
13270	1	200	366	590.0	590.0	1	
16995	1	-1	372	-1.3	-1.3	1	
15940	1	4	311	35.8	35.8	1	