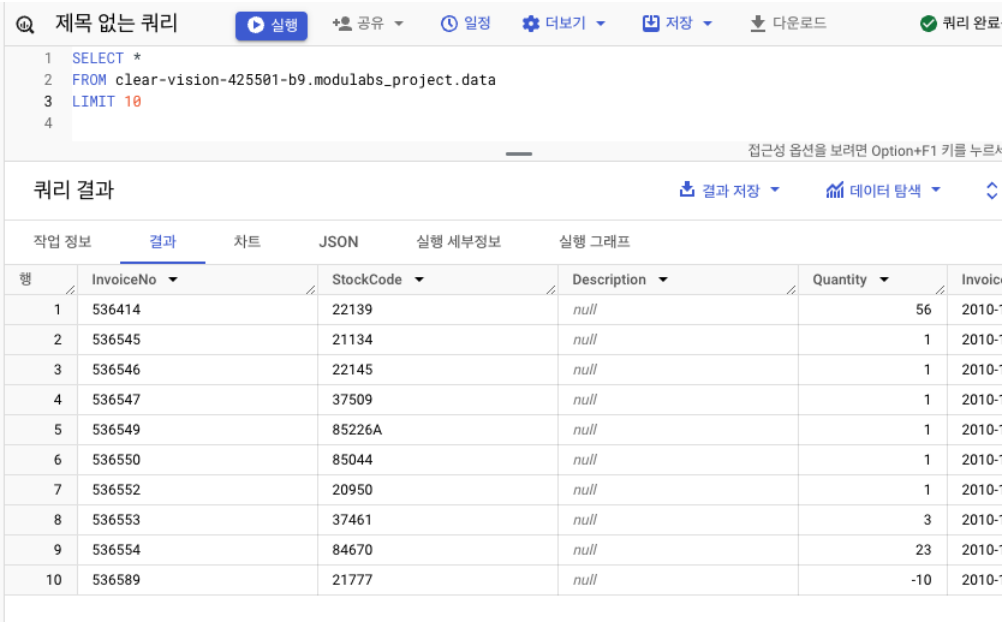
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

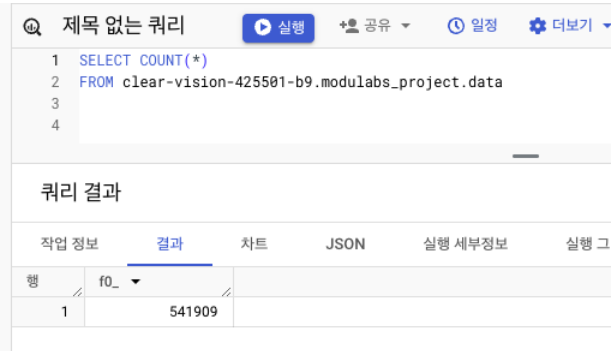
```
SELECT *
FROM clear-vision-425501-b9.modulabs_project.data
LIMIT 10;
```



행	InvoiceNo	StockCode	Description	Quantity	Invoic
1	536414	22139	null	56	2010-
2	536545	21134	null	1	2010-
3	536546	22145	null	1	2010-
4	536547	37509	null	1	2010-
5	536549	85226A	null	1	2010-
6	536550	85044	null	1	2010-
7	536552	20950	null	1	2010-
8	536553	37461	null	3	2010-
9	536554	84670	null	23	2010-
10	536589	21777	null	-10	2010-

- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
SELECT COUNT(*)
FROM clear-vision-425501-b9.modulabs_project.data;
```

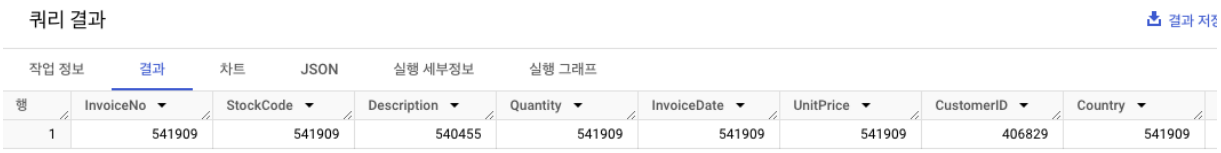


행	f0_
1	541909

데이터 수 세기

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

```
SELECT COUNT(InvoiceNo) AS InvoiceNo
  , COUNT(StockCode) AS StockCode
  , COUNT(Description) AS Description
  , COUNT(Quantity) AS Quantity
  , COUNT(InvoiceDate) AS InvoiceDate
  , COUNT(UnitPrice) AS UnitPrice
  , COUNT(CustomerID) AS CustomerID
  , COUNT(Country) AS Country
FROM clear-vision-425501-b9.modulabs_project.data;
```



행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	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**

```
WITH invoice AS (
  SELECT
    'InvoiceNo' AS invoiceCode,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
, stock AS (
  SELECT
    'StockCode' AS stockCode,
    ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
, descri AS (
  SELECT
    'Description' AS description,
    ROUND(SUM(CASE WHEN Description IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
  FROM clear-vision-425501-b9.modulabs_project.data
)
, quanti AS (
  SELECT
    'Quantity' AS quantity,
    ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
, invoiceDat AS (
  SELECT
    'InvoiceDate' AS invoiceDate,
    ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentag
```

```
  FROM clear-vision-425501-b9.modulabs_project.data
)
, unitP AS (
  SELECT
    'UnitPrice' AS unitPrice,
    ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
, custom AS (
  SELECT
    'CustomerID' AS customID,
    ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
, coun AS (
  SELECT
    'Country' AS country,
    ROUND(SUM(CASE WHEN Country IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
  FROM clear-vision-425501-b9.modulabs_project.data
)
SELECT invoiceCode, missing_percentage
FROM invoice
UNION ALL
SELECT stockCode, missing_percentage
FROM stock
UNION ALL
SELECT description, missing_percentage
FROM descri
UNION ALL
SELECT quantity, missing_percentage
FROM quanti
UNION ALL
SELECT invoiceDate, missing_percentage
FROM invoiceDat
UNION ALL
SELECT unitPrice, missing_percentage
FROM unitP
UNION ALL
SELECT customID, missing_percentage
FROM custom
UNION ALL
SELECT country, missing_percentage
FROM coun;
```

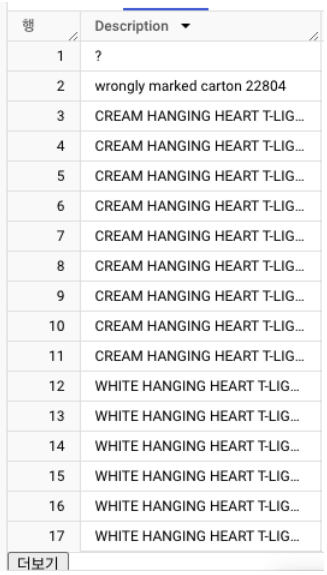
행	invoiceCode	missing_percentage
1	Custome	24.93
2	UnitPrice	0.0
3	Country	0.0
4	InvoiceDate	0.0
5	Description	0.27
6	Quantity	0.0
7	InvoiceNo	0.0
8	StockCode	0.0

invoiceCode

결측치 처리 전략

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

```
SELECT Description
FROM clear-vision-425501-b9.modulabs_project.data
WHERE StockCode = '85123A';
```



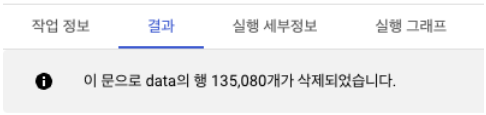
행	Description
1	?
2	wrongly marked carton 22804
3	CREAM HANGING HEART T-LIG...
4	CREAM HANGING HEART T-LIG...
5	CREAM HANGING HEART T-LIG...
6	CREAM HANGING HEART T-LIG...
7	CREAM HANGING HEART T-LIG...
8	CREAM HANGING HEART T-LIG...
9	CREAM HANGING HEART T-LIG...
10	CREAM HANGING HEART T-LIG...
11	CREAM HANGING HEART T-LIG...
12	WHITE HANGING HEART T-LIG...
13	WHITE HANGING HEART T-LIG...
14	WHITE HANGING HEART T-LIG...
15	WHITE HANGING HEART T-LIG...
16	WHITE HANGING HEART T-LIG...
17	WHITE HANGING HEART T-LIG...

더보기

결측치 처리

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

```
DELETE FROM clear-vision-425501-b9.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL;
```



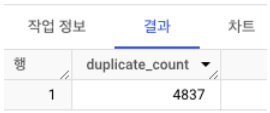
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT COUNT(*) AS duplicate_count
FROM (
    SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, COU
    FROM clear-vision-425501-b9.modulabs_project.data
    GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country
```

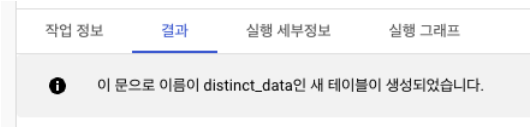
```
    HAVING COUNT(*) > 1
);
```



중복값 처리

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

```
CREATE OR REPLACE TABLE clear-vision-425501-b9.modulabs_project.distinct_data AS
SELECT DISTINCT *
FROM clear-vision-425501-b9.modulabs_project.data;
```

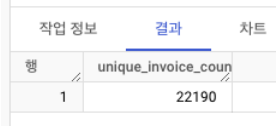


11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

```
SELECT COUNT(DISTINCT InvoiceNo) AS unique_invoice_count
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```



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

```
SELECT DISTINCT InvoiceNo
FROM clear-vision-425501-b9.modulabs_project.distinct_data
LIMIT 100;
```

작업 정보 | 결과 | 차트 | JSO

행	InvoiceNo
1	574301
2	C575531
3	557305
4	543008
5	549735
6	554032
7	561387
8	574868
9	574827
10	546015
11	551859
12	554665

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

```
SELECT *
FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

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

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	22840	ROUND CAKE TIN VINTAGE RED	-1	2011-06-26 11:35:00 UTC	7.95	15104	United Kingdom
3	C558080	22847	BREAD BIN DINER STYLE IVORY	-1	2011-06-26 11:35:00 UTC	16.95	15104	United Kingdom
4	C554983	47590A	BLUE HAPPY BIRTHDAY BUNTI...	-20	2011-05-29 12:18:00 UTC	4.65	17152	United Kingdom
5	C554983	47590B	PINK 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 Quantity < 0 THEN 1 ELSE 0 END) * 100.0 / (SELECT COUNT(*) FROM clear-vision-
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```

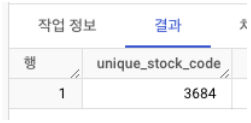
작업 정보 | 결과 | 차트

행	percentage
1	2.2

`StockCode` 살펴보기

- 고유한 `StockCode`의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stock_code
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```



행	unique_stock_code
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 clear-vision-425501-b9.modulabs_project.distinct_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

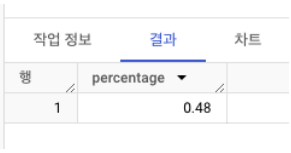
```
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM clear-vision-425501-b9.modulabs_project.distinct_data
)
WHERE number_count BETWEEN 0 AND 1;
```

작업 정보 결과 차트 JSON 실행 세부정보

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

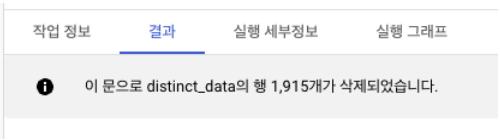
- `StockCode`**의 컬럼에 있던 값 중에서 숫자를 제외한 문자만 남기고 문자가 몇 자리 수 인지 세고**
  - **숫자가 0~1개인 값들을 가지고 있는 데이터 수는 전체 데이터 수 대비 몇 퍼센트**인지 구하기 (소수점 두 번째 자리까지)

```
SELECT
ROUND(
(SELECT COUNT()
FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')
) * 100 /
(SELECT COUNT()
FROM clear-vision-425501-b9.modulabs_project.distinct_data),
2) AS percentage;
```



- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM clear-vision-425501-b9.modulabs_project.data
  WHERE StockCode IN ('POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT', 'CRUK')
);
```



`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

행	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	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
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990
15	NATURAL SLATE HEART CHAL...	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POLKADOT	961
18	LUNCH BAG SUKI DESIGN	932

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE Description IN ('Next Day Carriage', 'High Resolution Image');
```

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 distinct_data의 행 83개가 삭제되었습니다.

- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE clear-vision-425501-b9.modulabs_project.distinct_data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```

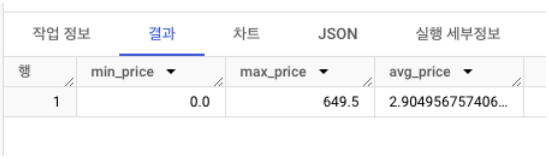
작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 distinct_data인 테이블이 교체되었습니다.

`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 clear-vision-425501-b9.modulabs_project.distinct_data;
```

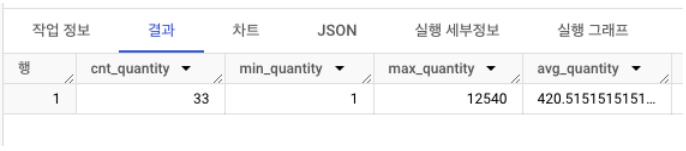


작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

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, AV
FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE UnitPrice = 0;
```



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

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 clear-vision-425501-b9.modulabs_project.distinct_data AS
SELECT *
FROM clear-vision-425501-b9.modulabs_project.distinct_data
WHERE UnitPrice = 0;
```

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 distinct_data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```

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

행	InvoiceDay	InvoiceNo	StockCode	Quantity
1	2011-11-03	574301	85049E	12
2	2011-11-03	574301	22621	12
3	2011-11-03	574301	22144	6
4	2011-11-03	574301	20749	4
5	2011-11-03	574301	22910	6
6	2011-11-03	574301	22086	6
7	2011-11-03	574301	23512	6
8	2011-11-03	574301	22751	4
9	2011-11-03	574301	20971	12

- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
   MAX(DATE(InvoiceDate)) AS most_recent_date,
FROM clear-vision-425501-b9.modulabs_project.distinct_data;
```

작업 정보 | 결과 | 차트

행	most_recent_date
1	2011-12-09

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

```
SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID;
```

작업 정보 | 결과 | 차트 | JSON

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
6	14592	2011-11-04
7	15104	2011-06-26
8	15360	2011-10-31
9	15872	2011-11-25
10	16128	2011-11-22

- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
SELECT
  CustomerID,
```

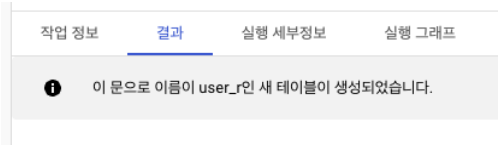
```
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM project_name.modulabs_project.data
  GROUP BY CustomerID
);
```

작업 정보 | 결과 | 차트 | JSON

행	CustomerID	recency
1	17408	163
2	16385	60
3	16131	51
4	13831	16
5	17941	130
6	15638	301
7	13606	29
8	17968	373
9	14652	77
10	14908	74

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

```
CREATE OR REPLACE TABLE clear-vision-425501-b9.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 clear-vision-425501-b9.modulabs_project.distinct_data
  GROUP BY CustomerID
);
```



작업 정보 | 결과 | 차트 | JSON

행	CustomerID	recency
1	12518	0
2	13426	0
3	12423	0
4	14422	0
5	17315	0
6	14446	0
7	16446	0
8	17389	0
9	15910	0
10	12748	0
11	16705	0
12	14397	0

Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

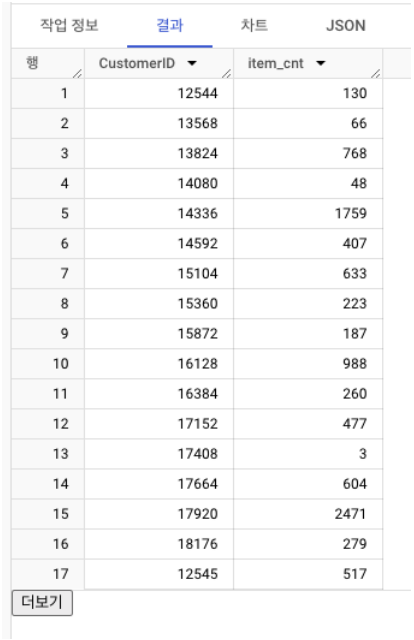
```
SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID;
```

작업 정보 | 결과 | 차트 | JSON

행	CustomerID	purchase_cnt
1	12544	19
2	13568	43
3	13824	46
4	14080	4
5	14336	90
6	14592	158
7	15104	69
8	15360	13
9	15872	108

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID;
```



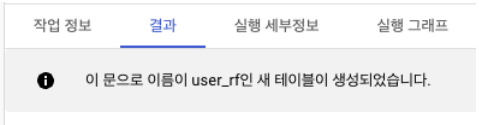
작업 정보 | 결과 | 차트 | JSON

행	CustomerID	item_cnt
1	12544	130
2	13568	66
3	13824	768
4	14080	48
5	14336	1759
6	14592	407
7	15104	633
8	15360	223
9	15872	187
10	16128	988
11	16384	260
12	17152	477
13	17408	3
14	17664	604
15	17920	2471
16	18176	279
17	12545	517

더보기

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

```
CREATE OR REPLACE TABLE clear-vision-425501-b9.modulabs_project.user_rf AS
WITH purchase_cnt AS (
 SELECT
  CustomerID,
  COUNT(InvoiceNo) AS purchase_cnt
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID
),
item_cnt AS (
  SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID
)
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 clear-vision-425501-b9.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;
```



작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

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

행	CustomerID	purchase_cnt	item_cnt	recency
1	17428	343	9435	0
2	17754	90	1767	0
3	12433	420	11071	0
4	12748	4440	23516	0
5	12526	68	624	0
6	14446	276	856	0
7	14441	59	430	0
8	12423	118	1312	0
9	13777	217	12807	0
10	13069	469	5454	0
11	14397	95	1852	0
12	15804	273	2513	0
13	14051	214	3740	0

더보기

Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

```
SELECT
  CustomerID,
  ROUND(SUM(UnitPrice),1) AS user_total
FROM clear-vision-425501-b9.modulabs_project.distinct_data
GROUP BY CustomerID;
```

작업 정보 | 결과 | 차트 | JSON

행	CustomerID	user_total
1	12544	54.3
2	13568	130.6
3	13824	126.6
4	14080	3.8
5	14336	145.4
6	14592	323.6
7	15104	365.4
8	15360	30.6
9	15872	225.7
10	16128	215.8
11	16384	115.6

- **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 clear-vision-425501-b9.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, 2) AS user_average
FROM clear-vision-425501-b9.modulabs_project.user_rf rf
LEFT JOIN (
   SELECT
      CustomerID,
      ROUND(SUM(UnitPrice),1) AS user_total
  FROM clear-vision-425501-b9.modulabs_project.distinct_data
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	15694	78	1664	0	357.1	4.58
2	15910	266	1013	0	482.3	1.81
3	15804	273	2513	0	1121.8	4.11
4	13113	278	2596	0	1031.5	3.71
5	14446	276	856	0	509.9	1.85
6	17490	85	1022	0	246.1	2.9
7	14422	222	2906	0	623.9	2.81
8	17428	343	9435	0	1249.1	3.64
9	16558	474	5346	0	1440.8	3.04
10	12423	118	1312	0	245.0	2.08
11	18102	431	64124	0	1940.9	4.5

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
SELECT *
FROM clear-vision-425501-b9.modulabs_project.user_rfm;
```

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

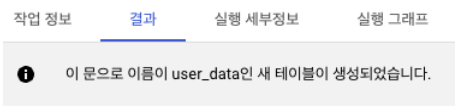
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	15694	78	1664	0	357.1	4.58
2	15910	266	1013	0	482.3	1.81
3	15804	273	2513	0	1121.8	4.11
4	13113	278	2596	0	1031.5	3.71
5	14446	276	856	0	509.9	1.85
6	17490	85	1022	0	246.1	2.9
7	14422	222	2906	0	623.9	2.81
8	17428	343	9435	0	1249.1	3.64
9	16558	474	5346	0	1440.8	3.04
10	12423	118	1312	0	245.0	2.08
11	18102	431	64124	0	1940.9	4.5

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 clear-vision-425501-b9.modulabs_project.user_data AS
WITH unique_products AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT StockCode) AS unique_products
  FROM clear-vision-425501-b9.modulabs_project.distinct_data
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM clear-vision-425501-b9.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



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

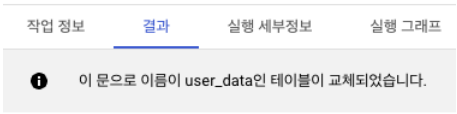
행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	14705	1	100	198	1.8	1.8	1
2	13270	1	200	366	3.0	3.0	1
3	17331	1	16	123	10.9	10.9	1
4	16323	1	50	196	4.2	4.2	1
5	15668	1	72	217	1.1	1.1	1
6	16579	1	-12	365	2.5	2.5	1
7	18141	1	-12	360	3.0	3.0	1
8	14679	1	-1	371	2.5	2.5	1
9	15313	1	25	110	2.1	2.1	1
10	15070	1	36	372	3.0	3.0	1
11	16078	1	16	283	5.0	5.0	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**

```
WITH purchase_intervals AS (
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_interval
  FROM (
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY) AS
    FROM
      clear-vision-425501-b9.modulabs_project.distinct_data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)
```

```
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM clear-vision-425501-b9.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```

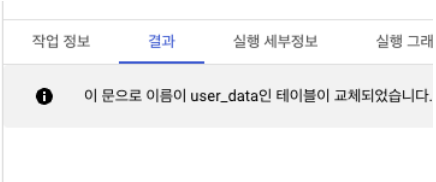


행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	12428	292	3477	25	863.2	2.96	256	0.87
2	14432	377	2013	9	552.7	1.47	256	0.2
3	13268	439	3525	17	690.9	1.57	256	0.56
4	18184	1	60	15	0.8	0.8	1	0.0
5	14351	1	12	164	4.3	4.3	1	0.0
6	15562	1	39	351	3.5	3.5	1	0.0
7	18174	1	50	7	2.1	2.1	1	0.0
8	16765	1	4	294	8.5	8.5	1	0.0
9	12814	1	48	101	1.8	1.8	1	0.0
10	15657	1	24	22	1.3	1.3	1	0.0
11	14576	1	12	372	3.0	3.0	1	0.0
12	16138	1	-1	368	8.0	8.0	1	0.0
13	16454	1	2	64	3.0	3.0	1	0.0
14	17347	1	216	86	1.1	1.1	1	0.0
15	15195	1	1404	2	2.8	2.8	1	0.0

3. 구매 취소 경향성

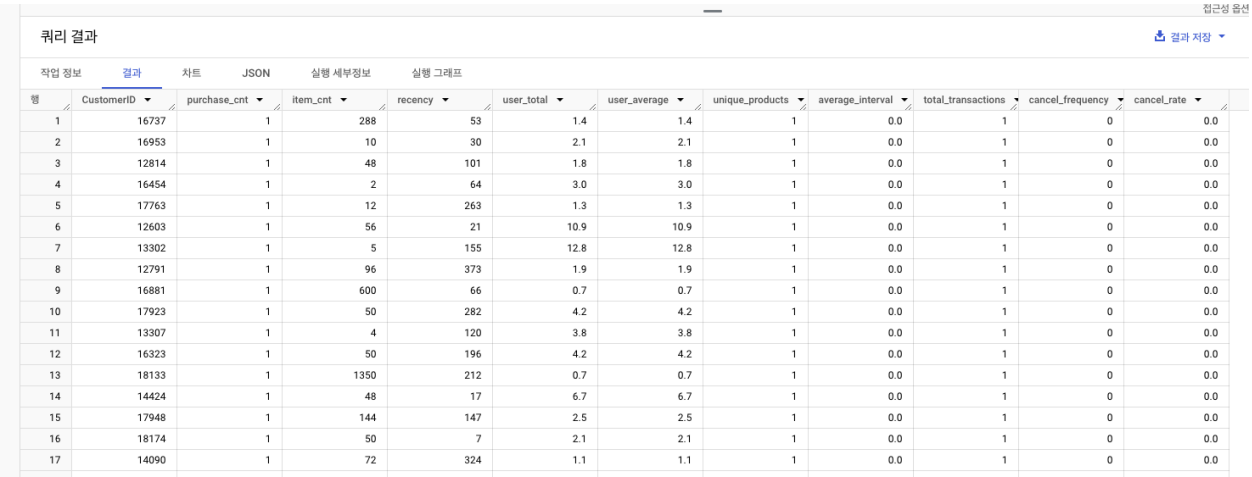
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
  - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data` 에 통합하기**
    (취소 비율은 소수점 두번째 자리)

```
CREATE OR REPLACE TABLE clear-vision-425501-b9.modulabs_project.user_data AS
WITH TransactionInfo AS (
  SELECT
    CustomerID,
    COUNT(InvoiceNo) AS total_transactions,
    COUNT(CASE WHEN Quantity <= 0 THEN 1 END) AS cancel_frequency
  FROM clear-vision-425501-b9.modulabs_project.distinct_data
  GROUP BY CustomerID
)
SELECT u.*, t.* EXCEPT(CustomerID),
ROUND(t.cancel_frequency / t.total_transactions, 2) AS cancel_rate
FROM clear-vision-425501-b9.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```



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

```
SELECT *
FROM clear-vision-425501-b9.modulabs_project.user_data;
```



쿼리 결과

결과 저장

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

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16737	1	288	53	1.4	1.4	1	0.0	1	0	0.0
2	16953	1	10	30	2.1	2.1	1	0.0	1	0	0.0
3	12814	1	48	101	1.8	1.8	1	0.0	1	0	0.0
4	16454	1	2	64	3.0	3.0	1	0.0	1	0	0.0
5	17763	1	12	263	1.3	1.3	1	0.0	1	0	0.0
6	12603	1	56	21	10.9	10.9	1	0.0	1	0	0.0
7	13302	1	5	155	12.8	12.8	1	0.0	1	0	0.0
8	12791	1	96	373	1.9	1.9	1	0.0	1	0	0.0
9	16881	1	600	66	0.7	0.7	1	0.0	1	0	0.0
10	17923	1	50	282	4.2	4.2	1	0.0	1	0	0.0
11	13307	1	4	120	3.8	3.8	1	0.0	1	0	0.0
12	16323	1	50	196	4.2	4.2	1	0.0	1	0	0.0
13	18133	1	1350	212	0.7	0.7	1	0.0	1	0	0.0
14	14424	1	48	17	6.7	6.7	1	0.0	1	0	0.0
15	17948	1	144	147	2.5	2.5	1	0.0	1	0	0.0
16	18174	1	50	7	2.1	2.1	1	0.0	1	0	0.0
17	14090	1	72	324	1.1	1.1	1	0.0	1	0	0.0