

고객을 세그멘테이션하자 [프로젝트]

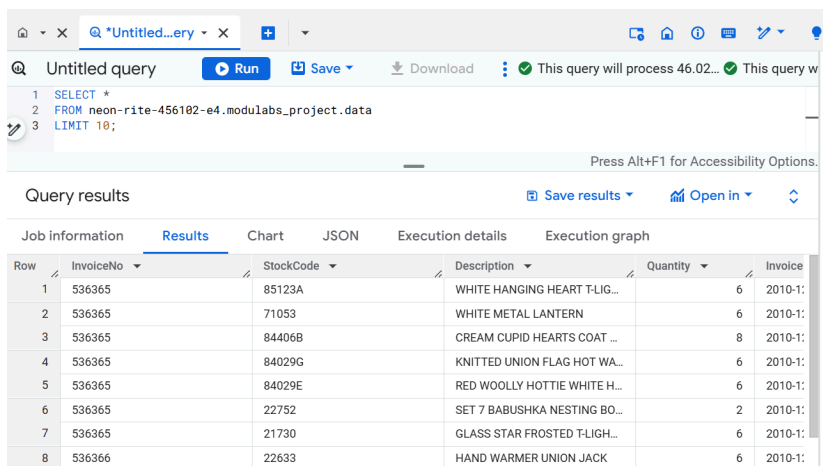
11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM neon-rite-456102-e4.modulabs_project.data  
LIMIT 10;
```

[결과 이미지를 넣어주세요]



The screenshot shows a query editor with the following SQL query:

```
1 SELECT *  
2 FROM neon-rite-456102-e4.modulabs_project.data  
3 LIMIT 10;
```

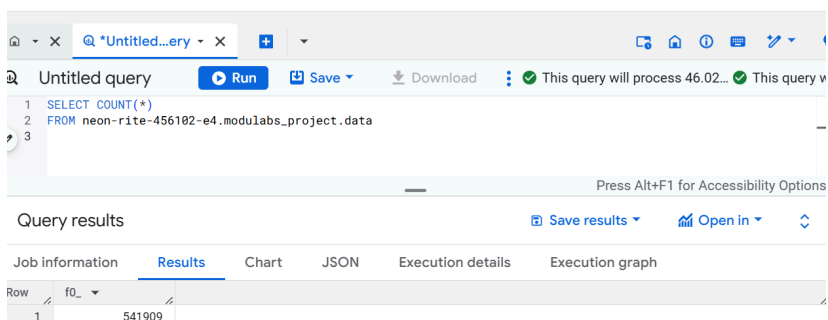
The query results are displayed in a table with the following columns: Row, InvoiceNo, StockCode, Description, Quantity, and Invoice. The results show 10 rows of data.

Row	InvoiceNo	StockCode	Description	Quantity	Invoice
1	536365	85123A	WHITE HANGING HEART T-LIG...	6	2010-1:
2	536365	71053	WHITE METAL LANTERN	6	2010-1:
3	536365	84406B	CREAM CUPID HEARTS COAT ...	8	2010-1:
4	536365	84029G	KNITTED UNION FLAG HOT WA...	6	2010-1:
5	536365	84029E	RED WOOLLY HOTTIE WHITE H...	6	2010-1:
6	536365	22752	SET 7 BABUSHKA NESTING BO...	2	2010-1:
7	536365	21730	GLASS STAR FROSTED T-LIGH...	6	2010-1:
8	536366	22633	HAND WARMER UNION JACK	6	2010-1:

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*)  
FROM neon-rite-456102-e4.modulabs_project.data
```

[결과 이미지를 넣어주세요]



The screenshot shows a query editor with the following SQL query:

```
1 SELECT COUNT(*)  
2 FROM neon-rite-456102-e4.modulabs_project.data  
3
```

The query results are displayed in a table with the following columns: Row, f0_ (representing COUNT(*)). The results show 1 row of data.

Row	f0_
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
'neon-rite-456102-e4'.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

⌵

<

11-4. 데이터 전처리 방법(1): 결측치 제거

컬럼 별 누락된 값의 비율 계산

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

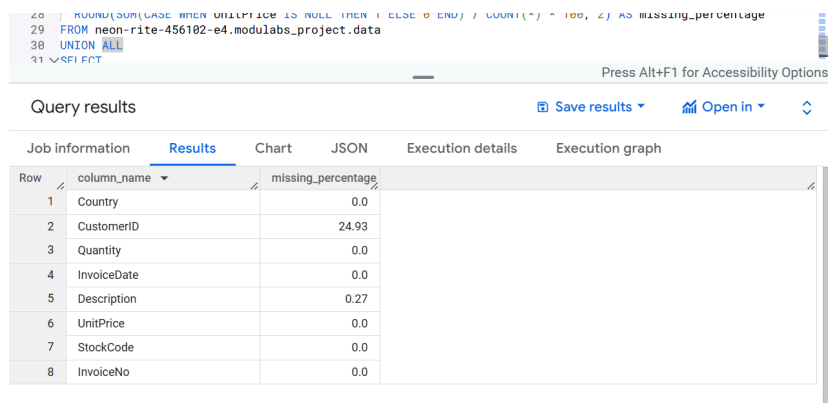
```
SELECT
'InvoiceNo' AS column_name,
ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM neon-rite-456102-e4.modulabs_project.data
UNION ALL
SELECT
'StockCode' AS column_name,
ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM neon-rite-456102-e4.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 neon-rite-456102-e4.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 neon-rite-456102-e4.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 neon-rite-456102-e4.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 neon-rite-456102-e4.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 neon-rite-456102-e4.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 neon-rite-456102-e4.modulabs_project.data

```

[결과 이미지를 넣어주세요]



The screenshot shows a SQL query editor with a query that calculates the missing percentage for various columns. Below the editor, the 'Query results' tab is active, displaying a table with 8 rows. The columns are 'column_name' and 'missing_percentage'. The rows list: Country (0.0), CustomerID (24.93), Quantity (0.0), InvoiceDate (0.0), Description (0.27), UnitPrice (0.0), StockCode (0.0), and InvoiceNo (0.0).

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

결측치 처리 전략

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

```

SELECT DISTINCT Description
FROM neon-rite-456102-e4.modulabs_project.data
WHERE StockCode = '85123A';

```

[결과 이미지를 넣어주세요]

LMS 예시와 다르게 ascending sort 했습니다.

Query results

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	Description				
1	?				
2	CREAM HANGING HEART T-LIGHT HOLDER				
3	WHITE HANGING HEART T-LIGHT HOLDER				
4	wrongly marked carton 22804				

결측치 처리

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

```
DELETE FROM neon-rite-456102-e4.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL;
```

[결과 이미지를 넣어주세요]

영어환경이어 결과가 영어로 나옵니다.

Press Alt+F1 for Accessibility Options

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>i This statement removed 135,080 rows from data. Go to table</p>			

11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
SELECT *
FROM neon-rite-456102-e4.modulabs_project.data
GROUP BY
  InvoiceNo,
  StockCode,
  Description,
  Quantity,
  InvoiceDate,
  UnitPrice,
  CustomerID,
  Country
HAVING COUNT(*) > 1;
```

[결과 이미지를 넣어주세요]

마지막 행이 보이게 이미지 잘랐습니다.

Query results

Save results Open in

Job information Results Chart JSON Execution details Execution graph

Row	InvoiceNo	StockCode	Description	Quantity	InvoiceDate
4831	575668	23209	LUNCH BAG VINTAGE DOLLY	1	2011-11-10 14:59:
4832	578262	22383	LUNCH BAG SUKI DESIGN	1	2011-11-23 13:27:
4833	578262	22952	60 CAKE CASES VINTAGE CHR...	5	2011-11-23 13:27:
4834	578262	84992	72 SWEETHEART FAIRY CAKE ...	1	2011-11-23 13:27:
4835	578262	21212	PACK OF 72 RETROSPOT CAKE...	1	2011-11-23 13:27:
4836	578262	84991	60 TEATIME FAIRY CAKE CASES	2	2011-11-23 13:27:
4837	579673	23208	LUNCH BAG VINTAGE LEAF DE...	1	2011-11-30 12:59:

Results per page: 50 4801 - 4837 of 4837

중복값 처리

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

```
CREATE OR REPLACE TABLE neon-rite-456102-e4.modulabs_project.data AS
SELECT
  DISTINCT *
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

영문환경이어서 결과가 영어로 나옵니다.

Query results

Save results Open in

Job information Results Execution details Execution graph

This statement replaced the table named data.

Go to table

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
SELECT COUNT(DISTINCT InvoiceNo)
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Query results

Save results Open in

Job information Results Chart JSON Execution details Execution graph

Row	f0_
1	22190

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM neon-rite-456102-e4.modulabs_project.data
LIMIT 100;
```

[결과 이미지를 넣어주세요]

LMS의 예시와 다르지만 C로 시작되는 invoice 찾을 수 있습니다.

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	InvoiceNo
93	C572532
94	563100
95	570681
96	570725
97	574694
98	580638
99	C565050
100	539840

Results per page: 100 1 - 100 of 100

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

```
SELECT *
FROM neon-rite-456102-e4.modulabs_project.data
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

마지막 100이 보이게 이미지 잘랐습니다.

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
93	C560540	22546	MINI JIGSAW ...	-1	2011-07-19 12:2...	0.42	12415	Australia
94	C560540	23237	SET OF 4 KNIC...	-1	2011-07-19 12:2...	4.15	12415	Australia
95	C560540	23242	TREASURE TIN...	-1	2011-07-19 12:2...	2.08	12415	Australia
96	C560540	23291	DOLLY GIRL C...	-1	2011-07-19 12:2...	1.25	12415	Australia
97	C560540	22492	MINI PAINT SE...	-36	2011-07-19 12:2...	0.65	12415	Australia
98	C560540	23293	SET OF 12 FAIR...	-2	2011-07-19 12:2...	0.83	12415	Australia
99	C560540	23192	BUNDLE OF 3 ...	-1	2011-07-19 12:2...	1.65	12415	Australia
100	C560540	20979	36 PENCILS TU...	-1	2011-07-19 12:2...	1.25	12415	Australia

Results per page: 100 1 - 100 of 100

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

```
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END)/ COUNT(*), 3)
FROM neon-rite-456102-e4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

Big Query에서 percentage로 결과를 보여주는 함수를 찾을 수 없어서 소수점 세째 자리에서 반올림 하여 결과 도출했습니다.

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	f0_
1	0.022

StockCode 살펴보기

- 고유한 StockCode 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode)
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Query results		Save results	Open in
Job information	Results	Chart	JSON
Execution details	Execution graph		
Row	f0_		
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 neon-rite-456102-e4.modulabs_project.data
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```

[결과 이미지를 넣어주세요]

Query results		Save results	Open in
Job information	Results	Chart	JSON
Execution details	Execution graph		
Row	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 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 neon-rite-4456102-e4.modulabs_project.data
)
WHERE number_count BETWEEN 0 AND 1;
```

[결과 이미지를 넣어주세요]

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	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
  ROUND(SUM(CASE WHEN StockCode IN ('POST', 'M', 'C2','D','BANK CHARGES','PADS','DOT','CRUK') THEN 1
    ELSE 0 END) / COUNT(*),4)
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Big Query에서 percentage로 결과를 보여주는 함수를 찾을 수 없어서 소수점 네째 자리에서 반올림 하여 결과 도출했습니다.

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	f0_	
1	0.0048	

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM neon-rite-456102-e4.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 neon-rite-456102-e4.modulabs_project.data)
  WHERE number_count BETWEEN 0 AND 1);
```

[결과 이미지를 넣어주세요]

Query results			Save results	Open in
Job information	Results	Execution details	Execution graph	
This statement removed 1,915 rows from data.				
Go to table				

Description 살펴보기

- 고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력하기


```
SELECT Description, COUNT(*) AS description_cnt
FROM neon-rite-456102-e4.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

[결과 이미지를 넣어주세요]

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	Description	description_cnt			
1	WHITE HANGING HEART T-LIGHT HOLDER	2058			
2	REGENCY CAKESTAND 3 TIER	1894			
3	JUMBO BAG RED RETROSPOT	1659			
4	PARTY BUNTING	1409			
5	ASSORTED COLOUR BIRD ORNAMENT	1405			
6	LUNCH BAG RED RETROSPOT	1345			
7	SET OF 3 CAKE TINS PANTRY DESIGN	1224			
8	LUNCH BAG BLACK SKULL	1099			
9	PACK OF 72 RETROSPOT CAKE CASES	1062			
10	SPOTTY BUNTING	1026			

Results per page: 50 1 - 30 of 30

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE
FROM neon-rite-456102-e4.modulabs_project.data
WHERE Description IN('High Resolution Image','Next Day Carriage')
```

[결과 이미지를 넣어주세요]

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>This statement removed 83 rows from data.</p> <p>Go to table</p>			

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE neon-rite-456102-e4.modulabs_project.data AS
SELECT
* EXCEPT (Description),
UPPER(Description) AS Description
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Query results Save results Open in

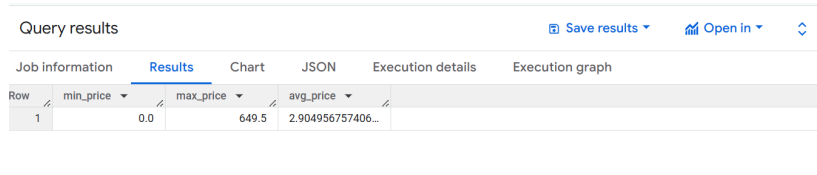
Job information	Results	Execution details	Execution graph
<p>This statement replaced the table named data.</p> <p>Go to table</p>			

UnitPrice 살펴보기

- UnitPrice 의 최소값, 최대값, 평균을 구하기

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

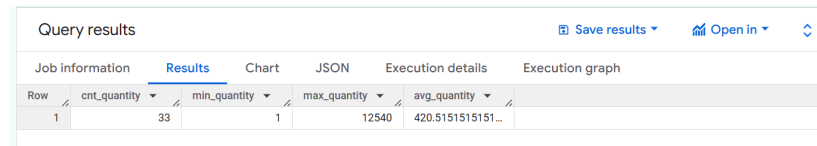


Row	min_price	max_price	avg_price
1	0.0	649.5	2.904956757406...

- 단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최소값, 최대값, 평균 구하기

```
SELECT COUNT(Quantity) AS cnt_quantity, MIN(Quantity) AS min_quantity, MAX(Quantity) AS max_quantity, AVG(Quantity) AS avg_
FROM neon-rite-456102-e4.modulabs_project.data
WHERE UnitPrice=0
```

[결과 이미지를 넣어주세요]



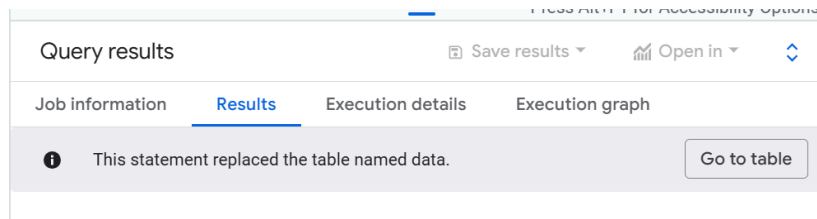
Row	cnt_quantity	min_quantity	max_quantity	avg_quantity
1	33	1	12540	420.5151515151...

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE neon-rite-456102-e4.modulabs_project.data AS
SELECT *
FROM neon-rite-456102-e4.modulabs_project.data
WHERE UnitPrice >0;
```

[결과 이미지를 넣어주세요]

데이터 처리후 남은 행은 399,573으로 확인됩니다.



Job information	Results	Execution details	Execution graph
This statement replaced the table named data.			

Query results							
Job information				Execution details			
Row	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom
2	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
3	537626	21064	6	2010-12-07 14:57:00 UTC	5.95	12347	Iceland
4	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
5	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland
6	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	12347	Iceland
7	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland
8	537626	84997B	6	2010-12-07 14:57:00 UTC	2.75	12347	Iceland

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Query results							
Job information				Execution details			
Row	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate		
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC		
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC		
3	2010-12-07	537626	21064	6	2010-12-07 14:57:00 UTC		
4	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC		
5	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC		
6	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC		
7	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC		
8	2010-12-07	537626	84997B	6	2010-12-07 14:57:00 UTC		

- 가장 최근 구매 일자를 **MAX()** 함수로 찾아보기

```
SELECT MAX(InvoiceDate) AS most_recent_date
FROM neon-rite-456102-e4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

Query results							
Job information				Execution details			
Row	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 neon-rite-456102-e4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

Query results [Save results](#) [Open in](#)

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	InvoiceDay
1	12346	2011-01-18
2	12347	2011-12-07
3	12348	2011-09-25
4	12349	2011-11-21
5	12350	2011-02-02
6	12352	2011-11-03
7	12353	2011-05-19
8	12354	2011-04-21
9	12355	2011-05-09
10	12356	2011-11-17

Results per page: 50 1 - 50 of 4362

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

```
CREATE OR REPLACE TABLE neon-rite-456102-e4.modulabs_project.user_r AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(InvoiceDate) AS InvoiceDay
  FROM neon-rite-456102-e4.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

Query results [Save results](#) [Open in](#)

Job information Results Chart JSON Execution details Execution graph

Row	CustomerID	recency
1	12347	2
2	12406	22
3	12631	57
4	12837	173
5	12840	143
6	12922	161
7	12989	3
8	12999	196
9	13339	200
10	13509	8

Results per page: 50 1 - 50 of 4362

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

[결과 이미지를 넣어주세요]

Query results	Save results	Open in
Job information	Results	Execution details
This statement created a new table named user_r. Go to table		

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM neon-rite-456102-e4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

Query results	Save results	Open in
Job information	Results	Chart
JSON	Execution details	Execution graph
Row	CustomerID	purchase_cnt
1	12346	2
2	12347	7
3	12348	4
4	12349	1
5	12350	1
6	12352	8
7	12353	1
8	12354	1
9	12355	1
10	12356	3
Results per page: 50 1 - 50 of 4362		

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID, SUM(Quantity) AS item_cnt
FROM neon-rite-456102-e4.modulabs_project.data
GROUP BY CustomerID;
```

[결과 이미지를 넣어주세요]

Query results	Save results	Open in
Job information	Results	Chart
JSON	Execution details	Execution graph
Row	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
Results per page: 50 1 - 50 of 4362		

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

```

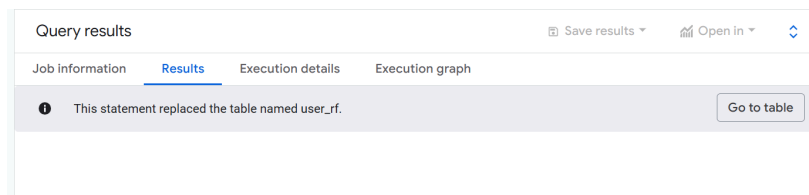
CREATE OR REPLACE TABLE neon-rite-456102-e4.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM
    `neon-rite-456102-e4`.modulabs_project.data
  GROUP BY
    CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    SUM(Quantity) AS item_cnt
  FROM
    `neon-rite-456102-e4`.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 neon-rite-456102-e4.modulabs_project.user_r AS ur
  ON pc.CustomerID = ur.CustomerID;

```

[결과 이미지를 넣어주세요]



Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```

SELECT
  CustomerID,
  # [[YOUR QUERY]] AS user_total
FROM project_name.modulabs_project.data
# [[YOUR QUERY]];

```

[결과 이미지를 넣어주세요]

- 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 project_name.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  # [[YOUR QUERY]] AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    # [[YOUR QUERY]]
  ) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- 최종 `user_rfm` 테이블을 출력하기

```
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

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

```
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```

[결과 이미지를 넣어주세요]

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 project_name.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
      project_name.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

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

[결과 이미지를 넣어주세요]

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 project_name.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    # [[YOUR QUERY]] AS total_transactions,
    # [[YOUR QUERY]] AS cancel_frequency
  FROM project_name.modulabs_project.data
  # [[YOUR QUERY]]
)
```



```
SELECT u.*, t.* EXCEPT(CustomerID), # [[YOUR QUERY]] AS cancel_rate
FROM `project_name.modulabs_project.user_data` AS u
LEFT JOIN TransactionInfo AS t
ON # [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

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

```
# [[YOUR QUERY]];
```

[결과 이미지를 넣어주세요]

회고

[회고 내용을 작성해주세요]

Keep : step by step 학습 내용을 잘 따라갔고, 몇개의 명령어는 익숙해졌다. 자잘한 실수도 줄어들었다.(e.g. 콤마 사용, 괄호사용, 따옴표 사용등과 파일명 변경등)

Problem : WITH 구문등 좀 더 익숙해져야 할 명령문이 많다.

Try : 자주 연습해보는 방법이 최선 인 것 같다. 시간을 더 들여야 할듯.