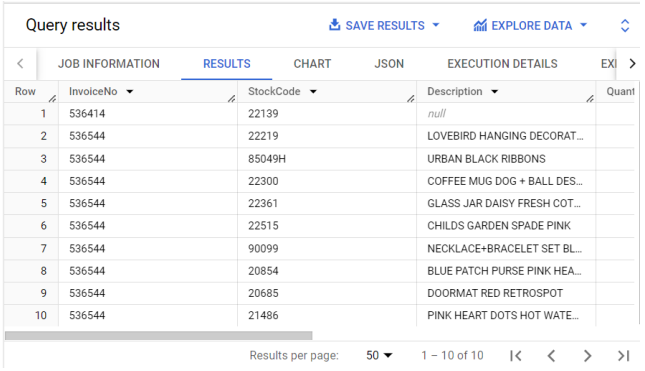
고객을 세그먼테이션하자 [프로젝트]

11-2. 데이터 불러오기

데이터 살펴보기

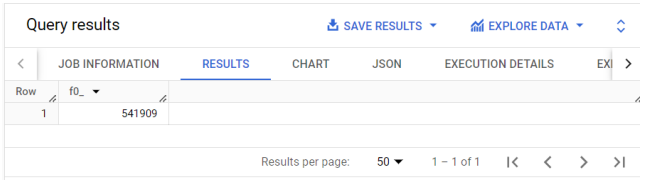
- **테이블에 있는 10개의 행만 출력하기**

```
-- 10개 행 출력
SELECT *
FROM uplifted-sphinx-439401-k5.modulabs_project.data
LIMIT 10;
```



- **전체 데이터는 몇 행으로 구성되어 있는지 확인하기**

```
-- 전체 행의 개수 출력
SELECT count(*)
FROM uplifted-sphinx-439401-k5.modulabs_project.data;
```

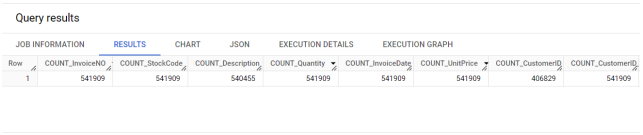


데이터 수 세기

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

```
BigQuery에서는 각 컬럼별로 데이터 포인트의 수를 세는 것이 가능하지만, 모든 컬럼을 자동으로 한 번에 계산하는 기능은 없고, 각각을 수동으로 지정해야 한다.

-- 데이터 수 세기
SELECT count(InvoiceNo) AS COUNT_InvoiceNO,
      count(StockCode) AS COUNT_StockCode,
      count(Description) AS COUNT_Description, -- 결측치 존재
      count(Quantity) AS COUNT_Quantity,
      count(InvoiceDate) AS COUNT_InvoiceDate,
      count(UnitPrice) AS COUNT_UnitPrice,
      count(CustomerID) AS COUNT_CustomerID,  -- 결측치 존재
```

```
      count(Country) AS COUNT_CustomerID
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```


Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

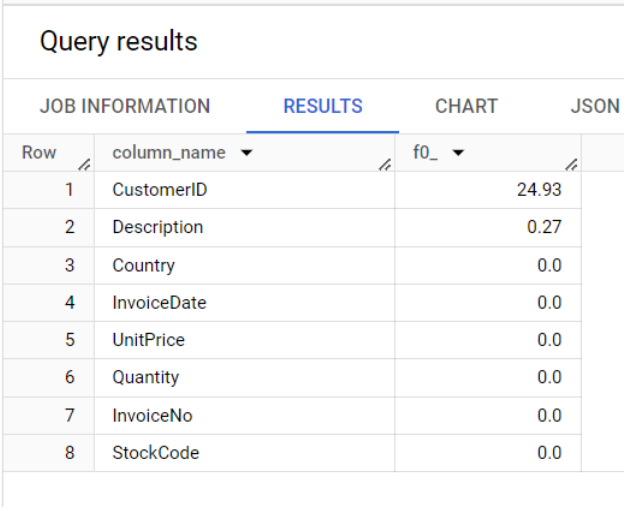
Row	COUNT_InvoiceNO	COUNT_StockCode	COUNT_Description	COUNT_Quantity	COUNT_InvoiceDate	COUNT_UnitPrice	COUNT_CustomerID	COUNT_CustomerID
1	541909	541909	540455	541909	541909	541909	406829	541909

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**

```
--  컬럼 별 누락된 값의 비율 계산
SELECT column_name, ROUND((total - column_value) / total * 100, 2)
FROM
(
    SELECT 'InvoiceNo' AS column_name, COUNT(InvoiceNo) AS column_value, COUNT(*) AS total FROM `u
plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'StockCode' AS column_name, COUNT(StockCode) AS column_value, COUNT(*) AS total FROM `u
plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'Description' AS column_name, COUNT(Description) AS column_value, COUNT(*) AS total FRO
M `uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'Quantity' AS column_name, COUNT(Quantity) AS column_value, COUNT(*) AS total FROM `upl
ifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'InvoiceDate' AS column_name, COUNT(InvoiceDate) AS column_value, COUNT(*) AS total FRO
M `uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'UnitPrice' AS column_name, COUNT(UnitPrice) AS column_value, COUNT(*) AS total FROM `u
plifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'CustomerID' AS column_name, COUNT(CustomerID) AS column_value, COUNT(*) AS total FROM
`uplifted-sphinx-439401-k5.modulabs_project.data` UNION ALL
    SELECT 'Country' AS column_name, COUNT(Country) AS column_value, COUNT(*) AS total FROM `uplif
ted-sphinx-439401-k5.modulabs_project.data`
) AS column_data
ORDER BY 2 DESC;
```


Query results

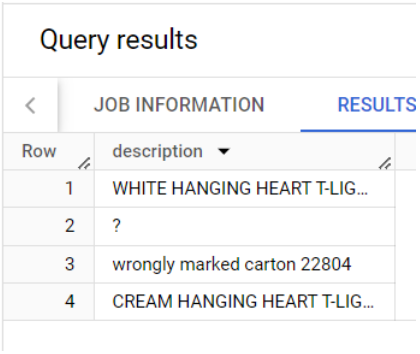
JOB INFORMATION | RESULTS | CHART | JSON

Row	column_name	f0_
1	CustomerID	24.93
2	Description	0.27
3	Country	0.0
4	InvoiceDate	0.0
5	UnitPrice	0.0
6	Quantity	0.0
7	InvoiceNo	0.0
8	StockCode	0.0

결측치 처리 전략

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

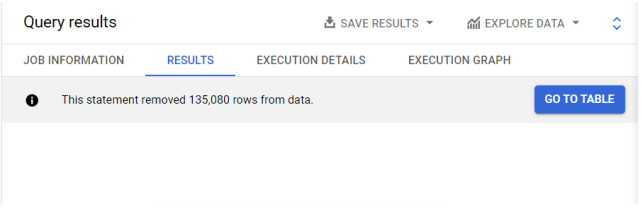
```
-- StockCode = '85123A'을 만족하는 Description 컬럼
SELECT DISTINCT description
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE StockCode = '85123A';
```



결측치 처리

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

```
-- 결측치 데이터 삭제
DELETE FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE description IS NULL
OR customerID IS NULL
```



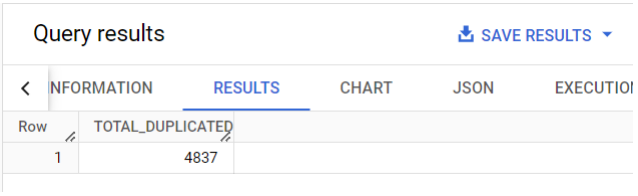
11-5. 데이터 전처리(2): 중복값 처리

중복값 확인

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

```
-- 중복된 행의 수 계산
WITH CHECK_DUPLICATED_ROWS AS (
SELECT *, count(*) AS DUPLICATED_ROWS
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY InvoiceNo,
          StockCode,
          Description,
          Quantity,
          InvoiceDate,
          UnitPrice,
```

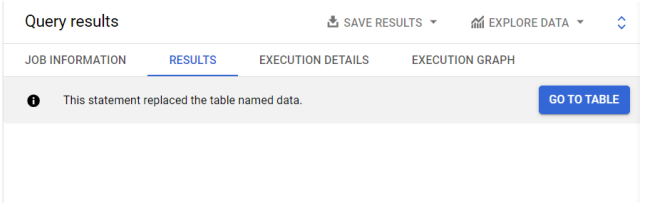
```
          CustomerID,
          Country
HAVING DUPLICATED_ROWS > 1
)
SELECT count(DUPLICATED_ROWS) AS TOTAL_DUPLICATED
FROM CHECK_DUPLICATED_ROWS;
```



중복값 처리

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

```
-- 중복된 행들을 제거하여 테이블을 대체
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT DISTINCT InvoiceNo,
                StockCode,
                Description,
                Quantity,
                InvoiceDate,
                UnitPrice,
                CustomerID,
                Country
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```

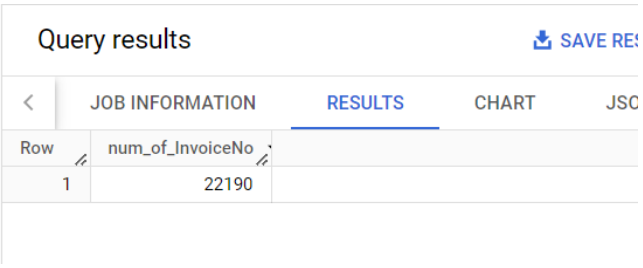


11-6. 데이터 전처리(3): 오류값 처리

`InvoiceNo` 살펴보기

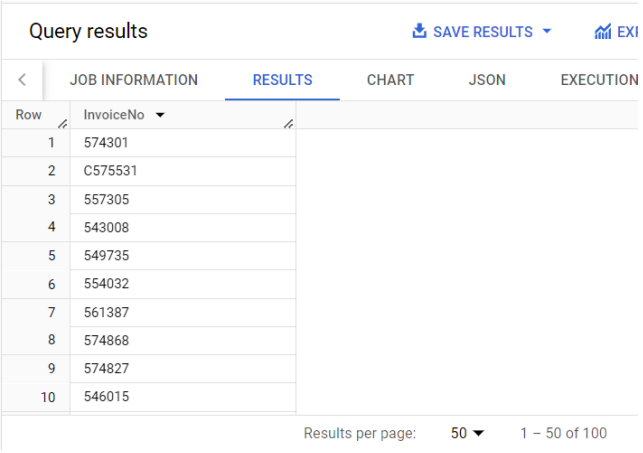
- **고유(unique)한 `InvoiceNo`의 개수를 출력하기**

```
--  고유(unique)한 InvoiceNo의 개수를 출력
SELECT count(DISTINCT InvoiceNo) AS num_of_InvoiceNo
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```



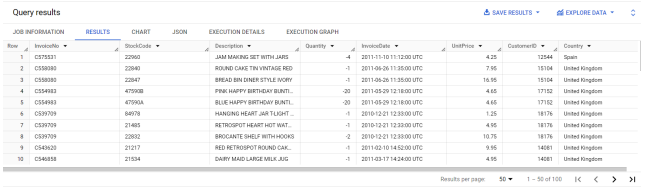
- **고유한 `InvoiceNo` 를 앞에서부터 100개를 출력하기**

```
--  고유한 InvoiceNo를 앞에서부터 100개를 출력
SELECT DISTINCT InvoiceNo
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
LIMIT 100;
```



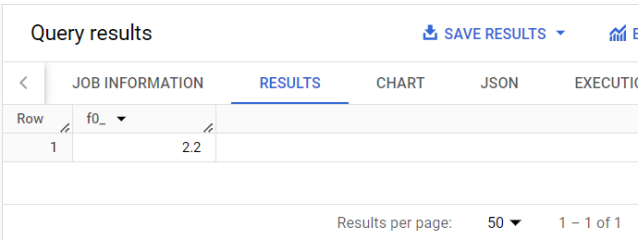
- **`InvoiceNo` 가 'C'로 시작하는 행을 필터링 할 수 있는 쿼리문을 작성하기 (100행까지만 출력)**

```
--  InvoiceNo가 'C'로 시작하는 행을 필터링(100개)
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE InvoiceNo LIKE 'C%'
LIMIT 100;
```



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

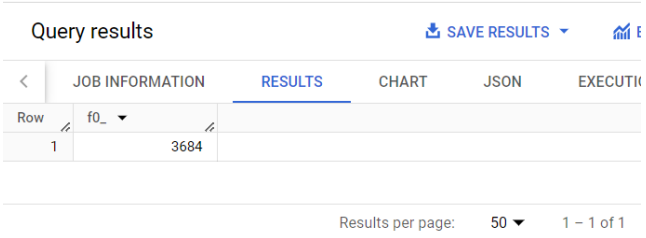
```
--  구매 건 상태가 Canceled 인 데이터의 비율(%)
SELECT ROUND(SUM(CASE WHEN InvoiceNo LIKE 'C%' THEN 1 ELSE 0 END) / COUNT(*) *100, 1)
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```



`StockCode` 살펴보기

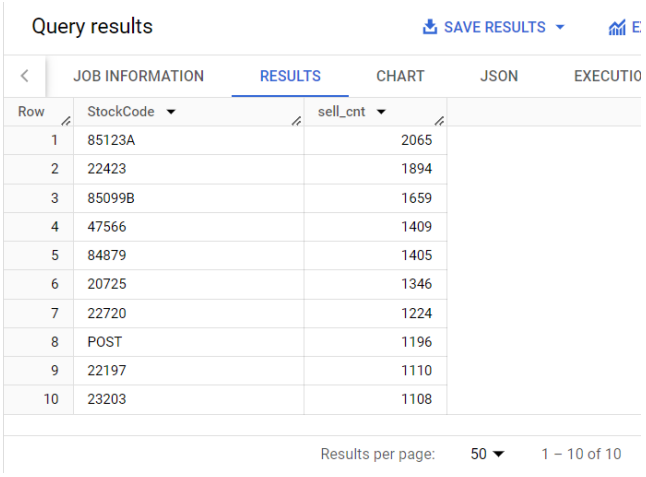
- **고유한 `StockCode`의 개수를 출력하기**

```
--  고유한 StockCode의 개수를 출력
SELECT count(DISTINCT StockCode)
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```



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

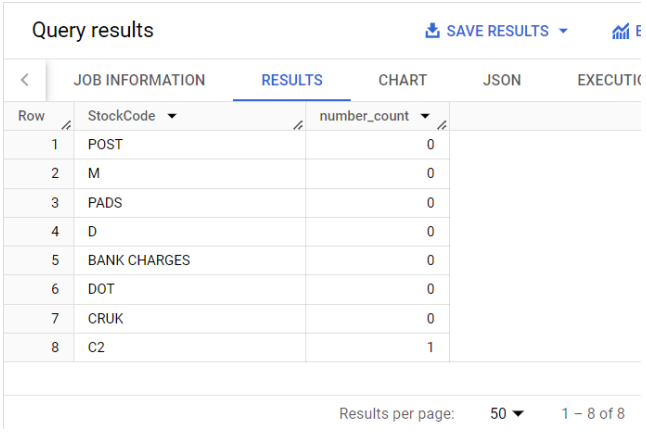
```
--  StockCode 별 등장 빈도를 출력(상위10개)
SELECT StockCode, COUNT(*) AS sell_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY StockCode
ORDER BY sell_cnt DESC
LIMIT 10;
```



- **`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

```
--  숫자가 0~1개인 값들에는 어떤 코드들이 들어가 있는지를 확인
SELECT DISTINCT StockCode, number_count
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
)
WHERE number_count = 0
OR number_count = 1;
```



Query results

SAVE RESULTS

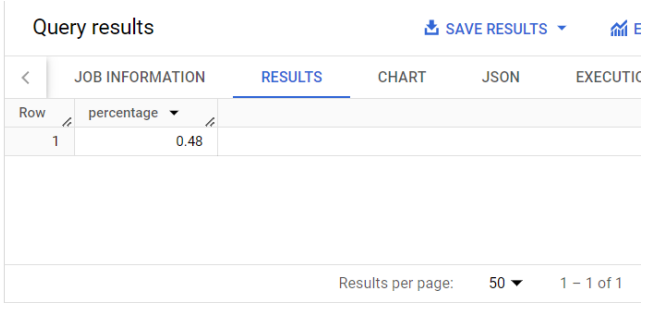
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	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

Results per page: 50 1 – 8 of 8

- `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)

```
--  해당 코드 값들을 가지고 있는 데이터 수가 전체 데이터 에서 차지하는 퍼센트
SELECT
  ROUND(SUM(CASE WHEN number_count = 0 OR number_count = 1 THEN 1 ELSE 0 END)/COUNT(*)*100, 2) AS
percentage
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
);
```



Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

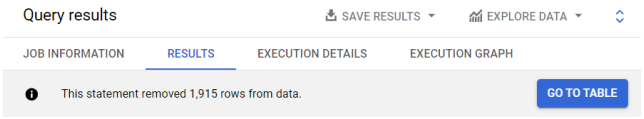
Row	percentage
1	0.48

Results per page: 50 1 – 1 of 1

- **제품과 관련되지 않은 거래 기록을 제거하기**

```
--  제품과 관련되지 않은 거래 기록을 제거
DELETE FROM `uplifted-sphinx-439401-k5.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 `uplifted-sphinx-439401-k5.modulabs_project.data`)
  WHERE number_count = 0 OR number_count = 1
);
```



`Description` 살펴보기

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

```
--  고유한 Description 별 출현 빈도를 계산하고 상위 30개를 출력
SELECT Description, COUNT(*) AS description_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY Description
LIMIT 30;
```

Query results

SAVE RESULTS

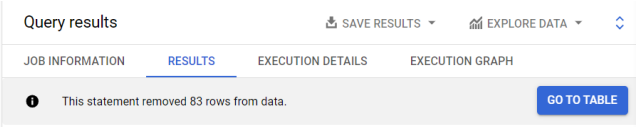
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIO

Row	Description	description_cnt
1	TRADITIONAL CHRISTMAS RIB...	272
2	PAPER CHAIN KIT 50'S CHRIST...	1013
3	CHRISTMAS CRAFT LITTLE FRI...	433
4	SCANDINAVIAN REDS RIBBONS	343
5	EMBROIDERED RIBBON REEL R...	49
6	SET OF 4 KNICK KNACK TINS ...	328
7	ASSORTED COLOUR MINI CAS...	372
8	FELTCRAFT PRINCESS LOLA D...	383
9	EMBROIDERED RIBBON REEL E...	87
10	ASSORTED COLOUR BIRD ORN...	1405
11	JAM MAKING SET WITH JARS	966
12	6 RIBBONS RUSTIC CHARM	747

Results per page: 50 1 – 30 of 30

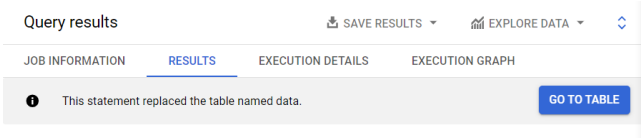
- **서비스 관련 정보를 포함하는 행들을 제거하기**

```
--  서비스 관련 정보를 포함하는 행들을 제거
-- 'Next Day Carriage'나'High Resolution Image'를 포함하는 행 제거
DELETE
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE description IN ('Next Day Carriage','High Resolution Image');
```



- **대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기**

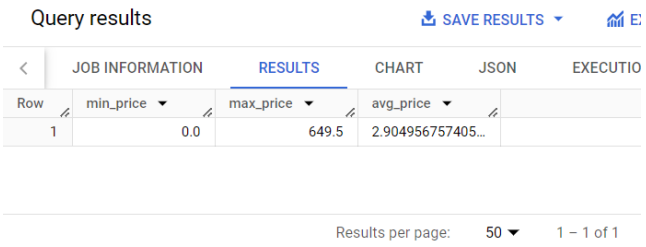
```
--  대소문자를 혼합하고 있는 데이터를 대문자로 표준화
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT * EXCEPT (Description),
        UPPER(Description) AS Description
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```



`UnitPrice` 살펴보기

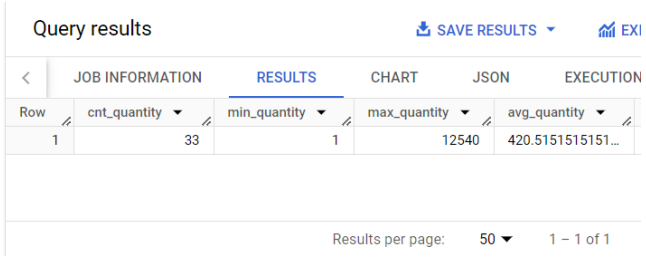
- `UnitPrice`**의 최솟값, 최댓값, 평균을 구하기**

```
--  UnitPrice의 최솟값, 최댓값, 평균
SELECT min(UnitPrice) AS min_price,
       max(UnitPrice) AS max_price,
       avg(UnitPrice) AS avg_price
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
```



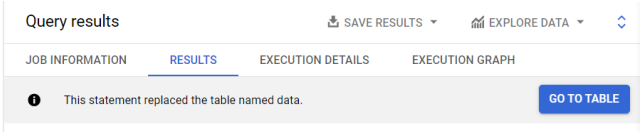
- **단가가 0원인 거래의 개수, 구매 수량(**`Quantity`**)의 최솟값, 최댓값, 평균 구하기**

```
--  단가가 0원인 거래의 개수, 구매 수량(Quantity)의 최솟값, 최댓값, 평균
SELECT count(Quantity) AS cnt_quantity,
       min(Quantity) AS min_quantity,
       max(Quantity) AS max_quantity,
       avg(Quantity) AS avg_quantity
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE UnitPrice = 0;
```



- `UnitPrice = 0` 를 제거하고 일관된 데이터셋을 유지하기

```
-- UnitPrice = 0 제거
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.data` AS
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
WHERE UnitPrice != 0;
```

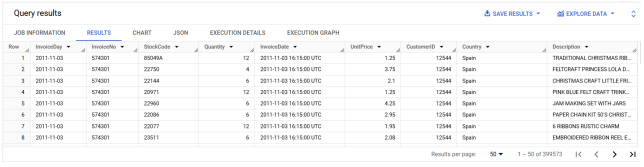


11-7. RFM 스코어

Recency

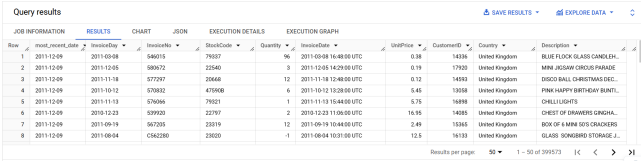
- `InvoiceDate` 컬럼을 연월일 자료형으로 변경하기

```
--  InvoiceDate 컬럼을 연월일 자료형으로 변경
SELECT DATE(InVoiceDate) AS InvoiceDay, *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```



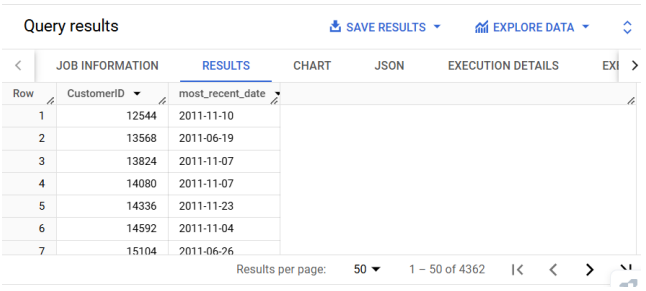
- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
--  가장 최근 구매 일자
SELECT max(DATE(InVoiceDate)) OVER () AS most_recent_date,
       DATE(InVoiceDate) AS InvoiceDay,
        *
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`;
```



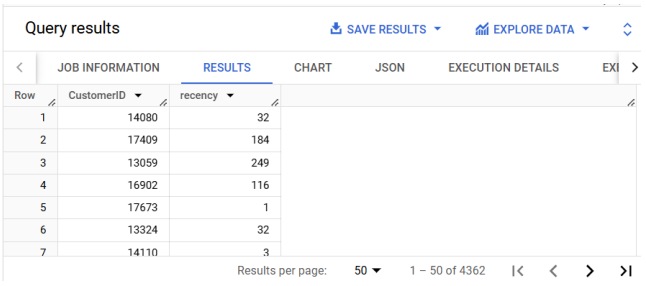
- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
--   유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장
SELECT CustomerID,  max(DATE(InVoiceDate)) AS most_recent_date
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```



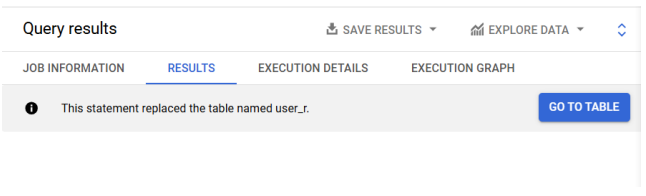
- **가장 최근 일자(`most_recent_date`)와 유저별 마지막 구매일(`InvoiceDay`)간의 차이를 계산하기**

```
-- 가장 최근 일자(most_recent_date)와 유저별 마지막 구매일(InvoiceDay)간의 차이를 계산
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
);
```



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

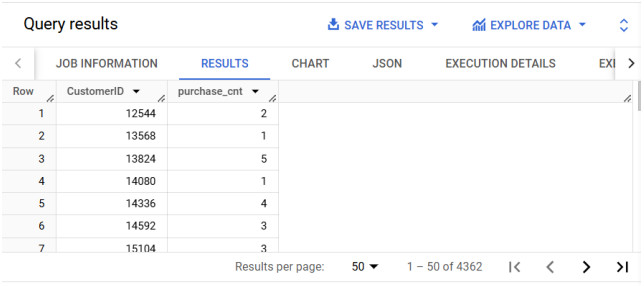
```
-- 지금까지의 결과를 user_r이라는 이름의 테이블로 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_r` AS
SELECT
  CustomerID,
  EXTRACT(DAY FROM MAX(InvoiceDay) OVER () - InvoiceDay) AS recency
FROM (
  SELECT
    CustomerID,
    MAX(DATE(InvoiceDate)) AS InvoiceDay
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
);
```



Frequency

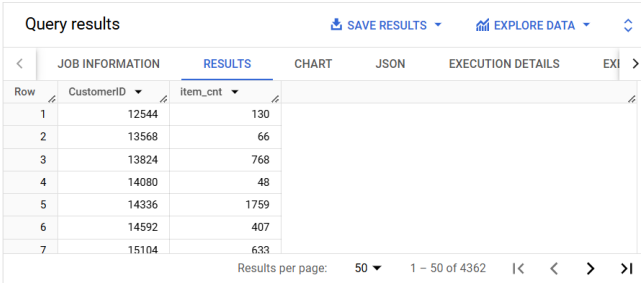
- **고객마다 고유한 InvoiceNo의 수를 세어보기**

```
-- 고객마다 고유한 InvoiceNo의 수
SELECT
  CustomerID,
  count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```



- **각 고객 별로 구매한 아이템의 총 수량 더하기**

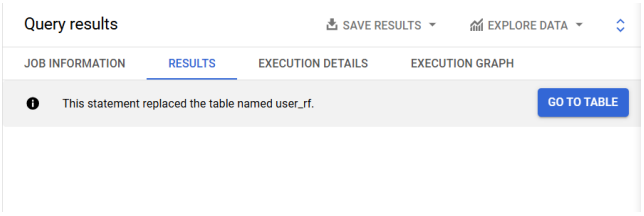
```
--  각 고객 별로 구매한 아이템의 총 수량
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```



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

```
--  1,2를 합쳐 user_rf라는 이름의 테이블에 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_rf` AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
SELECT
  CustomerID,
  count(DISTINCT InvoiceNo) AS purchase_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID
),
-- (2) 구매한 아이템 총 수량 계산
```

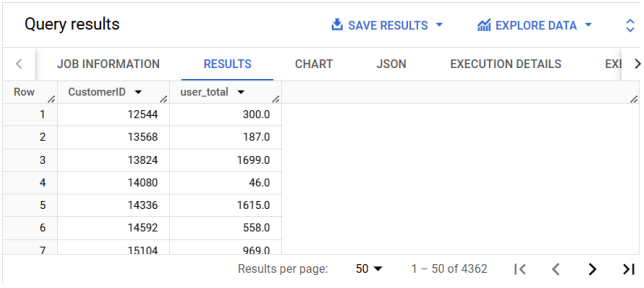
```
item_cnt AS (
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID
)
-- 기존의 user_r에 (1)과 (2)를 통합
SELECT
  pc.CustomerID,
  pc.purchase_cnt,
  ic.item_cnt,
  ur.recency
FROM purchase_cnt AS pc
JOIN item_cnt AS ic
  ON pc.CustomerID = ic.CustomerID
JOIN `uplifted-sphinx-439401-k5.modulabs_project.user_r` AS ur
  ON pc.CustomerID = ur.CustomerID;
```



Monetary

- **고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)**

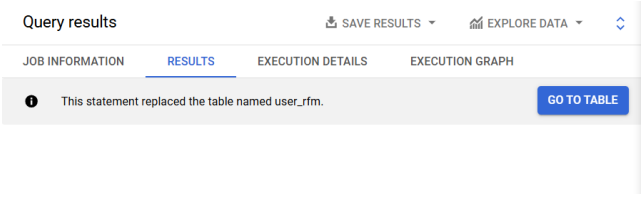
```
-- 고객별 총 지출액 계산
-- 소수점 첫째 자리에서 반올림
SELECT
  CustomerID,
  round(sum(Quantity*UnitPrice)) AS user_total
FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
GROUP BY CustomerID;
```



- **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**

```
-- user_rfm 테이블로 저장
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_rfm` AS
SELECT
```

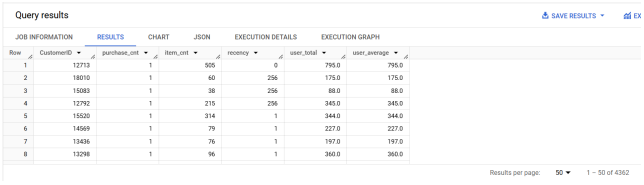
```
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.recency,
  ut.user_total,
  ut.user_total / purchase_cnt AS user_average
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rf` rf
LEFT JOIN (
  -- 고객 별 총 지출액
  SELECT
      CustomerID,
      round(sum(Quantity*UnitPrice)) AS user_total
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
) ut
ON rf.CustomerID = ut.CustomerID;
```



RFM 통합 테이블 출력하기

- **최종 user_rfm 테이블을 출력하기**

```
/*RFM 통합 테이블 출력*/
SELECT *
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rfm`
```



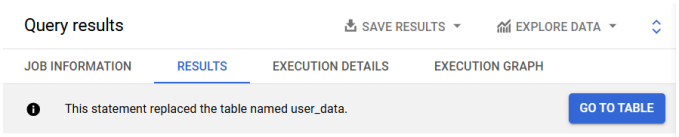
11-8. 추가 Feature 추출

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 `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS

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

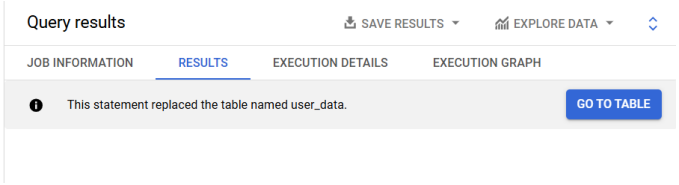
```
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
)
SELECT ur.*, up.* EXCEPT (CustomerID)
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_rfm` AS ur
JOIN unique_products AS up
ON ur.CustomerID = up.CustomerID;
```



2. 평균 구매 주기

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

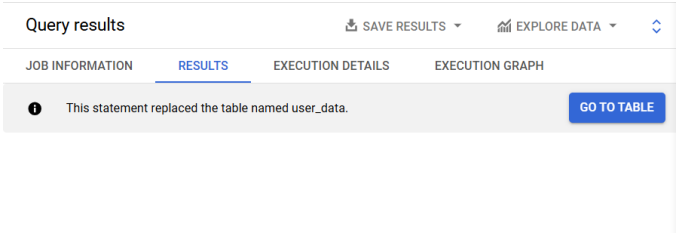
```
-- 평균 구매 소요 일수를 계산
-- 결과를 user_data에 통합
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS
WITH purchase_intervals AS (
  -- (2) 고객 별 구매와 구매 사이의 평균 소요 일수
  SELECT
    CustomerID,
    CASE WHEN ROUND(AVG(interval_), 2) IS NULL THEN 0 ELSE ROUND(AVG(interval_), 2) END AS average_in
terval
  FROM (
    -- (1) 구매와 구매 사이에 소요된 일수
    SELECT
      CustomerID,
      DATE_DIFF(InvoiceDate, LAG(InvoiceDate) OVER (PARTITION BY CustomerID ORDER BY InvoiceDate), DA
Y) AS interval_
    FROM
      `uplifted-sphinx-439401-k5.modulabs_project.data`
    WHERE CustomerID IS NOT NULL
  )
  GROUP BY CustomerID
)

SELECT u.*, pi.* EXCEPT (CustomerID)
FROM `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS u
LEFT JOIN purchase_intervals AS pi
ON u.CustomerID = pi.CustomerID;
```



3. 구매 취소 경향성

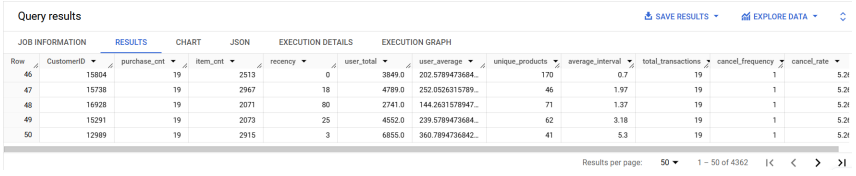
- **고객의 취소 패턴 파악하기**
  1) 취소 빈도(cancel_frequency) : 고객 별로 취소한 거래의 총 횟수
  2) 취소 비율(cancel_rate) : 각 고객이 한 모든 거래 중에서 취소를 한 거래의 비율
    - **취소 빈도와 취소 비율을 계산하고 그 결과를 `user_data`에 통합하기**
      (취소 비율은 소수점 두번째 자리)

```
-- 취소 빈도와 취소 비율을 계산
-- 결과를 user_data에 통합
-- 취소 비율은 소수점 두번째 자리까지 구하기
CREATE OR REPLACE TABLE `uplifted-sphinx-439401-k5.modulabs_project.user_data` AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(DISTINCT InvoiceNo) AS total_transactions,
    countif(DISTINCT InvoiceNo LIKE 'C%') AS cancel_frequency
  FROM `uplifted-sphinx-439401-k5.modulabs_project.data`
  GROUP BY CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round(cancel_frequency / total_transactions* 100, 2) AS cancel_ra
te
FROM `uplifted-sphinx-439401-k5.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 `uplifted-sphinx-439401-k5.modulabs_project.user_data`
```



회고

중간중간에 데이터를 잘못 추출한 경우가 생겨 되돌려야할 때가 있었습니다. 코드를 정확히 작성하고 데이터를 백업해놓는 것이 중요함을 느꼈습니다.