Problem Statement 1

Just like every other day at Autochek, The overall goal starts by getting our dealers to list their cars, displaying these cars to the prospective customers on our marketplace and then providing affordable car loans to make life easy, you as our data engineer suddenly gets a data request around one of our most delicate dataset, Apparently, the business leaders would like to see a summarized table generated from data of the **customer (borrower_table)**, the **loans they currently have(loans table)**, the dates they have been **scheduled to repay (payment_schedule)**, **how frequent they are paying back (loan_payment)**, lastly a table that shows, history of times customers have **missed their payments (missed_payment)** you have a simple job, super hero, given these tables, generate a sql query for the desire columns using one or more of these tables.

Note: Be creative with the use of **group by**, **order by** and **windows function**

Data and Schema Explanation

(1) Borrower_table (One borrower, one record) - https://docs.google.com/spreadsheets/d/1mfm4NUfv4wOJfMdjOIA5k99N79JFJDJeTBhzUflYtzs/edit?usp=sharing
- Borrower_id (pk)
- State
- City
- zip code

(2) Loan_table (One borrower, one or multiple loans) - https://drive.google.com/file/d/1Lid1RDMGNpXWv2PyU0Alsv-sSBnkGMuS/view?usp=sharing
- Borrower_id (fk)
- Loan_id(pk)
- Date_of_release
- Term
- InterestRate
- LoanAmount
- Downpayment
- Payment_frequency
- Maturity_date

(3) Payment_Schedule (One loan, one or multiple payment schedules) - https://docs.google.com/spreadsheets/d/1LawsJQtLGpO6AQZFDpgSUEn8A_TVFD9I/edit?usp=sharing&ouid=112035781102155860914&rtpof=true&sd=true
- loan_id(fk)
- schedule_id(pk)
- Expected_payment_date
- Expected_payment_amount

(4) Loan_payment (One loan, one or multiple payments) - https://drive.google.com/file/d/1qdV_WcVmvo7VsyxJxfU8RXpdOgKZ7V3A/view?usp=sharing
   - loan_id(fk)
   - payment_id(pk)
   - Amount_paid
   - Date_paid

Expected steps:

1. Go through the above schemas, try to understand the relationship between the schemas
2. Transform the data based on the output as listed in the figure below and save output in this format“output_name.xls”
3. Calculate PAR Days - Par Days means the number of days the loan was not paid in full. Eg If the loan repayment was due on the 10th Feb 2022 and payment was made on the 15th Feb 2022 the par days would be 5 days
4. Do note for each day a customer missed a payment the amount_at_risk is the total amount of money we are expecting from the customer as at that time, for instance, if the customer owes for 5000 from month 1 and 10000 for current month the amount_at_risk will be the total amount 5000 + 10000 = 15000,
5. You can use SQL or Python to achieve this
6. Push this to a github repo alongside the solution for statement 2, detailing your solution.

Using the above schema information, simulate a query that generates the following,

	city	zip code	payment_frequency	maturity_date	current_days_past_due	last_due_date	last_repayment_date	amount_at_risk

Problem Statement 2

Hey engineer, we would like to get all our exchange rate from one source, code up a script that gets the rate of 7 countries and a scheduler to pull this rate 2 times a day, first at 1am and second at 11pm. Rates should be saved per day meaning, no duplicate records for a single date.

Rate website url → https://www.xe.com/xecurrencydata/
Rate website doc → https://xecdapi.xe.com/docs/v1/

Expected steps:

7. Create a free account on XE
8. Go through the documentation
9. Write a script to pull the data and save in this format, **timestamp**, which is the datetime in which the record was pulled, **currency_from**, which is the currency you are converting from (should always be USD), **USD_to_currency_rate**, which is the rate to 1 NGN cost in USD, **currency_to_USD_rate**, which is the rate of 1 USD cost in NGN, **currency_to** is the currency you are converting to from USD, In our case 7 currencies, for **Nigeria**, **Ghana**, **Kenya**, **Uganda**, **Morocco**, **Côte d'Ivoire** and **Egypt**.
10. Set up a scheduler that pulls this data for the specified timeframe.
11. Push your code to a github page, detailing your script and how to run it.

timestamp	**currency_from**	USD_to_currency_rate	currency_to_USD_rate	currency_to

Note: You have a query limit and a 7days free trial so be sure to not exhaust your query limit.