

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import mysql.connector
5
6
7 db = mysql.connector.connect(host = "localhost",
8                               username = "username",
9                               password = "pass",
10                              database = "db_ec")
11
12 cur = db.cursor()
13 cur = db.cursor(buffered=True)
14

```

```

1 #List all unique cities where customers are located.
2 query = """ select distinct customer_city
3 from customers """
4
5 cur.execute(query)
6
7 data = cur.fetchall()
8
9 df = pd.DataFrame(data)
10 df.head()

```

	0
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes
4	campinas

```

1 #Count the number of orders placed in 2017.
2 query = """
3 SELECT COUNT(DISTINCT order_id)
4 FROM orders
5 WHERE YEAR(order_purchase_timestamp) = 2017
6 """
7
8 cur.execute(query)
9 data = cur.fetchone()
10 print("Total unique orders placed in 2017 are", data[0])
11

```

Total unique orders placed in 2017 are 45101

```

1 #Find the total sales per category.
2 query = """ select upper(products.product_category) category,
3 round(sum(payments.payment_value),2) sales
4 from products join order_items
5 on products.product_id = order_items.product_id
6 join payments
7 on payments.order_id = order_items.order_id
8 group by category
9 """
10
11 cur.execute(query)
12
13 data = cur.fetchall()
14
15 df = pd.DataFrame(data, columns = ["Category", "Sales"])
16 df

```

	Category	Sales
0	PERFUMERY	506738.66
1	FURNITURE DECORATION	1430176.39
2	TELEPHONY	486882.05
3	BED TABLE BATH	1712553.67
4	AUTOMOTIVE	852294.33
...
69	CDS MUSIC DVDS	1199.43
70	LA CUISINE	2913.53
71	FASHION CHILDREN'S CLOTHING	785.67
72	PC GAMER	2174.43
73	INSURANCE AND SERVICES	324.51

74 rows × 2 columns

```

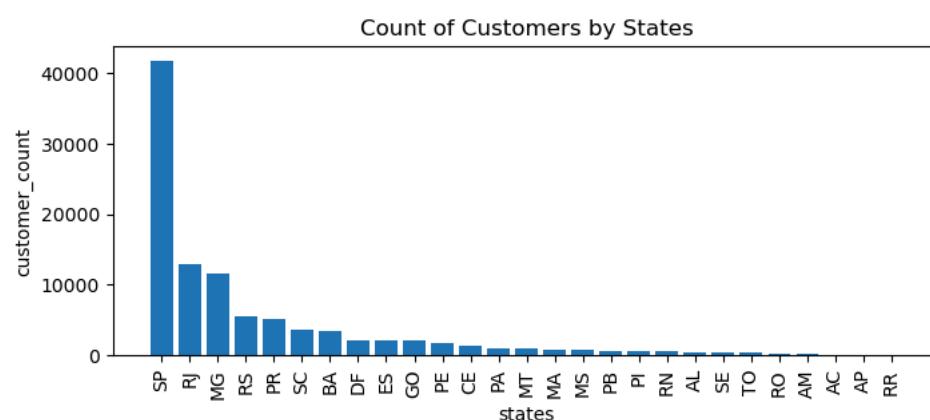
1 #Calculate the percentage of orders that were paid in installments.
2 query = """ select ((sum(case when payment_installments >= 1 then 1
3 else 0 end))/count(*))*100 from payments
4 """
5
6 cur.execute(query)
7
8 data = cur.fetchall()
9
10 "the percentage of orders that were paid in installments is", data[0][0]
('the percentage of orders that were paid in installments is',
Decimal('99.9981'))

```

```

1 #Count the number of customers from each state.
2 query = """ select customer_state ,count(customer_id)
3 from customers group by customer_state
4 """
5
6 cur.execute(query)
7
8 data = cur.fetchall()
9 df = pd.DataFrame(data, columns = ["state", "customer_count" ])
10 df = df.sort_values(by = "customer_count", ascending= False)
11
12 plt.figure(figsize = (8,3))
13 plt.bar(df["state"], df["customer_count"])
14 plt.xticks(rotation = 90)
15 plt.xlabel("states")
16 plt.ylabel("customer_count")
17 plt.title("Count of Customers by States")
18 plt.show()

```



```

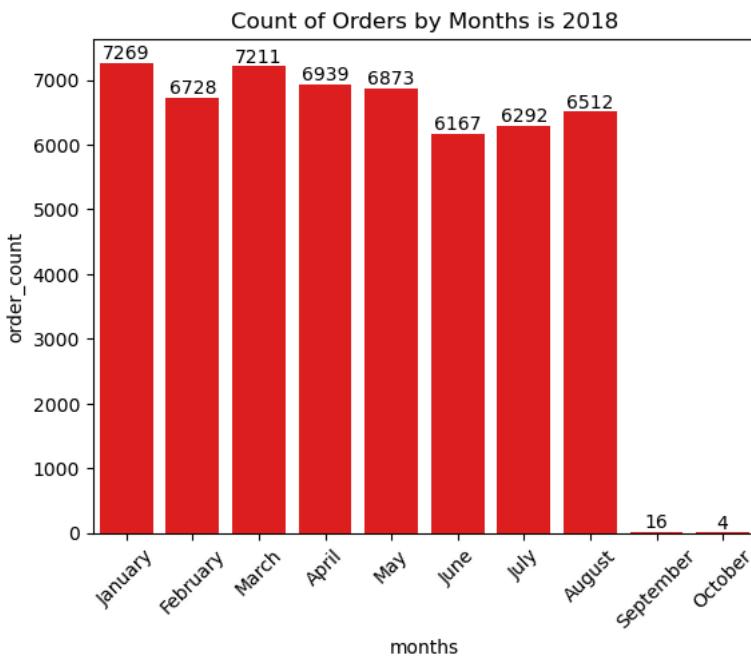
1 #Calculate the number of orders per month in 2018.
2 query = """ select monthname(order_purchase_timestamp) months, count(order_id) order_count
3 from orders where year(order_purchase_timestamp) = 2018
4 group by months
5 """
6
7 cur.execute(query)

```

```

8
9 data = cur.fetchall()
10 df = pd.DataFrame(data, columns = ["months", "order_count"])
11 o = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October"]
12
13 ax = sns.barplot(x = df["months"], y = df["order_count"], data = df, order = o, color = "red")
14 plt.xticks(rotation = 45)
15 ax.bar_label(ax.containers[0])
16 plt.title("Count of Orders by Months is 2018")
17
18 plt.show()

```



```

1 #Find the average number of products per order, grouped by customer city.
2 query = """with count_per_order as
3 (select orders.order_id, orders.customer_id, count(order_items.order_id) as oc
4 from orders join order_items
5 on orders.order_id = order_items.order_id
6 group by orders.order_id, orders.customer_id)
7
8 select customers.customer_city, round(avg(count_per_order.oc),2) average_orders
9 from customers join count_per_order
10 on customers.customer_id = count_per_order.customer_id
11 group by customers.customer_city order by average_orders desc
12 """
13
14 cur.execute(query)
15
16 data = cur.fetchall()
17 df = pd.DataFrame(data,columns = ["customer city", "average products/order"])
18 df.head(10)

```

	customer city	average products/order
0	padre carvalho	7.00
1	celso ramos	6.50
2	datas	6.00
3	candido godoi	6.00
4	matias olímpio	5.00
5	cidelândia	4.00
6	picarra	4.00
7	morro de sao paulo	4.00
8	teixeira soares	4.00
9	curralinho	4.00

```

1 #Calculate the percentage of total revenue contributed by each product category.
2 query = """select upper(products.product_category) category,
3 round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2) sales_percentage

```

```

4 from products join order_items
5 on products.product_id = order_items.product_id
6 join payments
7 on payments.order_id = order_items.order_id
8 group by category order by sales_percentage desc"""
9
10
11 cur.execute(query)
12 data = cur.fetchall()
13 df = pd.DataFrame(data,columns = ["Category", "percentage distribution"])
14 df.head()

```

Category percentage distribution		
0	BED TABLE BATH	10.70
1	HEALTH BEAUTY	10.35
2	COMPUTER ACCESSORIES	9.90
3	FURNITURE DECORATION	8.93
4	WATCHES PRESENT	8.93

```

1 #Identify the correlation between product price and the number of times a product has been purchased.
2 import numpy as np
3 cur = db.cursor()
4 query = """select products.product_category,
5 count(order_items.product_id),
6 round(avg(order_items.price),2)
7 from products join order_items
8 on products.product_id = order_items.product_id
9 group by products.product_category"""
10
11 cur.execute(query)
12 data = cur.fetchall()
13 df = pd.DataFrame(data,columns = ["Category", "order_count","price"])
14
15 arr1 = df["order_count"]
16 arr2 = df["price"]
17
18 a = np.corrcoef([arr1,arr2])
19 print("the correlation is", a[0][-1])

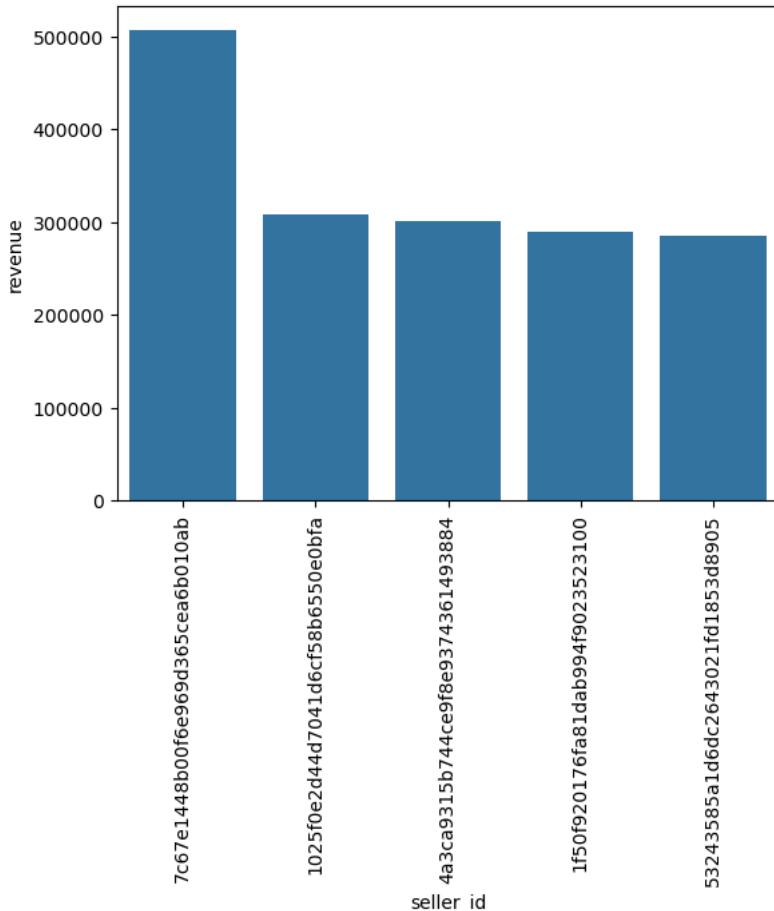
```

the correlation is -0.10631514167157562

```

1 #Calculate the total revenue generated by each seller, and rank them by revenue.
2 query = """ select *, dense_rank() over(order by revenue desc) as rn from
3 (select order_items.seller_id, sum(payments.payment_value)
4 revenue from order_items join payments
5 on order_items.order_id = payments.order_id
6 group by order_items.seller_id) as a """
7
8 cur.execute(query)
9 data = cur.fetchall()
10 df = pd.DataFrame(data, columns = ["seller_id", "revenue", "rank"])
11 df = df.head()
12 sns.barplot(x = "seller_id", y = "revenue", data = df)
13 plt.xticks(rotation = 90)
14 plt.show()

```



```

1 #Calculate the moving average of order values for each customer over their order history.
2 query = """select customer_id, order_purchase_timestamp, payment,
3 avg(payment) over(partition by customer_id order by order_purchase_timestamp
4 rows between 2 preceding and current row) as mov_avg
5 from
6 (select orders.customer_id, orders.order_purchase_timestamp,
7 payments.payment_value as payment
8 from payments join orders
9 on payments.order_id = orders.order_id) as a"""
10 cur.execute(query)
11 data = cur.fetchall()
12 df = pd.DataFrame(data)
13 df

```

	0	1	2	3
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.74	114.739998
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.41	67.410004
2	0001fd6190edaaf884bcf3d49edf079	2017-02-28 11:06:43	195.42	195.419998
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.35	179.350006
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.01	107.010002
...
103881	fffecc9f79fd8c764f843e9951b11341	2018-03-29 16:59:26	71.23	27.120001
103882	ffffeda5b6d849fbd39689bb92087f431	2018-05-22 13:36:02	63.13	63.130001
103883	ffff42319e9b2d713724ae527742af25	2018-06-13 16:57:05	214.13	214.130005
103884	fffffa3172527f765de70084a7e53aae8	2017-09-02 11:53:32	45.50	45.500000
103885	fffffe8b65bbe3087b653a978c870db99	2017-09-29 14:07:03	18.37	18.370001

103886 rows × 4 columns

```

1 #Calculate the cumulative sales per month for each year.
2 query = """select years, months , payment, sum(payment)
3 over(order by years, months) cumulative_sales from
4 (select year(orders.order_purchase_timestamp) as years,
5 month(orders.order_purchase_timestamp) as months,
6 round(sum(payments.payment_value),2) as payment from orders join payments

```

```

7 on orders.order_id = payments.order_id
8 group by years, months order by years, months) as a
9 """
10 cur.execute(query)
11 data = cur.fetchall()
12 df = pd.DataFrame(data)
13 df

```

	0	1	2	3
0	2016	9	252.24	252.24
1	2016	10	59090.48	59342.72
2	2016	12	19.62	59362.34
3	2017	1	138488.04	197850.38
4	2017	2	291908.01	489758.39
5	2017	3	449863.60	939621.99
6	2017	4	417788.03	1357410.02
7	2017	5	592918.82	1950328.84
8	2017	6	511276.38	2461605.22
9	2017	7	592382.92	3053988.14
10	2017	8	674396.32	3728384.46
11	2017	9	727762.45	4456146.91
12	2017	10	779677.88	5235824.79
13	2017	11	1194882.80	6430707.59
14	2017	12	878401.48	7309109.07
15	2018	1	1115004.18	8424113.25
16	2018	2	992463.34	9416576.59
17	2018	3	1159652.12	10576228.71
18	2018	4	1160785.48	11737014.19
19	2018	5	1153982.15	12890996.34
20	2018	6	1023880.50	13914876.84
21	2018	7	1066540.75	14981417.59
22	2018	8	1022425.32	16003842.91
23	2018	9	4439.54	16008282.45
24	2018	10	589.67	16008872.12

```

1 #Calculate the year-over-year growth rate of total sales.
2 query = """with a as(select year(orders.order_purchase_timestamp) as years,
3 round(sum(payments.payment_value),2) as payment from orders join payments
4 on orders.order_id = payments.order_id
5 group by years order by years)
6
7 select years, ((payment - lag(payment, 1) over(order by years))/lag(payment, 1) over(order by years)) * 100 from a"""
8
9
10 cur.execute(query)
11 data = cur.fetchall()
12 df = pd.DataFrame(data, columns = ["years", "oyy % growth"])
13 df

```

	years	oyy % growth
0	2016	NaN
1	2017	12112.703761
2	2018	20.000924

```

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

```

```

8 b as (select a.customer_id, count(distinct orders.order_purchase_timestamp) next_order
9 from a join orders
10 on orders.customer_id = a.customer_id
11 and orders.order_purchase_timestamp > first_order
12 and orders.order_purchase_timestamp <
13 date_add(first_order, interval 6 month)
14 group by a.customer_id)
15
16 select 100 * (count( distinct a.customer_id)/ count(distinct b.customer_id))
17 from a left join b
18 on a.customer_id = b.customer_id ;"""
19
20 cur.execute(query)
21 data = cur.fetchall()
22

```

[None,]

```

1 #Identify the top 3 customers who spent the most money in each year.
2 query = """select years, customer_id, payment, d_rank
3 from
4 (select year(orders.order_purchase_timestamp) years,
5 orders.customer_id,
6 sum(payments.payment_value) payment,
7 dense_rank() over(partition by year(orders.order_purchase_timestamp)
8 order by sum(payments.payment_value) desc) d_rank
9 from orders join payments
10 on payments.order_id = orders.order_id
11 group by year(orders.order_purchase_timestamp),
12 orders.customer_id) as a
13 where d_rank <= 3 ;"""
14
15 cur.execute(query)
16 data = cur.fetchall()
17 df = pd.DataFrame(data, columns = ["years","id","payment","rank"])
18 sns.barplot(x = "id", y = "payment", data = df, hue = "years")
19 plt.xticks(rotation = 90)
20 plt.show()

```

