

고객을 세그멘테이션하자 [프로젝트] (2)

11-2. 데이터 불러오기

데이터 살펴보기

- 테이블에 있는 10개의 행만 출력하기

```
select *
from coherent-server-456101-k4.
modulabs_project.data
limit 10
```

```
1 select * from coherent-server-456101-k4.modulabs_project.data
2 limit 10
3
4
```

접근성 옵션을 보려면 Option+F1 키를 누르세요

쿼리 결과

결과 저장 다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	InvoiceNo	StockCode	Descripti
1	536365	85123A	WHITE H
2	536365	71053	WHITE M
3	536365	84406B	CREAM C
4	536365	84029G	KNITTED
5	536365	84029E	RED WO
6	536365	22752	SET 7 BA
7	536365	21730	GLASS S
8	536366	22633	HAND W.
9	536366	22632	HAND W.
10	536367	84879	ASSORTE

- 전체 데이터는 몇 행으로 구성되어 있는지 확인하기

```
select count(*) from coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

```
6 select count(*) from coherent-server-456101-k4.modulabs_project.
7 data
8 select count(InvoiceNo) as cnt_InvoiceNo, count(StockCode) as
접근성 옵션을 보려면 Option+F1 키를 누르세요
```

쿼리 결과

결과 저장 다음에서 열기

작업 정보

결과

차트

JSON

실행 세부정보

실행 그래프

행	f0_	
1	541909	

데이터 수 세기

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

```
select count(InvoiceNo) as cnt_InvoiceNo, count(StockCode) as cnt_StockCode,
count(Description) as cnt_Description, count(Quantity) as cnt_Quantity,
count(InvoiceDate) as cnt_InvoiceDate, count(UnitPrice) as cnt_UnitPrice,
count(CustomerID) as cnt_CustomerID, count(Country) as cnt_Country,
from coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

```

1 /
2 select count(InvoiceNo) as cnt_InvoiceNo, count(StockCode) as cnt_StockCode,
3 count(Description) as cnt_Description, count(Quantity) as cnt_Quantity,
4 count(InvoiceDate) as cnt_InvoiceDate, count(UnitPrice) as cnt_UnitPrice,
5 count(CustomerID) as cnt_CustomerID, count(Country) as cnt_Country,
6 from coherent-server-456101-k4.modulabs_project.data
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

cnt_InvoiceNo	cnt_StockCode	cnt_Description	cnt_Quantity	cnt_InvoiceDate	cnt_UnitPrice	cnt_CustomerID	cnt_Country
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

```

SELECT
  'InvoiceNo' AS column_name,
  ROUND(SUM(CASE WHEN InvoiceNo IS NULL THEN 1 ELSE 0 END) / COUNT(*) * 100, 2) AS missing_percentage
FROM coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.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 coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	column_name ▼	missing_percentage			
1	Country	0.0			
2	StockCode	0.0			
3	Quantity	0.0			
4	Description	0.27			
5	InvoiceDate	0.0			
6	CustomerID	24.93			
7	InvoiceNo	0.0			
8	UnitPrice	0.0			

결측치 처리 전략

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

```
select distinct description
FROM coherent-server-456101-k4.modulabs_project.data
where stockCode ='85123A'
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보
행	description ▼			
1	WHITE HANGING HEART T-LIG...			
2	?			
3	wrongly marked carton 22804			
4	CREAM HANGING HEART T-LIG...			

결측치 처리

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

```
DELETE FROM coherent-server-456101-k4.modulabs_project.data
WHERE description is null
```

or CustomerID is null

[결과 이미지를 넣어주세요]

```
73 DELETE FROM coherent-server-456101-k4.modulabs_project.data
74 WHERE description is null
75 or CustomerID is null
76
77 SELECT count(cnt)
78 from (
79   SELECT count(*) as cnt
80   FROM coherent-server-456101-k4.modulabs_project.data
```

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

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(cnt)
from (
  SELECT count(*) as cnt
  FROM coherent-server-456101-k4.modulabs_project.data
  group by InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
  HAVING count(*) > 1) as mult_cnt
```

[결과 이미지를 넣어주세요]

```

77 SELECT count(cnt)
78 from (
79     SELECT count(*) as cnt
80     FROM coherent-server-456101-k4.modulabs_project.data
81     group by InvoiceNo, StockCode, Description, Quantity, Invo:
82     HAVING COUNT(*) > 1) as mult_cnt
83
84
85 select count(*) FROM coherent-server-456101-k4.modulabs_
86
87 CREATE OR REPLACE TABLE `coherent-server-456101-k4.modul:
88 SELECT DISTINCT *

```

쿼리 결과

작업 정보	결과	차트	JSON	실행 세부정보	실행
	f0_ ▼				
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 `coherent-server-456101-k4.modulabs_project.data` AS
SELECT DISTINCT *
FROM `coherent-server-456101-k4.modulabs_project.data`;

```

[결과 이미지를 넣어주세요]

```

86
87 CREATE OR REPLACE TABLE `coherent-server-456101-k4.modulabs_project.data` AS
88 SELECT DISTINCT *
89 FROM `coherent-server-456101-k4.modulabs_project.data`;
90
91
92 CREATE TABLE `coherent-server-456101-k4.modulabs_project.data_backup` AS
93 SELECT *
94 FROM `coherent-server-456101-k4.modulabs_project.data`;

```

쿼리 결과

작업 정보	결과	실행 세부정보	실행 그래프
<p>i 이 문으로 이름이 data인 테이블이 교체되었습니다.</p>			

11-6. 데이터 전처리(3): 오류값 처리

InvoiceNo 살펴보기

- 고유(unique)한 InvoiceNo 의 개수를 출력하기

```
select count(distinct InvoiceNo) FROM coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행		f0_ ▼				
1		22190				

- 고유한 InvoiceNo 를 앞에서부터 100개를 출력하기

```
select distinct InvoiceNo
FROM coherent-server-456101-k4.modulabs_project.data
limit 100
```

작업 정보		결과	차트	JSON	실행 세부정보	실행 그래프
행		InvoiceNo ▼				
1		541431				
2		C541433				
3		537626				
4		542237				
5		549222				
6		556201				
7		562032				
8		573511				
9		581180				
10		539318				

[결과 이미지를 넣어주세요]

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

```
SELECT *
FROM coherent-server-456101-k4.modulabs_project.data
WHERE InvoiceNo like 'C%'
LIMIT 100;
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	InvoiceNo	StockCode	Description	Quantity	Invoice	
1	C541433	23166	MEDIUM CERAMIC TOP STOR...	-74215	2011-01	
2	C545329	M	Manual	-1	2011-03	
3	C545329	M	Manual	-1	2011-03	
4	C545330	M	Manual	-1	2011-03	
5	C547388	22701	PINK DOG BOWL	-6	2011-03	
6	C547388	22645	CERAMIC HEART FAIRY CAKE ...	-12	2011-03	
7	C547388	22413	METAL SIGN TAKE IT OR LEAV...	-6	2011-03	
8	C547388	84050	PINK HEART SHAPE EGG FRYL...	-12	2011-03	
9	C547388	22784	LANTERN CREAM GAZEBO	-3	2011-03	
10	C547388	21914	BLUE HARMONICA IN BOX	-12	2011-03	
11	C547388	37448	CERAMIC CAKE DESIGN SPOT...	-12	2011-03	
12	C549955	22839	3 TIER CAKE TIN GREEN AND ...	-2	2011-04	

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

```
SELECT ROUND(SUM(CASE WHEN Quantity < 0 THEN 1 ELSE 0 END)/ count(Quantity)*100 , 1)
FROM coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON
행	f0_			
1	2.2			

StockCode 살펴보기

- 고유한 **StockCode** 의 개수를 출력하기

```
select count(distinct StockCode) from coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	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 coherent-server-456101-k4.modulabs_project.data
group by StockCode
ORDER BY sell_cnt DESC
limit 10
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행
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 coherent-server-456101-k4.modulabs_project.data
)
WHERE number_count <= 1
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	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 number_count <= 1 THEN 1 ELSE 0 END)/ count(StockCode)*100 , 2)
FROM (
  SELECT StockCode,
    LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, r'[0-9]', '')) AS number_count
  FROM coherent-server-456101-k4.modulabs_project.data
)
```


[결과 이미지를 넣어주세요]

```
4 SELECT ROUND(SUM(CASE WHEN number_count <= 1 THEN 1 ELSE 0 END) / count(StockCode)*100, 2)
5 FROM (
6   SELECT StockCode,
7   LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) AS number_count
8   FROM coherent-server-456101-k4.modulabs_project.data
9 )
10
11 begin transaction;
12
```

쿼리 결과

업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
	f0_				
1	0.48				

- 제품과 관련되지 않은 거래 기록을 제거하기

```
DELETE FROM coherent-server-456101-k4.modulabs_project.data
WHERE StockCode IN (
  SELECT DISTINCT StockCode
  FROM (
    SELECT StockCode
    FROM coherent-server-456101-k4.modulabs_project.data
    where LENGTH(StockCode) - LENGTH(REGEXP_REPLACE(StockCode, '[0-9]', '')) <= 1
  )
);
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 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 coherent-server-456101-k4.modulabs_project.data
group by Description
order by description_cnt desc
limit 30
```

[결과 이미지를 넣어주세요]

행	Description ▼	description_cnt ▼
1	WHITE HANGING HEART T-LIG...	2058
2	REGENCY CAKESTAND 3 TIER	1894
3	JUMBO BAG RED RETROSPOT	1659
4	PARTY BUNTING	1409
5	ASSORTED COLOUR BIRD ORN...	1405
6	LUNCH BAG RED RETROSPOT	1345
7	SET OF 3 CAKE TINS PANTRY ...	1224
8	LUNCH BAG BLACK SKULL.	1099
9	PACK OF 72 RETROSPOT CAKE...	1062
10	SPOTTY BUNTING	1026
11	PAPER CHAIN KIT 50'S CHRIST...	1013
12	LUNCH BAG SPACEBOY DESIGN	1006
13	LUNCH BAG CARS BILUF	1000

- 서비스 관련 정보를 포함하는 행들을 제거하기

```
delete coherent-server-456101-k4.modulabs_project.data
where Description in ('Next Day Carriage','High Resolution Image')
```

[결과 이미지를 넣어주세요]

쿼리 결과

작업 정보 **결과** 실행 세부정보 실행 그래프

i 이 문으로 data의 행 83개가 삭제되었습니다.

- 대소문자를 혼합하고 있는 데이터를 대문자로 표준화 하기

```
CREATE OR REPLACE TABLE coherent-server-456101-k4.modulabs_project.data AS
SELECT
  * EXCEPT (Description),
  UPPER(Description) AS Description
FROM coherent-server-456101-k4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 실행 세부정보 실행 그래프

D 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 coherent-server-456101-k4.modulabs_project.data;
```

[결과 이미지를 넣어주세요]

결과	차트	JSON	실행 세부정보	실행 그래프
min_price ▼	max_price ▼	avg_price ▼		
0.0	649.5	2.904956757405...		

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

```
SELECT count(quantity) AS cnt_quantity, min(quantity) AS min_quantity, max(quantity) AS max_quantity, avg(quantity) AS avg_c
FROM coherent-server-456101-k4.modulabs_project.data
WHERE UnitPrice = 0
```

[결과 이미지를 넣어주세요]

작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프
행	cnt_quantity ▼	min_quantity ▼	max_quantity ▼	avg_quantity ▼	
1	33	1	12540	420.5151515151...	

- UnitPrice = 0 를 제거하고 일관된 데이터셋을 유지하기

```
CREATE OR REPLACE TABLE coherent-server-456101-k4.modulabs_project.data AS
SELECT *
FROM coherent-server-456101-k4.modulabs_project.data
WHERE UnitPrice != 0
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 data인 테이블이 교체되었습니다.

11-7. RFM 스코어

Recency

- InvoiceDate 컬럼을 연월일 자료형으로 변경하기

```
SELECT DATE(InvoiceDate) AS InvoiceDay, *
```

FROM coherent-server-456101-k4.modulabs_project.data

[결과 이미지를 넣어주세요]

InvoiceDay ▼	InvoiceNo ▼	StockCode ▼	Quantity ▼	InvoiceID
2011-01-18	541431	23166	74215	2011-01
2011-01-18	C541433	23166	-74215	2011-01
2010-12-07	537626	22773	12	2010-12
2010-12-07	537626	22726	4	2010-12
2010-12-07	537626	22727	4	2010-12
2010-12-07	537626	85116	12	2010-12
2010-12-07	537626	85167B	30	2010-12
2010-12-07	537626	22774	12	2010-12
2010-12-07	537626	84997D	6	2010-12
2010-12-07	537626	21731	12	2010-12
2010-12-07	537626	22375	4	2010-12
2010-12-07	537626	85232D	3	2010-12
2010-12-07	537626	22725	4	2010-12

- 가장 최근 구매 일자를 MAX() 함수로 찾아보기

```
SELECT max(InvoiceDate) OVER () AS most_recent_date,
DATE(InvoiceDate) AS InvoiceDay, *
FROM coherent-server-456101-k4.modulabs_project.data
```

[결과 이미지를 넣어주세요]

most_recent_date ▼	InvoiceDay ▼	InvoiceNo ▼	StockCode ▼	Quantity ▼
2011-12-09 12:50:00 UTC	2011-07-06	559058	21843	2
2011-12-09 12:50:00 UTC	2011-06-10	556415	23008	6
2011-12-09 12:50:00 UTC	2011-06-10	556415	22427	12
2011-12-09 12:50:00 UTC	2011-09-22	567721	84968E	24
2011-12-09 12:50:00 UTC	2011-10-05	569650	22192	12
2011-12-09 12:50:00 UTC	2011-10-05	569650	23518	96
2011-12-09 12:50:00 UTC	2011-08-25	564378	23112	2
2011-12-09 12:50:00 UTC	2011-04-15	550281	23199	100
2011-12-09 12:50:00 UTC	2011-11-24	578472	23283	10
2011-12-09 12:50:00 UTC	2011-11-24	578472	23462	1
2011-12-09 12:50:00 UTC	2011-11-17	576910	22193	12
2011-12-09 12:50:00 UTC	2011-07-03	543055	22002	2

- 유저 별로 가장 큰 InvoiceDay를 찾아서 가장 최근 구매일로 저장하기

```
SELECT customerID,max(DATE(InvoiceDate)) AS InvoiceDay
FROM coherent-server-456101-k4.modulabs_project.data
group by customerID
```

[결과 이미지를 넣어주세요]

customerID	InvoiceDay
12346	2011-01-18
12347	2011-12-07
12348	2011-09-25
12349	2011-11-21
12350	2011-02-02
12352	2011-11-03
12353	2011-05-19
12354	2011-04-21
12355	2011-05-09
12356	2011-11-17
12357	2011-11-06
12358	2011-12-08

- 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 coherent-server-456101-k4.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]


CustomerID ▼	recency ▼	
12386	337	
12709	3	
12981	29	
13043	253	
13216	267	
13259	61	
13364	66	
13899	16	
13911	57	
14044	26	
14055	106	
14216	3	
14235	10	

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

```
CREATE OR REPLACE TABLE coherent-server-456101-k4.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 coherent-server-456101-k4.modulabs_project.data
  GROUP BY CustomerID
);
```

[결과 이미지를 넣어주세요]

작업 정보
결과
실행 세부정보
실행 그래프

 이 문으로 이름이 user_r인 테이블이 교체되었습니다.

Frequency

- 고객마다 고유한 InvoiceNo의 수를 세어보기

```
SELECT
  CustomerID,
  count(distinct InvoiceNo) AS purchase_cnt
```

```
FROM coherent-server-456101-k4.modulabs_project.data
group by CustomerID
```

[결과 이미지를 넣어주세요]

CustomerID	purchase_cnt
12346	2
12347	7
12348	4
12349	1
12350	1
12352	8
12353	1
12354	1
12355	1
12356	3
12357	1
12358	2
12359	6

- 각 고객 별로 구매한 아이템의 총 수량 더하기

```
SELECT
  CustomerID,
  sum(quantity) AS item_cnt
FROM coherent-server-456101-k4.modulabs_project.data
group by CustomerID
```

[결과 이미지를 넣어주세요]

CustomerID ▼	item_cnt ▼
12346	0
12347	2458
12348	2332
12349	630
12350	196
12352	463
12353	20
12354	530
12355	240
12356	1573
12357	2708
12358	242
12359	1599

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

```

CREATE OR REPLACE TABLE coherent-server-456101-k4.modulabs_project.user_rf AS

-- (1) 전체 거래 건수 계산
WITH purchase_cnt AS (
  SELECT
    CustomerID,
    count( distinct InvoiceNo) AS purchase_cnt
  FROM coherent-server-456101-k4.modulabs_project.data
  group by CustomerID
),

-- (2) 구매한 아이템 총 수량 계산
item_cnt AS (
  SELECT
    CustomerID,
    sum(quantity) AS item_cnt
  FROM coherent-server-456101-k4.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 coherent-server-456101-k4.modulabs_project.user_r AS ur
ON pc.CustomerID = ur.CustomerID;
```

[결과 이미지를 넣어주세요]

이 문으로 이름이 user_rf인 테이블이 교체되었습니다

Monetary

- 고객별 총 지출액 계산 (소수점 첫째 자리에서 반올림)

```
SELECT
  CustomerID,
  round(sum(UnitPrice * quantity)) AS user_total
FROM coherent-server-456101-k4.modulabs_project.data
group by CustomerID
```

[결과 이미지를 넣어주세요]

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

- 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 coherent-server-456101-k4.modulabs_project.user_rfm AS
SELECT
  rf.CustomerID AS CustomerID,
  rf.purchase_cnt,
  rf.item_cnt,
  rf.rececy,
  ut.user_total,
```

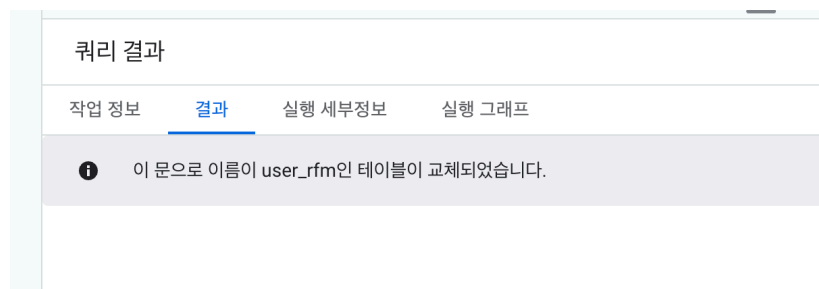
```

        (ut.user_total / rf.purchase_cnt) AS user_average
FROM coherent-server-456101-k4.modulabs_project.user_rf rf
LEFT JOIN ($@
-- 고객 별 총 지출액
SELECT
CustomerID,
round(sum(UnitPrice * quantity)) AS user_total
FROM coherent-server-456101-k4.modulabs_project.data
group by CustomerID

) ut
ON rf.CustomerID = ut.CustomerID;

```

[결과 이미지를 넣어주세요]



RFM 통합 테이블 출력하기

- 최종 **user_rfm** 테이블을 출력하기

```
select * from coherent-server-456101-k4.modulabs_project.user_rfm
```

[결과 이미지를 넣어주세요]

	작업 정보	결과	차트	JSON	실행 세부정보	실행 그래프	
		CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average
1		12713	1	505	0	795.0	795.0
2		13436	1	76	1	197.0	197.0
3		13298	1	96	1	360.0	360.0
4		15520	1	314	1	343.0	343.0
5		14569	1	79	1	227.0	227.0
6		15471	1	256	2	454.0	454.0
7		15195	1	1404	2	3861.0	3861.0
8		14204	1	72	2	151.0	151.0
9		15318	1	642	3	313.0	313.0
10		12478	1	233	3	546.0	546.0
11		14578	1	240	3	169.0	169.0
12		12442	1	181	3	144.0	144.0
13		15992	1	17	3	42.0	42.0

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]

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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]

작업 정보

결과

실행 세부정보

실행 그래프



이 문으로 이름이 user_data인 테이블이 교체되었습니다.

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 coherent-server-456101-k4.modulabs_project.user_data AS

WITH TransactionInfo AS (
  SELECT
    CustomerID,
    count(distinct InvoiceNo) AS total_transactions,
    sum(distinct CASE WHEN InvoiceNo LIKE 'C%' THEN 1 else 0 END) AS cancel_frequency
  FROM coherent-server-456101-k4.modulabs_project.data
  group by CustomerID
)

SELECT u.*, t.* EXCEPT(CustomerID), round( t.cancel_frequency / t.total_transactions * 100,2) AS cancel_rate
FROM coherent-server-456101-k4.modulabs_project.user_data AS u
LEFT JOIN TransactionInfo AS t
ON u.CustomerID = t.CustomerID;
```

[결과 이미지를 넣어주세요]

작업 정보 **결과** 실행 세부정보 실행 그래프

 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 coherent-server-456101-k4.modulabs_project.user_data
```

[결과 이미지를 넣어주세요]

정보		결과		차트	JSON	실행 세부정보	실행 그래프				
	CustomerID	purchase_cnt	item_cnt	recency	user_total	user_average	unique_products	average_interval	total_transactions	cancel_frequency	cancel_rate
1	14752	1	260	43	390.0	390.0	5	0.0	1	0	0.0
2	14861	1	117	52	126.0	126.0	6	0.0	1	0	0.0
3	17206	1	76	53	204.0	204.0	7	0.0	1	0	0.0
4	16597	1	184	4	90.0	90.0	7	0.0	1	0	0.0
5	14406	1	77	119	157.0	157.0	8	0.0	1	0	0.0
6	17881	1	32	304	133.0	133.0	8	0.0	1	0	0.0
7	13937	1	123	169	159.0	159.0	9	0.0	1	0	0.0
8	16349	1	9	290	54.0	54.0	9	0.0	1	0	0.0
9	13062	1	342	191	332.0	332.0	10	0.0	1	0	0.0
0	15388	1	88	270	141.0	141.0	10	0.0	1	0	0.0
1	15319	1	175	283	205.0	205.0	16	0.0	1	0	0.0
2	18203	1	43	157	160.0	160.0	19	0.0	1	0	0.0
3	15090	1	208	308	381.0	381.0	23	0.0	1	0	0.0

회고

[회고 내용을 작성해주세요]

Keep : 어제 복습을 해서 그래도 수월했다. 또 검토를 열심히 했다.

Problem : 아직 디테일 부족한 것 같다.

Try : 조금 더 숙련되자