Data Analyst Challenge

You are hired at a relatively new bank called Unbank. Unbank is growing fast and has a microservice backend architecture that is generating a bunch of data on a daily basis. Unbank culture empowers employees, everyone can take actions based on data and it is simple for unbankers to create new datasets that fulfill their needs. However, with great power comes great responsibility, and Unbank data warehouse is becoming a big mess.

One analyst, in hurry to solve a problem that involves transactions and installments, created a table named "**transactions"** and added this table to the already existing data warehouse. The representation of a small part of the data warehouse and a small preview of what is inside the "**transactions"** table is presented below:

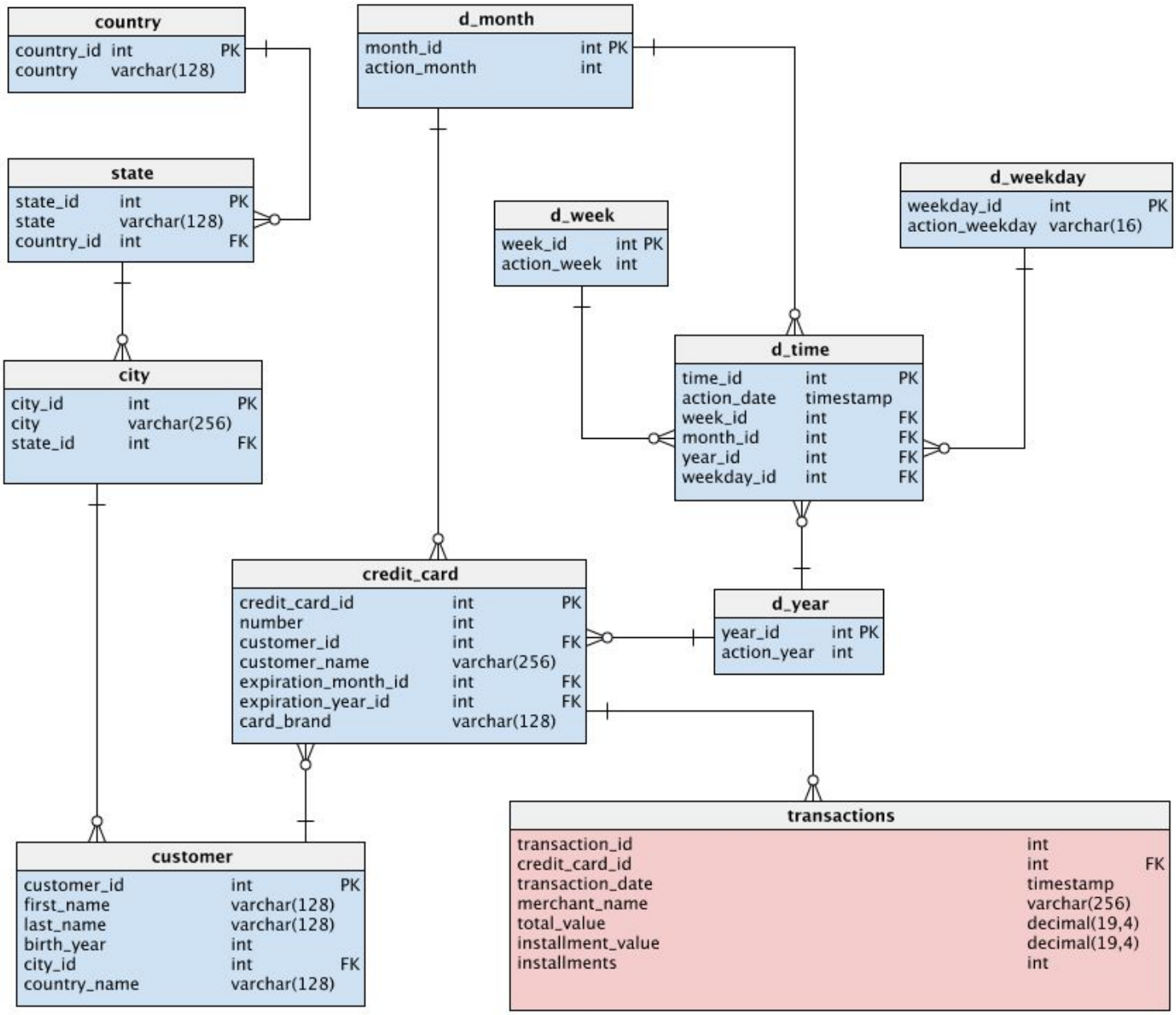

Image 1: data warehouse representation

transaction_id	credit_card_id	transaction_date	merchant_name	total_value	installment_value	installments
1	11111111	2018-01-10T00:00:00	Colorful Soaps	19.99	19.99	1
2	22222222	2018-01-11T00:01:00	Cantina da Mamma	43.5	43.5	1
3	33333333	2018-01-12T01:02:00	Boulevard Hotel	129	129	1
4	11111111	2018-01-15T11:11:11	Micas Bar	225.9	75.3	3
5	11111111	2018-01-15T11:11:11	Micas Bar	225.9	75.3	3
6	11111111	2018-01-15T11:11:11	Micas Bar	225.9	75.3	3
7	22222222	2018-01-18T22:10:01	IPear Store	9999.99	9999.99	1
8	11111111	2018-02-20T21:08:32	Forrest Paintball	1337	1337	1
9	44444444	2018-02-22T00:05:30	Unicorn Costumes	100	50	2
10	44444444	2018-02-22T00:05:30	Unicorn Costumes	100	50	2

Table 1: a small preview of transactions table

In the analyst conception, this table is perfect because you can query the total value of a transaction and also check the value of all installments, everything in just one place. If I want to know how much credit card *"11111111"* spent on "*Micas Bar*" on "2018-01-15" and also know how much this person needs to pay, I can just query this table and get *"total_value"* and *"installment_value"*.

However, for some strange reason, another analyst called Jane Hopper is having problems to create an experiment using this table. She is trying to create a monthly bill for our clients that consists on the sum of all "*installments_values*" for that month. That means, if the number of "*installments*" is 1, the "*total_value*" should be charged for the same month of the transaction, if the number of installments is n > 1, the "installments_value" should be charged for the current transaction month and the following months.

Example: The current monthly bills, based on the small preview presented before, for the *"credit_card_id" = "11111111"*, should be:

January: R$ 75.30 + R$ 19.99 = R$ 95.29
February: R$ 75.30 + R$ 1337 = R$ 1412.30
March: R$ 75.30

Jane tried hard but did not succeed. She decided to ask for your help to solve this problem - help Jane to get the information she needs! You can assume that your changes will be made on a test environment - no worries about breaking things, the company will survive. Create a presentation about your findings, explaining the decisions you made to solve this problem.

What we expect from you:

- You should not only create a SQL query that retrieves the information needed but also fix design problems that you found on the data warehouse implementation (image 1).
- Change table/fields, and even create new tables, in order to generate a better database design.
- Given that this sort of problem can happen again, how would you act to prevent/mitigate the recurrence of it?
- As mentioned, we're empowering people to use data as they wish. How would you help them to better find, understand, and consume data?