PDA

고객을 세그먼테이션하자 [프로젝트]

5-2. 데이터 불러오기

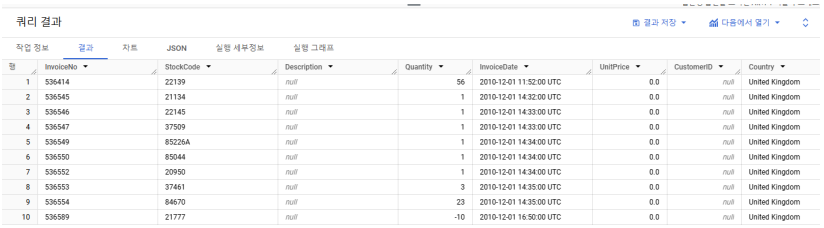
데이터 살펴보기

- **테이블에 있는 10개의 행만 출력하기**

```
# [[YOUR QUERY]]

select *
from 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 modulabs_project.data
limit 10 ;
```

[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 `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_INVOICEDAT	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]]

ELECT
    'InvoiceNo' AS column_name,
    ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percent
FROM 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 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 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 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 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 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_percen
FROM 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 modulabs_project.data
```

[결과 이미지를 넣어주세요]

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

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

결측치 처리 전략

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

```
SELECT StockCode, Description
FROM modulabs_project.data
where StockCode = '85123A'
```

[결과 이미지를 넣어주세요]

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

	StockCode	Description
1	85123A	?
2	85123A	wrongly marked carton 22804
3	85123A	CREAM HANGING HEART T-LIG...
4	85123A	CREAM HANGING HEART T-LIG...
5	85123A	CREAM HANGING HEART T-LIG...
6	85123A	CREAM HANGING HEART T-LIG...
7	85123A	CREAM HANGING HEART T-LIG...
8	85123A	CREAM HANGING HEART T-LIG...
9	85123A	CREAM HANGING HEART T-LIG...
10	85123A	CREAM HANGING HEART T-LIG...
11	85123A	CREAM HANGING HEART T-LIG

결측치 처리

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

```
DELETE FROM project_name.modulabs_project.data
WHERE CustomerID IS NULL OR Description IS NULL
```

[결과 이미지를 넣어주세요]

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 *,COUNT(*)
FROM modulabs_project.data
GROUP BY InvoiceNo, StockCode, Description,Quantity, InvoiceDate, UnitPrice,CustomerID, Country
HAVING COUNT(*) > 1
```

[결과 이미지를 넣어주세요]

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
41	558700	21544	SKULLS WATER TRANSFER TA...	3	2011-07-01 12:33:00 UTC	0.85	17920	United Kingdom
42	558700	22553	PLASTERS IN TIN SKULLS	1	2011-07-01 12:33:00 UTC	1.65	17920	United Kingdom
43	558700	22332	SKULLS PARTY BAG + STICKE...	1	2011-07-01 12:33:00 UTC	1.65	17920	United Kingdom
44	558700	21242	RED RETROSPOT PLATE	1	2011-07-01 12:33:00 UTC	1.69	17920	United Kingdom
45	560937	22543	MINI JIGSAW BAKE A CAKE	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom
46	560937	21311	SET/4 BIRD MIRROR MAGNETS	1	2011-07-22 10:52:00 UTC	0.29	17920	United Kingdom
47	560937	22541	MINI JIGSAW LEAP FROG	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom
48	560937	22539	MINI JIGSAW DOLLY GIRL	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom
49	560937	23154	SET OF 4 JAM JAR MAGNETS	1	2011-07-22 10:52:00 UTC	2.08	17920	United Kingdom
50	560937	22546	MINI JIGSAW PURDEY	1	2011-07-22 10:52:00 UTC	0.42	17920	United Kingdom

페이지당 결과 수: 50 1 – 50 (전체 4837행)

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_distinct AS
SELECT DISTINCT *
FROM modulabs_project.data;
```

[결과 이미지를 넣어주세요]

행	f0_
1	401604

5-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

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

```
# [[YOUR QUERY]]

SELECT COUNT(DISTINCT InvoiceNo)
FROM modulabs_project.data_distinct
```

[결과 이미지를 넣어주세요]

작업 정보 결과 차트 JSON

행	uc
1	22190

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

```
# [[YOUR QUERY]]

select DISTINCT InvoiceNo
FROM modulabs_project.data_distinct
limit 100;
```

[결과 이미지를 넣어주세요]

작업 정보 | 결과 | 차트 | JSON | 실행 세부정보

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

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

```
SELECT *
FROM modulabs_project.data_distinct
WHERE InvoiceNo like 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

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

	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	21485	RETROSPOT HEART HOT WAT...	-1	2010-12-21 12:33:00 UTC	4.95	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	22832	BROCANTE SHELF WITH HOOKS	-2	2010-12-21 12:33:00 UTC	10.75	18176	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)
FROM modulabs_project.data_distinct
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 차트 | JSON

	f0_
1	2.2

`StockCode` 살펴보기

- **고유한 `StockCode` 의 개수를 출력하기**

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

```
SELECT COUNT(DISTINCT(STOCKCODE))
FROM modulabs_project.data_distinct
```

[결과 이미지를 넣어주세요]

- **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_distinct
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10
```

[결과 이미지를 넣어주세요]

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

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

- **`StockCode` 의 문자열 내 숫자의 길이를 구해보기**

```
WITH UniqueStockCodes AS (
  SELECT DISTINCT StockCode
  FROM project_name.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]

- **`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 project_name.modulabs_project.data
)
WHERE number_count <=1
```

[결과 이미지를 넣어주세요]

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
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]', '')) <=1
THEN 1 ELSE 0 END) / COUNT(*) *100,2)
FROM modulabs_project.data_distinct
```

[결과 이미지를 넣어주세요]

쿼리 결과

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

행	f0_
1	0.48

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
DELETE FROM modulabs_project.dt
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM modulabs_project.dt
)
WHERE number_count <=1)
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 | 결과 | 실행 세부정보 | 실행 그래프

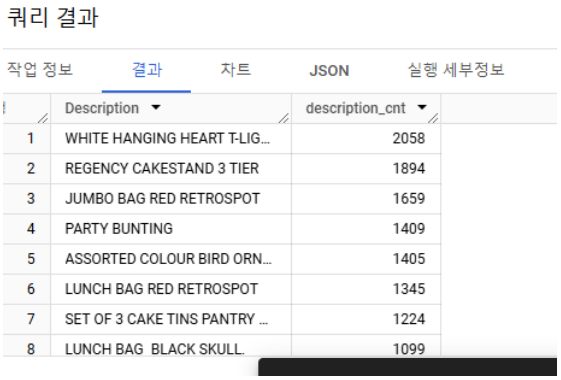
이 문으로 dt의 행 1,915개가 삭제되었습니다.

`Description` 살펴보기

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

```
SELECT Description, COUNT(*) AS description_cnt
FROM modulabs_project.dt
GROUP BY Description
order by description_cnt DESC
LIMIT 30 ;
```

[결과 이미지를 넣어주세요]

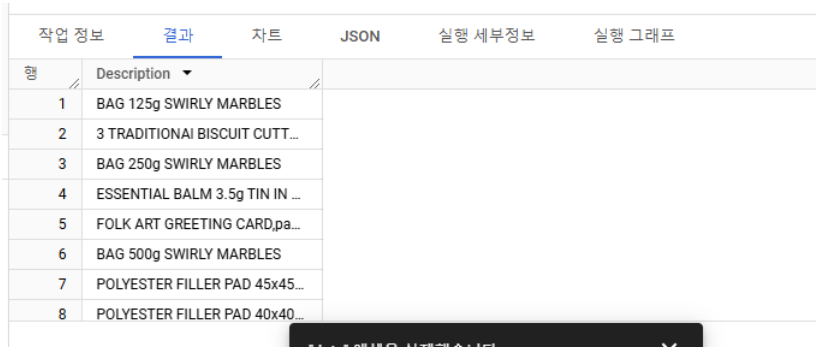


	Description	description_cnt
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	LUNCH BAG BLACK SKULL.	1099

- **대소문자가 혼합된 Description이 있는지 확인하기**

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

[결과 이미지를 넣어주세요]

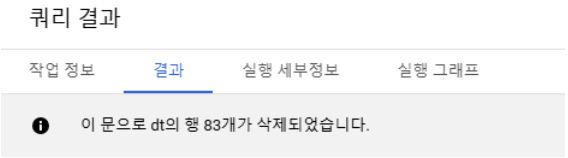


행	Description
1	BAG 125g SWIRLY MARBLES
2	3 TRADITIONAI BISCUIT CUTT...
3	BAG 250g SWIRLY MARBLES
4	ESSENTIAL BALM 3.5g TIN IN ...
5	FOLK ART GREETING CARD,pa...
6	BAG 500g SWIRLY MARBLES
7	POLYESTER FILLER PAD 45x45...
8	POLYESTER FILLER PAD 40x40...

- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
DELETE
FROM project_name.modulabs_project.data
WHERE Description IN('Next Day Carriage','High Resolution Image' )
```

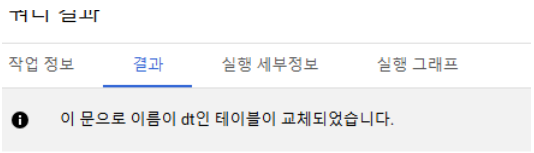
[결과 이미지를 넣어주세요]



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

```
CREATE OR REPLACE TABLE modulabs_project.dt AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM modulabs_project.dt
```

[결과 이미지를 넣어주세요]

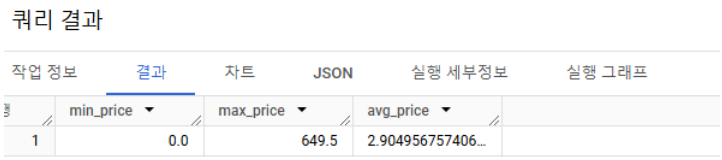


`UnitPrice` 살펴보기

- `UnitPrice` **의 최솟값, 최댓값, 평균을 구하기**

```
SELECT MIN(UnitPrice) AS min_price, MAX(UnitPrice) AS max_price, AVG(UnitPrice) AS avg_price
FROM project_name.modulabs_project.data;
```

[결과 이미지를 넣어주세요]



- **단가가 0원인 거래의 개수, 구매 수량(** `Quantity` **)의 최솟값, 최댓값, 평균 구하기**

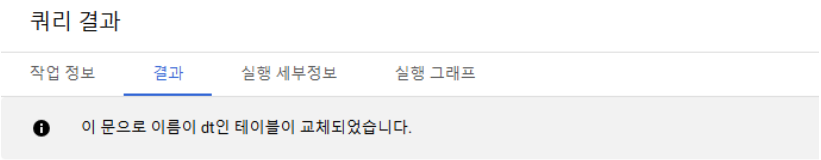
```
SELECT COUNT(Quantity) AS cnt_quantity, min(Quantity) AS min_quantity,MAX(Quantity) max_quantity, AV
FROM modulabs_project.dt
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

- `UnitPrice = 0` **를 제거하고 일관된 데이터셋을 유지하기**

```
CREATE OR REPLACE TABLE project_name.modulabs_project.data AS
SELECT *
FROM project_name.modulabs_project.data
WHERE UnitPrice > 0
```

[결과 이미지를 넣어주세요]



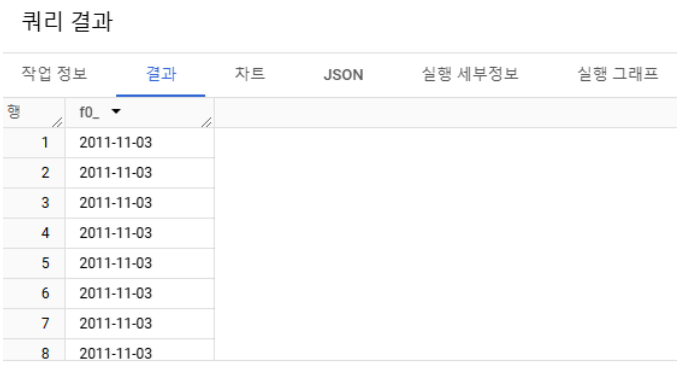
5-7. RFM 스코어

Recency

- `InvoiceDate` **컬럼을 연월일 자료형으로 변경하기**

```
SELECT DATE(InvoiceDate) , *
FROM modulabs_project.dt
```

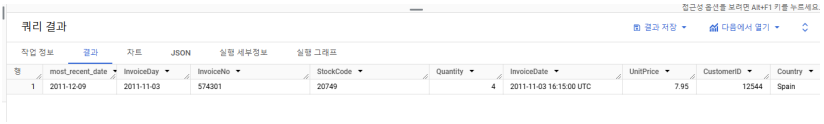
[결과 이미지를 넣어주세요]



- **가장 최근 구매 일자를 MAX() 함수로 찾아보기**

```
SELECT
   (SELECT MAX(DATE(InvoiceDate)) FROM modulabs_project.dt) AS most_recent_date,
   DATE(InvoiceDate) AS InvoiceDay,
   *
FROM modulabs_project.dt
ORDER BY most_recent_date
LIMIT 1
;
```

[결과 이미지를 넣어주세요]



쿼리 결과

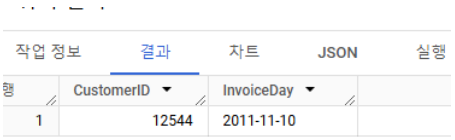
작업 정보 | 결과 | 차트 | JSON | 실행 세부정보 | 실행 그래프

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	2011-12-09	2011-11-03	574301	20749	4	2011-11-03 16:15:00 UTC	7.95	12544	Spain

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

```
SELECT
    CustomerID,
    MAX(date(invoicedate)) AS InvoiceDay
FROM modulabs_project.dt
GROUP BY CustomerID
limit 1;
```

[결과 이미지를 넣어주세요]



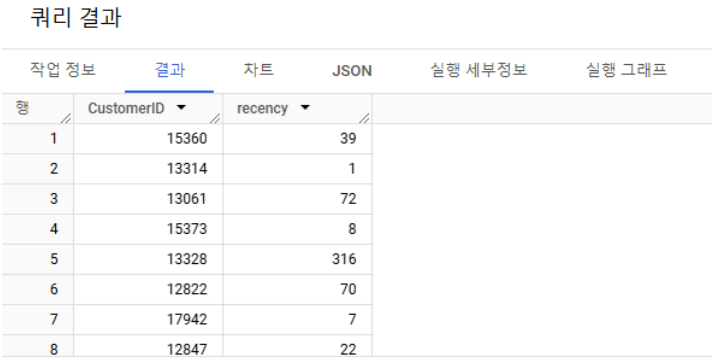
작업 정보 | 결과 | 차트 | JSON | 실행

행	CustomerID	InvoiceDay
1	12544	2011-11-10

- **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
);
```

[결과 이미지를 넣어주세요]



쿼리 결과

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

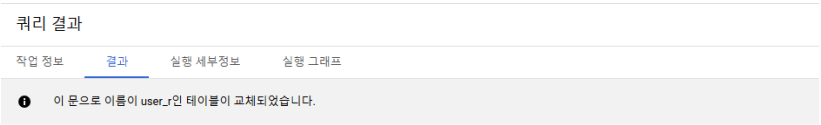
행	CustomerID	recency
1	15360	39
2	13314	1
3	13061	72
4	15373	8
5	13328	316
6	12822	70
7	17942	7
8	12847	22

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

```
CCREATE OR REPLACE TABLE main.user_r AS
SELECT *
FROM (
SELECT
CustomerID,
MAX(DATE(InvoiceDate)) AS InvoiceDay,EXTRACT(DAY FROM (SELECT MAX(DATE(InvoiceDate)) FROM main.quest
```

```
FROM main.quest6
GROUP BY CustomerID
)
```

[결과 이미지를 넣어주세요]



Frequency

- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
SELECT
  CustomerID,
  COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM  modulabs_project.dt
GROUP BY CustomerID
```

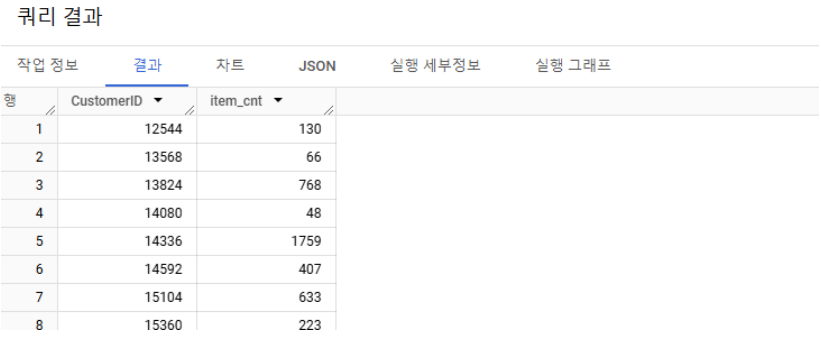
[결과 이미지를 넣어주세요]

행	CustomerID	purchase_cnt
1	12544	2
2	13568	1
3	13824	5
4	14080	1
5	14336	4
6	14592	3
7	15104	3
8	15360	1

- **각 고객 별로 구매한 아이템의 총 수량 더하기**

```
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM modulabs_project.dt
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
6	14592	407
7	15104	633
8	15360	223

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

```
CREATE OR REPLACE TABLE project_name.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
 SELECT
  CustomerID,
  COUNT(InvoiceNo) AS item_cnt
FROM modulabs_project.dt
GROUP BY  CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
SELECT
  CustomerID,
  SUM(Quantity) AS item_cnt
FROM modulabs_project.dt
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 project_name.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,
  ROUND(SUM(UNITPRICE),1) AS user_total
FROM modulabs_project.dt
GROUP BY CustomerID
```

[결과 이미지를 넣어주세요]

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

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

- **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,
 round(ut.user_toal)/rf.purchase_cnt, AS user_average
FROM project_name.modulabs_project.user_rf rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
    CustomerID
  round(sum(Quantity+UNITPRICE),1) as user_total
from modulabs_project.dt
GROUP by CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```

[결과 이미지를 넣어주세요]

RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

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

[결과 이미지를 넣어주세요]

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 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]

	customerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	17923	1	50	282	208.0	208.0	1
2	15753	1	144	304	79.0	79.0	1
3	18113	1	72	368	76.0	76.0	1
4	13307	1	4	120	15.0	15.0	1
5	17925	1	72	372	244.0	244.0	1
6	13017	1	48	7	204.0	204.0	1
7	16323	1	50	196	208.0	208.0	1
8	14679	1	-1	371	-3.0	-3.0	1
9	16738	1	3	297	4.0	4.0	1
10	12814	1	48	101	86.0	86.0	1
11	16737	1	288	53	418.0	418.0	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 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_inte
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DAY)
    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]

1	14432	6	2013	9	2248.0	375.0	256
2	12428	11	3477	25	6366.0	579.0	256
3	13268	14	3525	17	3106.0	222.0	256
4	16738	1	3	297	4.0	4.0	1
5	17925	1	72	372	244.0	244.0	1
6	17715	1	384	200	326.0	326.0	1
7	13120	1	12	238	31.0	31.0	1
8	15195	1	1404	2	3861.0	3861.0	1
9	13135	1	4300	196	3096.0	3096.0	1
10	13302	1	5	155	64.0	64.0	1

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 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]

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	COUNT_invoiceNO	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_INVOICEDAT	COUNT_UnitPrice	COUNT_CustomerID	COUNT_COUNTRY
1	541909	541909	540455	541909	541909	541909	406829	541909

작업 정보 결과 차트 JSON 실행 세부정보 실행 그래프

행	COUNT_invoiceNO	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_INVOICEDAT	COUNT_UnitPrice	COUNT_CustomerID	COUNT_COUNTRY
1	541909	541909	540455	541909	541909	541909	406829	541909

-- 좋았던점

sql을 활용해서 고객 정보를 전처리하고 필요한 정보를 찾아내는 과정을 배우게 되서 좋았습니다.

또 수업시간 이후 사람들끼리 모여 토론하며 문제푸는 방식이 좋았습니다.

— 아쉬웠던 점

5-7에서 부터 오류가 나서 오류를 찾지 못해 다음 문제들을 못푼 문제들이 있어 아쉬웠습니다.