

고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
SELECT *  
FROM capable-sled-456102-t0.modulabs_project.data  
LIMIT 10;
```

행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00 UTC	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00 UTC	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00 UTC	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00 UTC	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00 UTC	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00 UTC	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00 UTC	1.69	13047	United Kingdom

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
SELECT COUNT(*) AS row_count  
FROM capable-sled-456102-t0.modulabs_project.data;
```

행	row_count
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 capable-sled-456102-t0.modulabs_project.data;
```

-- Description 컬럼과 CustomerID 컬럼에 누락된 값 존재

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

```
-- InvoiceNo  
SELECT 'InvoiceNo' AS column_name,  
       ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1  
                   ELSE 0  
                   END) / COUNT(*) * 100, 2) AS missing_percentage
```

```

FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- StockCode
SELECT 'StockCode',
      ROUND(SUM(CASE WHEN StockCode IS NULL THEN 1
                    ELSE 0
                  END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- Description
SELECT 'Description',
      ROUND(SUM(CASE WHEN Description IS NULL THEN 1
                    ELSE 0
                  END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- Quantity
SELECT 'Quantity',
      ROUND(SUM(CASE WHEN Quantity IS NULL THEN 1
                    ELSE 0
                  END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- InvoiceDate
SELECT 'InvoiceDate',
      ROUND(SUM(CASE WHEN InvoiceDate IS NULL THEN 1
                    ELSE 0
                  END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- UnitPrice
SELECT 'UnitPrice',
      ROUND(SUM(CASE WHEN UnitPrice IS NULL THEN 1
                    ELSE 0
                  END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- CustomerID

```

```

SELECT 'CustomerID',
       ROUND(SUM(CASE WHEN CustomerID IS NULL THEN 1
                     ELSE 0
                     END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data
UNION ALL
-- Country
SELECT 'Country',
       ROUND(SUM(CASE WHEN Country IS NULL THEN 1
                     ELSE 0
                     END) / COUNT(*) * 100, 2)
FROM capable-sled-456102-t0.modulabs_project.data;

```

행	column_name ▼	missing_percentage
1	Country	0.0
2	CustomerID	24.93
3	InvoiceDate	0.0
4	Quantity	0.0
5	UnitPrice	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 DISTINCT Description
FROM capable-sled-456102-t0.modulabs_project.data
WHERE StockCode = '85123A';

```

행	Description ▼
1	WHITE HANGING HEART T-LIGHT HOLDER
2	?
3	wrongly marked carton 22804
4	CREAM HANGING HEART T-LIGHT HOLDER

결측치 처리

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

```
DELETE FROM capable-sled-456102-t0.modulabs_project.data
WHERE CustomerID IS NULL
OR Description IS NULL;
```

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 COUNT(*) AS duplicate_count
FROM (
    SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate,
           UnitPrice, CustomerID, Country,
           COUNT(*) AS cnt
    FROM capable-sled-456102-t0.modulabs_project.data
    GROUP BY InvoiceNo, StockCode, Description, Quantity, InvoiceDate,
             UnitPrice, CustomerID, Country
    HAVING cnt > 1
);

```

행	duplicate_count	
1	4837	

중복값 처리

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

```

CREATE OR REPLACE TABLE `capable-sled-456102-t0.modulabs_project.data` AS
SELECT DISTINCT *
FROM capable-sled-456102-t0.modulabs_project.data;

-- 중복값 처리 후 전체 데이터 행 개수 확인

```

i 이 문으로 이름이 data인 테이블이 교체되었습니다.

행	row_count ▼	
1	401604	

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 capable-sled-456102-t0.modulabs_project.data;
```

행	unique_invoice_count ▼	
1	22190	

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
SELECT DISTINCT InvoiceNo
FROM capable-sled-456102-t0.modulabs_project.data
LIMIT 100;
```

-- InvoiceNo를 기준으로 정렬하면

-- InvoiceNo가 'C'로 시작하는 행이 있는지 조회하기에 적합하지 않아 정렬 없이 출력

행	InvoiceNo
1	541431
2	C541433
3	537626
4	542237
5	549222
6	556201
7	562032
8	573511
9	581180
10	539318
11	541998
12	548955
13	568172
14	577609
15	543037
16	544156

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

```
SELECT *  
FROM capable-sled-456102-t0.modulabs_project.data  
WHERE InvoiceNo LIKE 'C%'  
LIMIT 100;
```


행	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom
2	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	183.75	12352	Norway
3	C545329	M	Manual	-1	2011-03-01 15:47:00 UTC	280.05	12352	Norway
4	C545330	M	Manual	-1	2011-03-01 15:49:00 UTC	376.5	12352	Norway
5	C547388	22701	PINK DOG BOWL	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
6	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03-22 16:07:00 UTC	4.95	12352	Norway
7	C547388	22413	METAL SIGN TAKE IT OR LEAVE IT	-6	2011-03-22 16:07:00 UTC	2.95	12352	Norway
8	C547388	84050	PINK HEART SHAPE EGG FRYING PAN	-12	2011-03-22 16:07:00 UTC	1.65	12352	Norway
9	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03-22 16:07:00 UTC	1.25	12352	Norway
10	C547388	22645	CERAMIC HEART FAIRY CAKE MONEY BANK	-12	2011-03-22 16:07:00 UTC	1.45	12352	Norway
11	C547388	37448	CERAMIC CAKE DESIGN SPOTTED MUG	-12	2011-03-22 16:07:00 UTC	1.49	12352	Norway
12	C549955	22666	RECIPE BOX PANTRY YELLOW DESIGN	-2	2011-04-13 13:38:00 UTC	2.95	12359	Cyprus
13	C549955	22839	3 TIER CAKE TIN GREEN AND CREAM	-2	2011-04-13 13:38:00 UTC	14.95	12359	Cyprus
14	C580165	22797	CHEST OF DRAWERS GINGHAM HEART	-2	2011-12-02 11:21:00 UTC	16.95	12359	Cyprus
15	C580165	22720	SET OF 3 CAKE TINS PANTRY DESIGN	-1	2011-12-02 11:21:00 UTC	4.95	12359	Cyprus
16	C580165	23245	SET OF 3 REGENCY CAKE TINS	-2	2011-12-02 11:21:00 UTC	4.95	12359	Cyprus
17	C580165	22826	LOVE SEAT ANTIQUE WHITE METAL	-1	2011-12-02 11:21:00 UTC	42.5	12359	Cyprus
18	C544002	22272	SET OF 3 CAKE TINS PANTRY DESIGN	-1	2011-03-22 16:07:00 UTC	2.95	12352	Norway

- 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 cancel_percent
FROM capable-sled-456102-t0.modulabs_project.data;
```

행	cancel_percent
1	2.2

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
SELECT COUNT(DISTINCT StockCode) AS unique_stock_count
FROM capable-sled-456102-t0.modulabs_project.data;
```

행	unique_stock_count	
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 capable-sled-456102-t0.modulabs_project.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 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 capable-sled-456102-t0.modulabs_project.data
)
WHERE number_count <= 1
ORDER BY number_count, StockCode;
```

행	StockCode ▼	number_count ▼
1	BANK CHARGES	0
2	CRUK	0
3	D	0
4	DOT	0
5	M	0
6	PADS	0
7	POST	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 퍼센트인지 구하기 (소수점 두 번째 자리까지)

```
WITH StockNumberCount AS (
  SELECT *,
```

```

        LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode,
r'[0-9]', '')) AS number_count
    FROM capable-sled-456102-t0.modulabs_project.data
),
StockNumberFilter AS (
    SELECT *
    FROM StockNumberCount
    WHERE number_count <= 1
)
SELECT ROUND(COUNT(*) /
    (SELECT COUNT(*)
        FROM capable-sled-456102-t0.modulabs_project.data) * 100, 2) A
S percent_of_total
FROM StockNumberFilter;

```

행	percent_of_total ▼	
1	0.48	

- 제품과 관련되지 않은 거래 기록을 제거하기

```

DELETE FROM capable-sled-456102-t0.modulabs_project.data
WHERE StockCode IN (
    SELECT StockCode
    FROM (
        SELECT DISTINCT StockCode,
            LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode,
r'[0-9]', '')) AS number_count
        FROM capable-sled-456102-t0.modulabs_project.data
    )
    WHERE number_count <= 1
);

```



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 capable-sled-456102-t0.modulabs_project.data
GROUP BY Description
ORDER BY description_cnt DESC
LIMIT 30;
```

행	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
11	PAPER CHAIN KIT 50'S CHRISTMAS	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BLUE	1000
14	HEART OF WICKER SMALL	990
15	NATURAL SLATE HEART CHALKBOARD	989
16	JAM MAKING SET WITH JARS	966
17	LUNCH BAG PINK POI KADOT	961


- 서비스 관련 정보를 포함하는 행들을 제거하기

```
DELETE FROM capable-sled-456102-t0.modulabs_project.data
WHERE Description IN (
  'Next Day Carriage',
  'High Resolution Image'
);
```

 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE capable-sled-456102-t0.modulabs_project.d
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM capable-sled-456102-t0.modulabs_project.data
```

 이 문으로 이름이 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 capable-sled-456102-t0.modulabs_project.data;
```

행	min_price ▼	max_price ▼	avg_price ▼
1	0.0	649.5	2.90495675740608

- 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 capable-sled-456102-t0.modulabs_project.data  
WHERE UnitPrice = 0;
```

행	cnt_quantity ▼	min_quantity ▼	max_quantity ▼	avg_quantity ▼
1	33	1	12540	420.51515151515139

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE capable-sled-456102-t0.modulabs_project.d  
SELECT *
```

```
FROM capable-sled-456102-t0.modulabs_project.data
WHERE UnitPrice > 0;
```



이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- **InvoiceDate** 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay,
*
FROM capable-sled-456102-t0.modulabs_project.data;
```

행	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR
2	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STORAGE JAR
3	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	BOX OF 6 ASSORTED COLOUR TEASPOONS
4	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNOB ACRYLIC EDWARDIAN
5	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC	5.49	12347	Iceland	CAMOUFLAGE EAR MUFF HEADPHONES
6	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	CLEAR DRAWER KNOB ACRYLIC EDWARDIAN
7	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PIECE POLKADOT CUTLERY SET
8	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	PINK 3 PIECE POLKADOT CUTLERY SET
9	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLACK GRAND BAROQUE PHOTO FRAME
10	2010-12-07	537626	22725	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE CHOCOLATE
11	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	SET OF 2 TINS VINTAGE BATHROOM
12	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE RED
13	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PLAYING CARDS
14	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	RED DRAWER KNOB ACRYLIC EDWARDIAN
15	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland	FOUR HOOK WHITE LOVEBIRDS
16	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland	MINI PAINT SET VINTAGE
17	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB ACRYLIC EDWARDIAN
18	2010-12-07	537626	22195	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	LARGE HEART MEASURING SPOONS
19	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE PINK
20	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PURPLE DRAWER KNOB ACRYLIC EDWARDIAN

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT
  (SELECT MAX(InvoiceDate))
  FROM `capable-sled-456102-t0.modulabs_project.data`) AS most_rec
ent_date,
  DATE(InvoiceDate) AS InvoiceDay,
  *
FROM capable-sled-456102-t0.modulabs_project.data;
```

행	most_recent_date	InvoiceDay	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1	2011-12-09	2011-01-18	541431	23166	74215	2011-01-18 10:01:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STOR...
2	2011-12-09	2011-01-18	C541433	23166	-74215	2011-01-18 10:17:00 UTC	1.04	12346	United Kingdom	MEDIUM CERAMIC TOP STOR...
3	2011-12-09	2010-12-07	537626	84969	6	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	BOX OF 6 ASSORTED COLOUR ...
4	2011-12-09	2010-12-07	537626	22773	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	GREEN DRAWER KNOB ACRYLI...
5	2011-12-09	2010-12-07	537626	20782	6	2010-12-07 14:57:00 UTC	5.49	12347	Iceland	CAMOUFLAGE EAR MUFF HEA...
6	2011-12-09	2010-12-07	537626	22771	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	CLEAR DRAWER KNOB ACRYLI...
7	2011-12-09	2010-12-07	537626	84997C	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	BLUE 3 PIECE POLKADOT CUT...
8	2011-12-09	2010-12-07	537626	84997D	6	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	PINK 3 PIECE POLKADOT CUT...
9	2011-12-09	2010-12-07	537626	85167B	30	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLACK GRAND BAROQUE PHO...
10	2011-12-09	2010-12-07	537626	22725	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE CHO...
11	2011-12-09	2010-12-07	537626	22497	4	2010-12-07 14:57:00 UTC	4.25	12347	Iceland	SET OF 2 TINS VINTAGE BATH...
12	2011-12-09	2010-12-07	537626	22727	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE RED
13	2011-12-09	2010-12-07	537626	84558A	24	2010-12-07 14:57:00 UTC	2.95	12347	Iceland	3D DOG PICTURE PLAYING CA...
14	2011-12-09	2010-12-07	537626	22774	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	RED DRAWER KNOB ACRYLIC ...
15	2011-12-09	2010-12-07	537626	22212	6	2010-12-07 14:57:00 UTC	2.1	12347	Iceland	FOUR HOOK WHITE LOVEBIRDS
16	2011-12-09	2010-12-07	537626	22492	36	2010-12-07 14:57:00 UTC	0.65	12347	Iceland	MINI PAINT SET VINTAGE
17	2011-12-09	2010-12-07	537626	22805	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	BLUE DRAWER KNOB ACRYLIC...
18	2011-12-09	2010-12-07	537626	22195	12	2010-12-07 14:57:00 UTC	1.65	12347	Iceland	LARGE HEART MEASURING SP...
19	2011-12-09	2010-12-07	537626	22728	4	2010-12-07 14:57:00 UTC	3.75	12347	Iceland	ALARM CLOCK BAKELIKE PINK
20	2011-12-09	2010-12-07	537626	22775	12	2010-12-07 14:57:00 UTC	1.25	12347	Iceland	PURPLE DRAWER KNOB ACRYL...

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT CustomerID,
  MAX(InvoiceDate) AS InvoiceDay
FROM capable-sled-456102-t0.modulabs_project.data
GROUP BY CustomerID;
```

행	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	
11	12357	2011-11-06	
12	12358	2011-12-08	
13	12359	2011-12-02	
14	12360	2011-10-18	
15	12361	2011-02-25	
16	12362	2011-12-06	
17	12363	2011-08-22	

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

```

SELECT CustomerID,
      -- EXTRACT 함수: 날짜 또는 시간 데이터 타입에서 특정 부분을 추출
      EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS rece
FROM (
  SELECT CustomerID,
        MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM capable-sled-456102-t0.modulabs_project.data
  GROUP BY CustomerID
);

```

행	CustomerID ▼	recency ▼
1	12843	65
2	12882	9
3	12921	3
4	12947	143
5	12949	30
6	13279	64
7	13297	7
8	13314	1
9	13408	1
10	13593	108
11	13667	149
12	13695	31
13	13833	159
14	14047	8
15	14125	10
16	14138	1
17	14426	18

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

```
CREATE OR REPLACE TABLE `capable-sled-456102-t0.modulabs_project.modulabs_project`
SELECT CustomerID,
       EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT CustomerID,
         MAX(InvoiceDate) AS InvoiceDay
  FROM capable-sled-456102-t0.modulabs_project.data
  GROUP BY CustomerID
);
```



이 문으로 이름이 user_r인 새 테이블이 생성되었습니다.

행	CustomerID	recency	
1	17490	0	
2	12662	0	
3	16558	0	
4	13113	0	
5	12423	0	
6	16626	0	
7	17364	0	
8	12526	0	
9	12433	0	
10	14397	0	
11	15311	0	
12	15344	0	
13	16954	0	
14	15694	0	
15	12710	0	

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT CustomerID,
       COUNT(DISTINCT InvoiceNo) AS purchase_cnt
FROM capable-sled-456102-t0.modulabs_project.data
GROUP BY CustomerID;
```

행	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
11	12357	1
12	12358	2
13	12359	6
14	12360	3
15	12361	1

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT CustomerID,
       SUM(Quantity) AS item_cnt
FROM capable-sled-456102-t0.modulabs_project.data
GROUP BY CustomerID;
```

행	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	
9	12355	240	
10	12356	1573	
11	12357	2708	
12	12358	242	
13	12359	1599	
14	12360	1156	
15	12361	90	

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

```
CREATE OR REPLACE TABLE `capable-sled-456102-t0.modulabs_project.

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT CustomerID,
         COUNT(DISTINCT InvoiceNo) AS purchase_cnt
  FROM capable-sled-456102-t0.modulabs_project.data
  GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT CustomerID,
```

```

        SUM(Quantity) AS item_cnt
    FROM capable-sled-456102-t0.modulabs_project.data
    GROUP BY CustomerID
)
-- 기존 user_r 테이블과 병합
SELECT pc.CustomerID,
       pc.purchase_cnt,
       ic.item_cnt,
       ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
    USING(CustomerID)
JOIN capable-sled-456102-t0.modulabs_project.user_r AS ur
    USING(CustomerID);

```



이 문으로 이름이 user_rf인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency
1	12713	1	505	0
2	14569	1	79	1
3	15520	1	314	1
4	13298	1	96	1
5	13436	1	76	1
6	15195	1	1404	2
7	15471	1	256	2
8	14204	1	72	2
9	14578	1	240	3
10	15992	1	17	3
11	17914	1	457	3
12	12478	1	233	3
13	15318	1	642	3
14	12442	1	181	3
15	16528	1	171	3

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT CustomerID,  
       ROUND(SUM(Quantity * UnitPrice), 1) AS user_total  
FROM capable-sled-456102-t0.modulabs_project.data  
GROUP BY CustomerID;
```

행	CustomerID	user_total
1	12346	0.0
2	12347	4310.0
3	12348	1437.2
4	12349	1457.5
5	12350	294.4
6	12352	1265.4
7	12353	89.0
8	12354	1079.4
9	12355	459.4
10	12356	2487.4
11	12357	6207.7
12	12358	928.1
13	12359	6183.0
14	12360	2302.1
15	12361	174.9

- 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 `capable-sled-456102-t0.modulabs_project.i
SELECT rf.CustomerID AS CustomerID,
       rf.purchase_cnt,
       rf.item_cnt,
       rf.recency,
       ut.user_total,
       ROUND(ut.user_total / rf.purchase_cnt, 1) AS user_average
FROM capable-sled-456102-t0.modulabs_project.user_rf AS rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT CustomerID,
         ROUND(SUM(Quantity * UnitPrice), 1) AS user_total
  FROM capable-sled-456102-t0.modulabs_project.data
  GROUP BY CustomerID
) AS ut
  USING(CustomerID);

```

i 이 문으로 이름이 user_rfm인 새 테이블이 생성되었습니다.

RFM 통합 테이블 출력하기

- 최종 user_rfm 테이블을 출력하기

```

SELECT *
FROM capable-sled-456102-t0.modulabs_project.user_rfm;

```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1	12713	1	505	0	794.6	794.6
2	15520	1	314	1	343.5	343.5
3	13436	1	76	1	196.9	196.9
4	14569	1	79	1	227.4	227.4
5	13298	1	96	1	360.0	360.0
6	15471	1	256	2	454.5	454.5
7	15195	1	1404	2	3861.0	3861.0
8	14204	1	72	2	150.6	150.6
9	12478	1	233	3	546.0	546.0
10	15992	1	17	3	42.0	42.0
11	16528	1	171	3	244.4	244.4
12	12650	1	250	3	242.4	242.4
13	15318	1	642	3	312.6	312.6
14	14578	1	240	3	168.6	168.6
15	12442	1	181	3	144.1	144.1

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

```
-- (3) user_data 테이블로 저장
```

```
CREATE OR REPLACE TABLE capable-sled-456102-t0.modulabs_project.user_
WITH unique_products AS (
```

```
-- (1) 고객별 구매한 고유 제품 수 계산
```

```
-- -> 고객이 얼마나 다양한 제품을 구매했는가?를 알려주는 지표
```

```
SELECT CustomerID,
       COUNT(DISTINCT StockCode) AS unique_products
```

```

FROM capable-sled-456102-t0.modulabs_project.data
GROUP BY CustomerID
)
-- (2) 기존 user_rfm 테이블과 JOIN
SELECT ur.*,          -- user_rfm 테이블의 모든 컬럼
       up.* EXCEPT (CustomerID) -- unique_products 테이블의 CustomerID 제외
FROM capable-sled-456102-t0.modulabs_project.user_rfm AS ur
JOIN unique_products AS up
  ON ur.CustomerID = up.CustomerID;

```

i 이 문으로 이름이 user_data인 새 테이블이 생성되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products
1	16093	1	20	106	17.0	17.0	1
2	15313	1	25	110	52.0	52.0	1
3	15316	1	100	326	165.0	165.0	1
4	13841	1	100	252	85.0	85.0	1
5	15118	1	1440	134	244.8	244.8	1
6	16738	1	3	297	3.8	3.8	1
7	18184	1	60	15	49.8	49.8	1
8	16765	1	4	294	34.0	34.0	1
9	16737	1	288	53	417.6	417.6	1
10	17763	1	12	263	15.0	15.0	1
11	15488	1	72	92	76.3	76.3	1
12	17331	1	16	123	175.2	175.2	1
13	13099	1	288	99	207.4	207.4	1
14	18068	1	6	289	101.7	101.7	1
15	16138	1	-1	368	-8.0	-8.0	1

2. 평균 구매 주기

- 고객들의 쇼핑 패턴을 이해하는 것을 목표 (고객 별 재방문 주기 살펴보기)
 - 군 구매 소요 일수를 계산하고, 그 결과를 **user_data**에 통합

```

CREATE OR REPLACE TABLE `capable-sled-456102-t0.modulabs_project.user
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)) AS interval_
    FROM capable-sled-456102-t0.modulabs_project.data
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)
SELECT u.*, pi.* EXCEPT (CustomerID)
FROM capable-sled-456102-t0.modulabs_project.user_data AS u
LEFT JOIN purchase_intervals AS pi
  ON u.CustomerID = pi.CustomerID;

```



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval
1	15940	1	4	311	35.8	35.8	1	0.0
2	13366	1	144	50	56.2	56.2	1	0.0
3	16078	1	16	283	79.2	79.2	1	0.0
4	15657	1	24	22	30.0	30.0	1	0.0
5	16738	1	3	297	3.8	3.8	1	0.0
6	18068	1	6	289	101.7	101.7	1	0.0
7	15389	1	400	172	500.0	500.0	1	0.0
8	16995	1	-1	372	-1.3	-1.3	1	0.0
9	17382	1	24	65	50.4	50.4	1	0.0
10	18113	1	72	368	76.3	76.3	1	0.0
11	18174	1	50	7	104.0	104.0	1	0.0
12	16148	1	72	296	76.3	76.3	1	0.0
13	17923	1	50	282	207.5	207.5	1	0.0
14	13703	1	10	318	99.5	99.5	1	0.0
15	15524	1	4	24	440.0	440.0	1	0.0

3. 구매 취소 경향성

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

```
CREATE OR REPLACE TABLE capable-sled-456102-t0.modulabs_project.user_data
WITH TransactionInfo AS (
  SELECT CustomerID,
    -- 고객별 전체 거래 수 계산
    COUNT(InvoiceNo) AS total_transactions,
    -- 고객별 취소 거래 수 계산
    -- STARTS_WITH(): 정확하게 "시작하는지 여부"만 판단하는 함수
    -- LIKE 'C%'를 사용하면 인덱스를 못쓰고 성능상 더 무거움
    COUNTIF(STARTS_WITH(InvoiceNo, 'C')) AS cancel_frequency
  FROM capable-sled-456102-t0.modulabs_project.data
  WHERE CustomerID IS NOT NULL
  GROUP BY CustomerID
)

SELECT u.*,
  t.* EXCEPT(CustomerID),
  -- 취소 비율은 소수점 두번째 자리까지
  ROUND(cancel_frequency / total_transactions, 2) AS cancel_rate
FROM capable-sled-456102-t0.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
  ON u.CustomerID = t.CustomerID;
```



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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

```
SELECT *
FROM capable-sled-456102-t0.modulabs_project.user_data
```

행	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	16454	1	2	64	5.9	5.9	1	0.0	1	0	0.0
2	14576	1	12	372	35.4	35.4	1	0.0	1	0	0.0
3	16148	1	72	296	76.3	76.3	1	0.0	1	0	0.0
4	13841	1	100	252	85.0	85.0	1	0.0	1	0	0.0
5	13302	1	5	155	63.8	63.8	1	0.0	1	0	0.0
6	16738	1	3	297	3.8	3.8	1	0.0	1	0	0.0
7	15524	1	4	24	440.0	440.0	1	0.0	1	0	0.0
8	13120	1	12	238	30.6	30.6	1	0.0	1	0	0.0
9	14679	1	-1	371	-2.5	-2.5	1	0.0	1	1	1.0
10	16995	1	-1	372	-1.3	-1.3	1	0.0	1	1	1.0
11	15118	1	1440	134	244.8	244.8	1	0.0	1	0	0.0
12	12791	1	96	373	177.6	177.6	1	0.0	1	0	0.0
13	13099	1	288	99	207.4	207.4	1	0.0	1	0	0.0
14	16737	1	288	53	417.6	417.6	1	0.0	1	0	0.0
15	16990	1	100	218	179.0	179.0	1	0.0	1	0	0.0
16	14351	1	12	164	51.0	51.0	1	0.0	1	0	0.0
17	13366	1	144	50	56.2	56.2	1	0.0	1	0	0.0
18	16765	1	4	294	34.0	34.0	1	0.0	1	0	0.0
19	18133	1	1350	212	931.5	931.5	1	0.0	1	0	0.0
20	17752	1	192	359	80.6	80.6	1	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep :

캐글에 공개된 E-Commerce 데이터를 직접 BigQuery에 업로드하고, SQL로 데이터 전처리, RFM 지표 분석, Feature Engineering, 이어지는 실습을 위한 Python 연동까지 해보았다. 최종적으로 분석된 user_data 테이블을 주피터 노트북으로 불러오고 GitHub에 공유하면서 실제로 업무에서 분석 결과를 전달하는 과정까지 생각해 볼 수 있었다.

아직 기초 단계이겠지만, 데이터를 다룰 때의 End-to-End 분석 파이프라인 흐름을 직접 경험해 볼 수 있어서 굉장히 흥미로웠다.

데이터를 전처리할 때,

행과 열 수 등의 컬럼 구조를 파악하고 데이터 타입을 확인하는 과정이 꼭 필요한 과정이라는 것을 이후에 데이터로 여러 지표를 구해보면서 체감할 수 있었다.

데이터 전처리 과정은 이론으로만 알고 있었는데, SQL로 직접 데이터를 다뤄보면서 결측치와 이상치를 제거할 때 고려해야 할 점을 생각해볼 수 있었다. 누락된 값의 비율이 크다면 다른 값으로 대체하는 것은 분석에 상당한 편향을 주거나 노이즈가 될 수 있다는 사실이 흥미로웠다.

또, 쿼리를 작성하고 실행한 후 내가 작성한 쿼리가 어떤 순서로 실행되고 동작하는지도 꼼꼼하게 이해하고 넘어가려고 했다. 처음 보는 문법은 어떨 때 사용되는지 비교하며 정리해보았다. 기억에 남는 것 중 하나인데 고객별 취소 거래 수 계산을 계산할 때 기존에 배웠던 LIKE를 사용해도 되지만, STARTS_WITH() 함수를 사용하면 성능상 더 가볍고, 나중에 인덱스를 사용할 때 확장할 수 있다는 사실도 알게 되었다.

추가 Feature 추출하는 과정에서도,

하나의 데이터 셋을 가지고 이렇게 다양한 의미를 도출해낼 수 있다는 사실이 되게 놀라웠다. 생각했던 것보다 더 논리적인 과정을 통해 데이터가 분석된다는 것을 느꼈다.

궁극적으로 고객 세그멘테이션을 하기 위한 과정임을 강조하면서, 각 단계마다 가장 합리적인 접근 방법을 짚어주어서 시야를 넓히는 데에 도움이 되었다. 실습 가이드가 없었다면 과연 분석의 방향을 제대로 잡을 수 있었을지는 모르겠다 ㅎㅎ

Problem :

RFM 지표, 제품 다양성, 평균적인 재구매 주기, 취소 비율 등을 계산하는 것에 몰두하다 보니 데이터를 분석하는 것의 본질을 잊고, 정확한 실행 결과를 위한 SQL 쿼리문 작성에만 집중하고 있었다.

이전 노트에서도 강조해 주셨던 부분이지만

데이터를 단순히 분석하는 것이 아니라, 비즈니스적 의미를 고려한 지표 설계를 통해 행동을 다양한 각도로 이해하려는 노력을 잊지 않아야겠다는 반성을 하게 되었다.

"지표가 왜 이렇게 나왔을까?" 라는 질문을 계속 던져보아야겠다.

Try :

처음으로 캐글 데이터를 분석해 보고, 실제 이커머스 데이터를 기반으로 분석을 수행했다는 점에서 매우 의미 있었다. 데이터를 다루는 과정에서 생길 수 있는 여러 문제들을 마주할 수 있었다. 결측치, 이상치, 데이터 표준화, 도메인 이해 등)를 실제로 마주할 수 있었다.

오늘 프로젝트를 기회로 앞으로 다양한 도메인의 캐글 데이터셋을 분석해 보면서 도메인별로 어떤 지표가 중요한지도 비교해보고 싶다.

또 그렇게 분석 프로젝트를 진행하면서 그 과정과 결과를 정리할 때, README와 블로그에 어떤 내용을 담고 어떻게 기록해 나가면 좋을지도 생각해 보아야겠다.