RACHID EL KHAYAT

FINTECH

Capestone Project:

Analyzing the data and finding patterns for " buy now pay later" startup company

The outline

- The Questions
- Data exploration
- Answer to Question 1 ----------
- Answer to Question 2 ----------
- Answer to Question 3 ----------
- Answer to Question 4 ----------
- Answer to Question 5 ----------

The Questions

1. Using the payment information, calculate:

    a. Delinquency Rate

$$Delinquency\ Rate = \frac{\#\ of\ paid\ late\ payments\ +\ \#\ of\ unpaid\ late\ payments}{Total\ \#\ of\ \ payments}$$

    b. Recovery Rate

$$Recovery\ Rate = \frac{\#\ of\ paid\ late\ payments}{\#\ of\ paid\ late\ payments\ +\ \#\ of\ unpaid\ late\ payments}$$

2. We observe a large volume of expired orders, but they are multiple attempts of the same order. Find the number of unique orders that were unsuccessful due to expiry, and were not eventually successful.

3. Provide the information for the orders where the 3rd installment has been made before the 2nd. Down payments should not be included in this calculation as they will always be paid to confirm the order.

4. What are the top reasons for orders being declined, and what would you suggest to Tamara to reduce the decline rate?

5. Using evidence from the data, how would you suggest that ABC improves our bottom-line?

Exploring the data

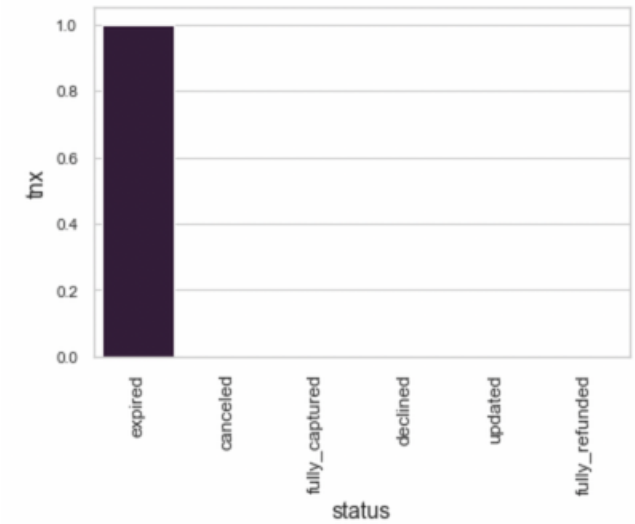
orders Dataframe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 979609 entries, 0 to 979608
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   created_at         979609 non-null  datetime64[ns]
 1   id_order           979609 non-null  object
 2   customer_id        958981 non-null  object
 3   merchant_id        979609 non-null  object
 4   merchant_category  979609 non-null  object
 5   payment_type       979609 non-null  object
 6   status             979609 non-null  object
 7   total_amount       979609 non-null  float64
 8   canceled_amount    979609 non-null  float64
 9   captured_amount    979609 non-null  float64
 10  refunded_amount    979609 non-null  float64
 11  paid_amount        979609 non-null  float64
dtypes: datetime64[ns](1), float64(5), object(6)
```

After running thorough quality check on each data field to ensure the integrity of the data, we can conclude that the data satisfy by the most part the data quality features: timeliness, consistency, completeness, and accuracy. a deep dive investigation will take place as we move along with the exploration.

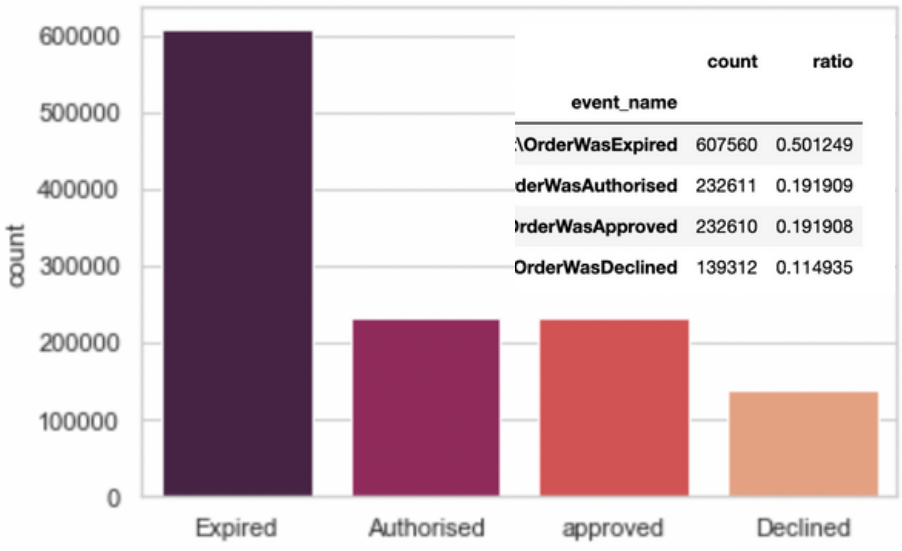
Expired orders represent 62% of the entire orders

There are only 9 of the customers with NaN customer_id has a 'fully captured status



status	tnx
expired	220575
canceled	27
fully_captured	9
declined	9
updated	7
fully_refunded	1

order_events Data frame

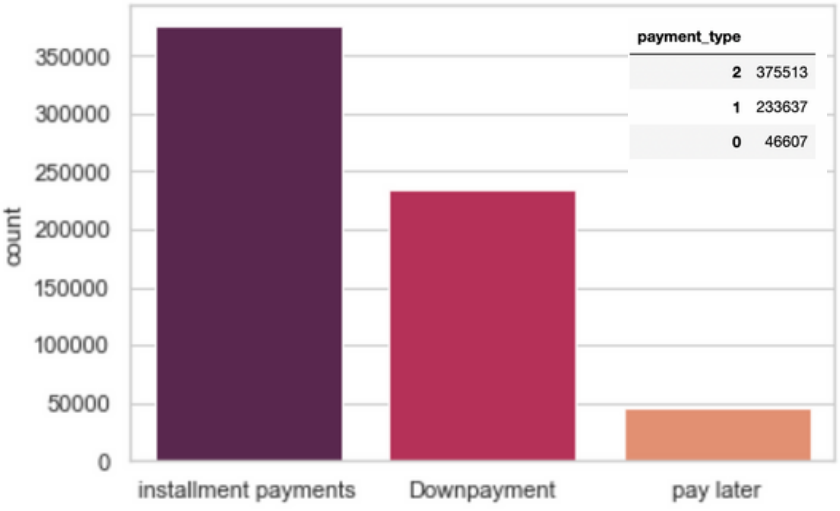


event_name	count	ratio
\OrderWasExpired	607560	0.501249
derWasAuthorised	232611	0.191909
rderWasApproved	232610	0.191908
OrderWasDeclined	139312	0.114935

```
RangeIndex: 1212093 entries, 0 to 1212092
Data columns (total 6 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   id           1212093 non-null  int64
 1   event_id     1212093 non-null  object
 2   order_id     1212093 non-null  object
 3   event_name   1212093 non-null  object
 4   payload      1212093 non-null  object
 5   created_at   1212093 non-null  object
dtypes: int64(1), object(5)
```

Payment Data frame

	payment_type	total_amount	refunded_amount	paid_amount	pending_paid_amount	issued_refund_amount	refund_shipping_fee_amount	late_fee_amount
count	655757.000000	655757.000000	655757.000000	655757.000000	655757.0	655757.000000	655757.0	655757.000000
mean	1.501567	295.484069	36.959233	256.546644	0.0	3.122639	0.0	3.890568
std	0.626215	250.892193	142.699661	245.771025	0.0	41.745382	0.0	21.297464
min	0.000000	29.115900	0.000000	0.000000	0.0	0.000000	0.0	0.000000
25%	1.000000	155.700000	0.000000	122.830000	0.0	0.000000	0.0	0.000000
50%	2.000000	227.771800	0.000000	199.815000	0.0	0.000000	0.0	0.000000
75%	2.000000	352.331800	0.000000	319.859700	0.0	0.000000	0.0	0.000000
max	2.000000	6713.092000	6107.592000	6713.092000	0.0	4196.980000	0.0	259.500000



payment_type	
2	375513
1	233637
0	46607

```
RangeIndex: 655757 entries, 0 to 655756
Data columns (total 18 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   id_payment                  655757 non-null  object
 1   payment_type                655757 non-null  int64
 2   order_id                    655757 non-null  object
 3   customer_id                 655757 non-null  object
 4   merchant_id                 655757 non-null  object
 5   status                      655757 non-null  object
 6   total_amount                655757 non-null  float64
 7   refunded_amount             655757 non-null  float64
 8   paid_amount                 655757 non-null  float64
 9   pending_paid_amount         655757 non-null  float64
 10  issued_refund_amount        655757 non-null  float64
 11  refund_shipping_fee_amount  655757 non-null  float64
 12  late_fee_amount             655757 non-null  float64
 13  due_date                    655757 non-null  datetime64[ns]
 14  paid_date                   576836 non-null  datetime64[ns]
 15  last_late_fee_recorded_at   36420 non-null   object
 16  created_at                  655757 non-null  datetime64[ns]
 17  updated_at                  655757 non-null  datetime64[ns]
dtypes: datetime64[ns](4), float64(7), int64(1), object(6)
memory usage: 90.1+ MB
```

Question 1

a. Delinquency Rate

$$Delinquency\ Rate = \frac{\#of\ paid\ late\ payments\ +\ \#of\ unpaid\ late\ payments}{Total\ \#of\ \ payments}$$

5.9%

b. Recovery Rate

$$Recovery\ Rate = \frac{\#of\ paid\ late\ payments}{\#of\ paid\ late\ payments\ +\ \#of\ unpaid\ late\ payments}$$

79%

Steps implements to achieve this result:

1. filter out the unneeded records where the status is related to refund.
2. Identify the criteria that will allow us to identify the payment as late as my identifier.
3. Create a field 'late_payment' that flags the payment as 'paid' or 'unpaid' , non_late payment were considerd as zero in that field.
4. Apply the calculations

Important Note:

The 'late_fee_amount' was chosen to be the flag that determines if the record to be considered a late_payment or not. kindly find hereunder the reason that lead to this decision.

When comparing the two dates, where the paid_date is bigger than the due_date, logically this should be considered as a late payment, however there was no late fee and the"last_late_fee_recorded_at" is empty.

so i found that the criteria of taking date differences between the date seems to be not very reliable and for that reason i opt to chose the late_fee_amount to be my indicator.

Question 2

Find the number of unique orders that were unsuccessful due to expiry, and were not eventually successful.

12

Tracing expired order input in all the three tables

Expired orders

Total # of records is:
607588/ 979609
62%

expired	607588
fully_captured	217827
declined	139302
canceled	6958
partially_refunded	3657
fully_refunded	3511
partially_captured	382
authorised	266
updated	118

Unique records

created_at	921889
id_order	979609
customer_id	309284
merchant_id	898
merchant_category	16
payment_type	2
status	9
total_amount	85394
canceled_amount	4524
captured_amount	42437
refunded_amount	3095
paid_amount	46371

****Data integrity check has been done on this dataset to ensure that we are working with a proper data that will lead to trusted results., the quality seems to be fine interms of consistency, Accuracy, relevency timleness.
however we have 220628 missing data (NAs) in the customer_id which we have to investigate.

Expired orders Events

Total # of records is:
607635/ 1212093
50%

Event_name Count

OrderWasExpired	607553
OrderWasApproved	38
OrderWasAuthorised	35
OrderWasDeclined	9

Unique records

id	607635
event_id	607635
order_id	607588
event_name	4
payload	607635
created_at	54031

****We have 37 duplicated record that seem to be an anomaly to the data, as they follow the following workflow subsequently (approved, authorised) , 3 were approved and then expired.
in addition to te other 10 declined tnxs that were expired later on.
We can say that we have 34 tnxs that shouldnt be included with the expired tnxs. the next page shows the details of all the above mentioned orders

Expired orders Payments

Total # of records is:
47501/ 655757
7.2%

Status of expired records

refunded	47447
paid	34
pending	20

Unique records

id_payment	47501
payment_type	2
order_id	47499
customer_id	33228
merchant_id	709
status	3
total_amount	11009
refunded_amount	10992
paid_amount	35
pending_paid_amount	1
issued_refund_amount	2
late_fee_amount	3
due_date	47288
paid_date	47
last_late_fee_recorded_at	5
created_at	47289
updated_at	34030

****there are two payments with order_ids (aabfb088-b98f-4ac7-b8e8-cbbf396e30b5 , 4bb122c5-67b2-415d-9156-7282bc1621e2) were refunded twice,
looking at the time delta between the first and the second transaction in the field (updated_at), we can deduce that it was a system glitch

- tnxs : means transactions

order_id	approved	authorised	Declined	expired
2ce5ed04-5d9e-4c98-b1d3-8dced7709d45	1.0	NaN	NaN	1.0
3185154b-bab1-4670-a62c-f4ee18d7f8bb	1.0	NaN	NaN	1.0
de10e18d-6264-4108-815a-030252d028b2	1.0	NaN	NaN	1.0
13fa6e96-bb3b-442f-93fd-97590d65edf7	NaN	NaN	1.0	1.0
1ae43e2d-ffdc-497b-9f8d-55e4b32e9164	NaN	NaN	1.0	1.0
3a7df0d0-2ab4-4998-ab8d-e5f5fa574c01	NaN	NaN	1.0	1.0
6341b680-79a7-4c94-a672-3eddee8b651c	NaN	NaN	1.0	1.0
683b16a4-a3f9-4f48-b836-73615450cee5	NaN	NaN	1.0	1.0
a57b8692-f585-4d46-ae3f-25a6bdf2763f	NaN	NaN	1.0	1.0
ae398d24-56fe-462b-b054-c3c2af10b68a	NaN	NaN	1.0	1.0
af389fd4-7a38-4d62-bd5a-8dd7a0e1ada9	NaN	NaN	1.0	1.0
db49dc36-b539-4ae4-b656-3d23f3cbd82e	NaN	NaN	1.0	1.0

3 orders were approved then they eventually expired
9 orders were declined and then they eventually expired

Steps implements to achieve this result:
1- filtering the orders with 'expired' status
2- look up the list of 'expired' orders in the order_events df
3- checking duplicates to know which order has the event name changed
4-pivoting the orders against the event_name to trace the change of status
5- displaying results

Question 3

Provide the information for the orders where the 3rd installment has been made before the 2nd. Down payments should not be included in this calculation as they will always be paid to confirm the order.

17

Steps implements to achieve this result:

1- filtering the payment df based on status and payment_type
2- counting the number of instalment for each order, as we only need the orders that has 3 or more installments.
3-create a column to rank the due_date of the installments per order
4-create a column to rank the payments for each order
5-compare the second rank of the due_date with the rank of the paid
6-present results.

** We noticed that most of the payments for the second and third happened on te same same, which means that theclients were settling the overdue payment(s) along with the third payment

	order_id	payment_type	status	total_amount	due_date	paid_date	paid_amount	due_rank	paid_rank
508086	61f1fb27-45fd-4f77-9dce-27cfba66c140	2	paid	288.0277	2021-05-10 07:55:45	2021-07-07 13:38:07	374.5277	2	3
541667	dd06aecf-c5ee-41f5-be40-7427de991ba5	2	paid	244.7950	2021-05-16 11:01:56	2021-03-26 00:42:19	244.7950	2	3
546285	7e115081-fa33-4a1c-98d7-c62302adee8f	2	paid	65.7400	2021-05-17 08:04:44	2021-03-25 20:11:47	65.7400	2	3
548147	293ce15e-9987-4a13-8af0-1baf31b14b37	2	paid	402.2077	2021-05-17 16:31:29	2021-04-02 22:38:16	402.2077	2	3
548893	1688bbca-ca02-41a4-8605-6f6864f82b69	2	paid	1622.1518	2021-05-17 19:04:36	2021-07-05 11:59:13	1622.1518	2	3
576567	66eb5832-15de-4153-9fd4-2bcffbd6d2a5	2	paid	288.0277	2021-05-24 08:36:35	2021-05-22 07:15:22	288.0277	2	3
611438	7799116a-fdd1-4ff4-9ea6-d0461839825c	2	paid	648.4559	2021-05-27 10:49:48	2021-05-27 05:46:16	648.4559	2	3
614156	63cec0b4-5670-4375-95c6-c38d4a8cb767	2	paid	1601.6167	2021-05-27 15:07:55	2021-06-28 05:53:11	1601.6167	2	3
621798	c15b7dd6-6345-4b80-a7c4-6b23191aa607	2	paid	190.3000	2021-05-28 04:22:58	2021-06-07 14:53:46	190.3000	2	3
627839	cbcf8159-3052-4fd3-93f0-97c7bc74f0b3	2	paid	285.1559	2021-05-28 17:01:30	2021-04-16 06:30:12	285.1559	2	3
630677	7ae09b85-7d7d-4d1b-b2a3-e85574c05d27	2	paid	874.4458	2021-05-28 21:26:07	2021-07-09 08:10:35	874.4458	2	3
637709	bbfc96e7-0af0-4a3c-b357-acf93c9b1af3	2	paid	115.0450	2021-05-29 14:15:37	2021-05-27 09:27:43	115.0450	2	3
641127	d4ad6f61-9253-4051-a78c-aa35800cd4a0	2	paid	547.8218	2021-05-29 18:54:50	2021-06-28 00:19:22	547.8218	2	3
642613	baba263e-1255-4f54-ad2b-fb394f8cb04a	2	paid	547.8218	2021-05-29 21:13:10	2021-05-27 16:24:31	547.8218	2	3
647836	7762b914-c479-4bab-b030-2f6c1e172df3	2	paid	193.9676	2021-05-30 09:48:55	2021-05-28 06:55:05	193.9676	2	3
653387	4cf39110-29e6-450b-8958-a63608dc47a2	2	paid	134.9400	2021-05-30 19:43:10	2021-05-15 16:56:18	134.9400	2	3
654654	afb4067b-0d46-4ae4-86f4-ac66fed036e5	2	paid	646.0858	2021-05-30 21:48:42	2021-05-08 06:35:45	646.0858	2	3

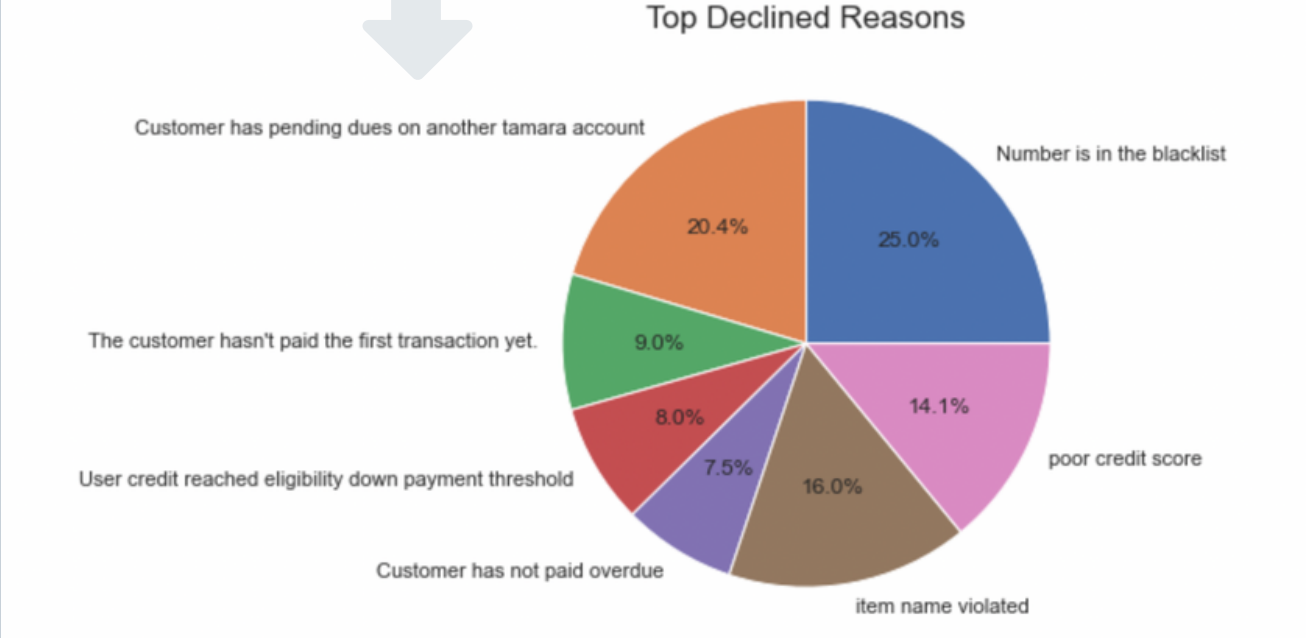
Question 4

What are the top reasons for orders being declined, and what would you suggest to Tamara to reduce the decline rate?

-Blacklisted
-customer dues
-item name violated
-Poor credit score
.
.

	reason	count	ratio
0	Number is in the blacklist	30478	0.218775
1	Customer has pending dues on another tamara ac...	24863	0.178470
2	item name violated	19542	0.140275
3	poor credit score	17158	0.123162
4	The customer hasn't paid the first transaction...	10978	0.078802
5	User credit reached eligibility down payment t...	9732	0.069858
6	Customer has not paid overdue	9148	0.065666

In addition to the to the reason count that we calculated from the payload, once we dig more into the data we notice that we have two important reasons that also have a major effect on the declined items.



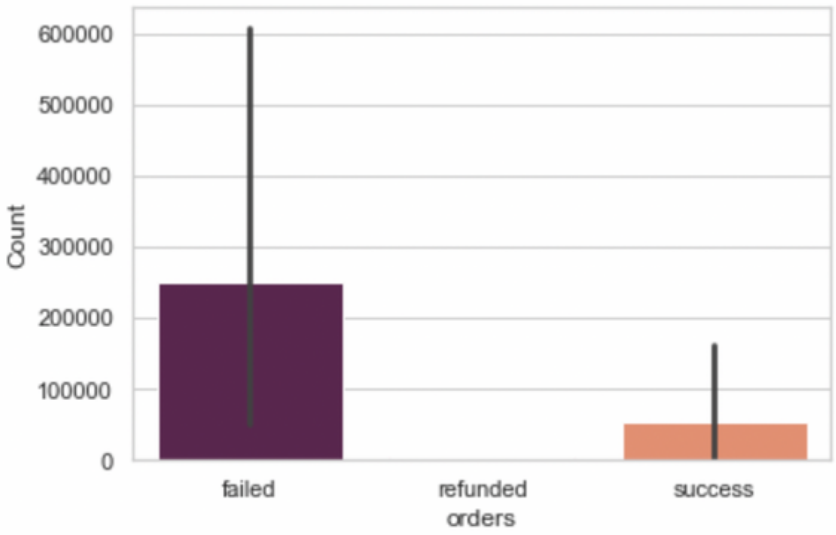
Steps implemented to achieve this result:
2- filter by event_name equals to 'orderwasdeclined'
1- transform 'payload' field from json to a DF
3- aggregate and count the frequency of each reason
4- deep dive into the resulting df thoroughly to get more insights.
5- presenting the results

To reduce the declined amount, we need to start by looking at the key reason to figure out the way that will contribute in increasing the successful orders and not compromising on the safety of scams and fraud from the data i would suggest the following:
-Raise the threshold to the credit score.
-Revise the list of items were the names are mentioned as violated.
-increase the down payment threshold.

Question 5

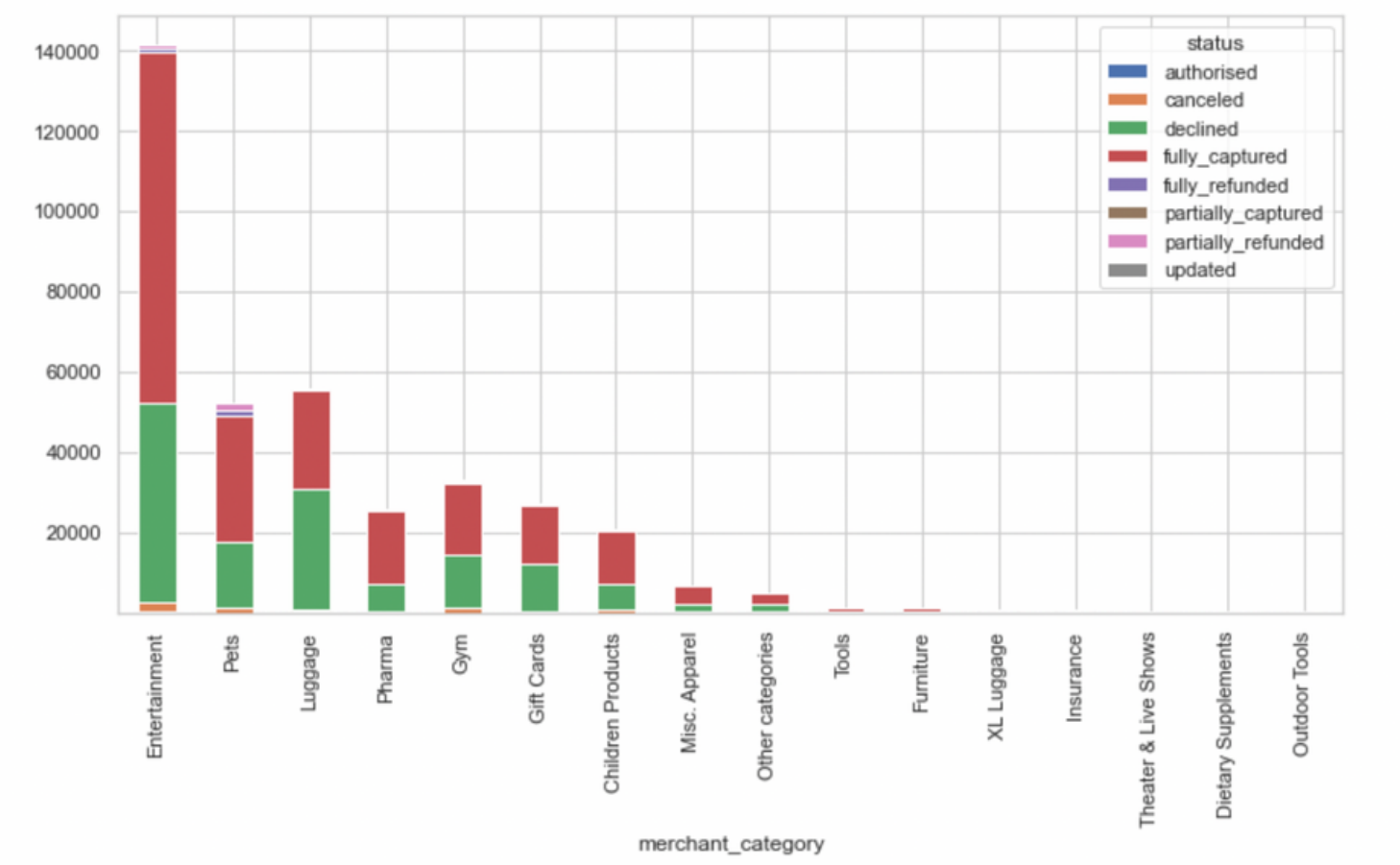
In order to increase our bottomline by looking at the data we to to the following:

1- Increase the number of successful orders by identifying the challenges and implement the right solutions. As seen in the below image the failure ratio now is 3:1 which seems to be a serious issue that we need to look at.



cat	status	Count	Ratio
failed	canceled	6958	0.007103
	declined	139302	0.142202
	expired	607588	0.620235
refunded	fully_refunded	3511	0.003584
	partially_refunded	3657	0.003733
success	authorised	266	0.000272
	fully_captured	217827	0.222361
	partially_captured	382	0.000390
	updated	118	0.000120

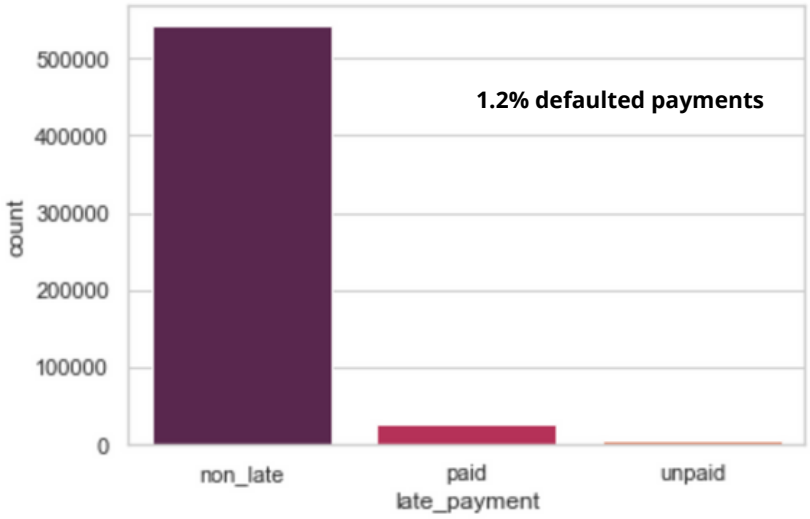
By plotting the merchant_category against the status of the order we can notice that entertainment category has the lion share followed by the pets and the luggage. it seems that there is a huge potential to increase the number of orders from different industries which seems to be untapped yet.
We can also notice that pets category is generating many refund orders, which absolutely effects the bottomlines in a direct way. " this issue to be investigated.



status	authorised	canceled	declined	expired	fully_captured	fully_refunded	partially_captured	partially_refunded	updated
merchant_category									
Entertainment	68	2639	49246	279514	87505	848	49	904	12
Pets	35	981	16555	65799	31465	1218	11	1924	15
Luggage	106	793	30010	96431	24250	455	7	166	18
Pharma	18	267	6741	44259	18476	228	8	103	2
Gym	8	1058	13442	36891	17697	292	151	224	35
Gift Cards	0	389	11555	34398	14913	294	2	186	4
Children Products	4	529	6399	25361	13423	21	151	7	23
Misc. Apparel	6	135	1988	10308	4541	75	1	50	3
Other categories	10	110	2066	5929	2461	47	2	54	6
Tools	4	6	307	2111	738	5	0	4	0
Furniture	3	22	327	930	630	2	0	4	0
XL Luggage	2	6	136	3058	616	9	0	2	0
Insurance	1	9	261	1482	554	4	0	1	0
Theater & Live Shows	0	10	90	484	300	5	0	1	0
Dietary Supplements	1	3	161	547	217	6	0	27	0
Outdoor Tools	0	1	18	86	41	2	0	0	0

- Decrease the number of defaulters This is the main reason that effects the bottom line in a direct way. looking at the chart below (expired orders were removed) we can a defaulters rate of 1.2% which is considered a big rate for the type of business we are in. we can do that by applying higher restrictions of the approval process, by decreasing the threshold of tolerance in the credit score and create a bigger black listed customers. In addition to restricting the access of a defaulter as we can see that same client has defaulted on different orders in different dates.

late_payment	count	ratio
non_late	541867.0	0.940027
paid	27429.0	0.047584
unpaid	7142.0	0.012390
total	576438.0	1.000000



orders	cust
1	5422
2	546
3	121
4	36
5	12
6	4
7	3
8	2

In the above table we can see that we have many clients who defaulted on many orders not only one.

applying these steps will help the company to decreasing the delinquency rate and increasing the recovery rate. which in its terms will boost the bottom lines

	created_at	customer_id	order_id	late_fee_amount	due_date	paid_date
39770	2020-12-05 21:42:35	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	89b170e8-ac90-4e03-acad-1ec122a6dfed	0.00000	2021-02-04 21:42:15	NaN
39771	2020-12-05 21:42:15	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	89b170e8-ac90-4e03-acad-1ec122a6dfed	0.00000	2020-12-05 21:42:15	2020-12-05 21:42:33
39772	2020-12-05 21:42:35	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	89b170e8-ac90-4e03-acad-1ec122a6dfed	0.00000	2021-01-04 21:42:15	NaN
63940	2020-12-11 02:48:36	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	b7581b45-4010-4968-b1fd-76538de328d8	0.00000	2020-12-11 02:48:36	2020-12-11 02:48:42
63941	2020-12-11 02:48:44	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	b7581b45-4010-4968-b1fd-76538de328d8	0.00000	2021-01-10 02:48:36	2021-01-06 19:23:41
63942	2020-12-11 02:48:44	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	b7581b45-4010-4968-b1fd-76538de328d8	86.50000	2021-02-10 02:48:36	2021-03-04 12:55:18
462967	2021-03-04 13:19:03	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	8e217950-72b2-4f0a-a811-9df9b06a32dc	194.30495	2021-04-03 13:19:03	NaN
463048	2021-03-04 13:33:18	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	cc106109-05d7-44b8-ac7e-3f6bed6f107b	201.89100	2021-04-03 13:33:18	NaN
463255	2021-03-04 14:13:16	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	9e4e6669-0088-4680-bfa3-044c1097c62c	0.00000	2021-04-03 14:13:16	2021-05-03 16:02:12
482440	2021-03-07 19:12:40	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	57f5460b-f9c4-4a85-ac8c-a4ee81261519	75.07335	2021-04-06 19:12:40	NaN
505520	2021-03-10 22:05:01	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	ece9de02-dcc3-49b3-83bd-60df18143a30	79.83950	2021-04-09 22:05:01	NaN
548426	2021-03-18 17:35:13	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	6d760cdf-9d92-43ad-ab55-8619532fd2e7	94.28500	2021-04-17 17:35:13	NaN
575143	2021-03-25 00:43:42	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	e4b55dbf-96a0-40cb-b2bb-79fd0f6bbf82	109.54360	2021-04-24 00:43:42	NaN
581347	2021-03-25 22:19:23	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	ba2b3e93-8578-44df-8a0a-c4a996bb75db	71.03380	2021-04-24 22:19:23	NaN
602012	2021-03-27 16:22:32	b1f2723e-dd4e-4d0a-a6ac-f9f28cc126b9	2728792b-2763-4a74-995a-ab57006750cf	133.97120	2021-04-26 16:22:32	NaN

In the above table we are showing an example of the same client who has defaulted on different orders in different dates

Thank you