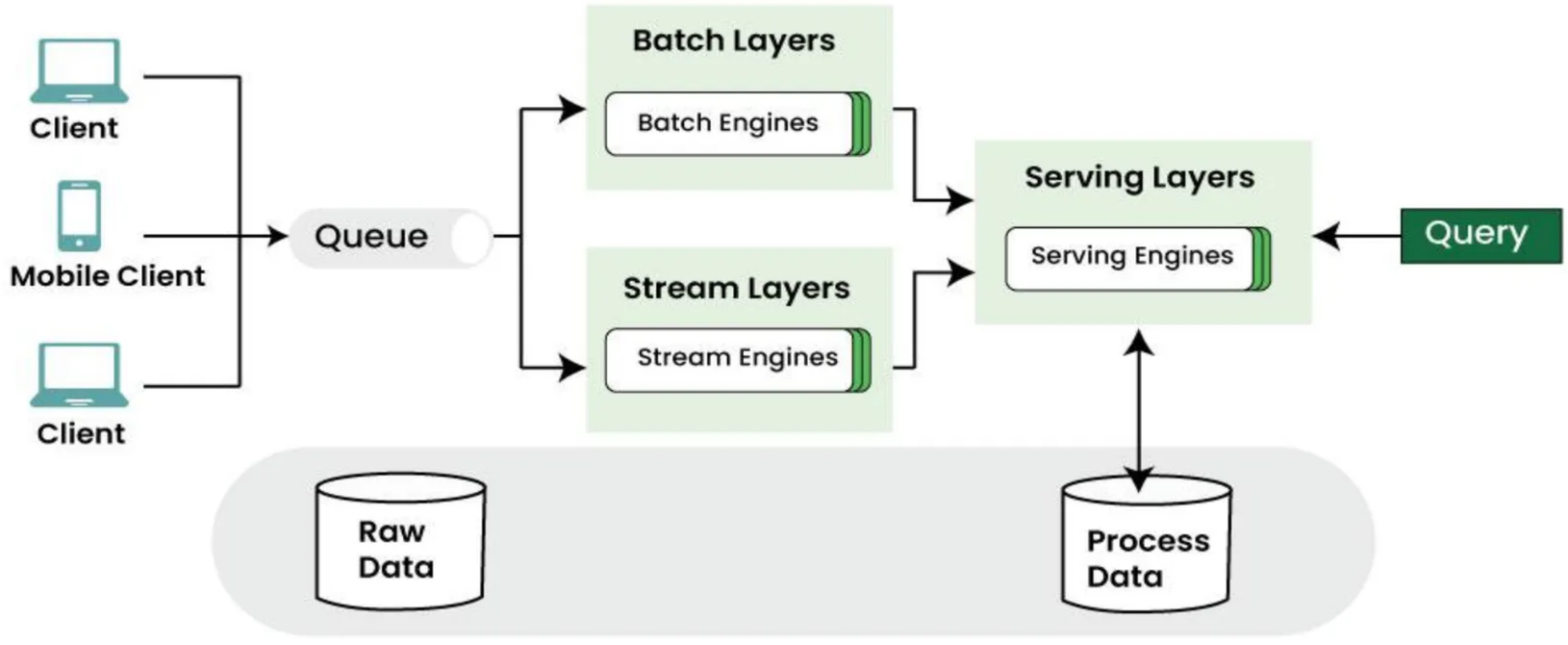
Project 4 Banking System with Lambda Architecture

Project Overview

This project implements a credit card transaction processing system using a Lambda Architecture with both stream and batch processing components. The system tracks card balances, approves or declines transactions based on validation rules, and manages data flow between stream and batch layers.

1. **Stream Layer**: Processes transactions in real-time, validating them and tracking pending balances.
2. **Batch Layer**: Periodically processes accumulated data, updating the master dataset with finalized values.
3. **Serving Layer**: Provides access to the processed data through relational database.



Dataset

You will work on different datasets individually. The link to the datasets is provided here, and your dataset ID remains the same as Project 2 and 3. Submission with wrong dataset is not accepted.

1. **Customers.csv** records all important information about the credit card users

customer_id	name	phone_number	address	email	credit_score	annual_income
1	Charles Garc	632-232-1488	3658 Angel Harbor, Staten Is	garzaanthony@exar	685	81000
2	Cristian Sant	803-484-1106	9196 Jesse Rapids, Rocheste	lrobinson@example	705	24700
3	Gina Moore	589-299-6881	4553 Katherine Estate, State	curtis61@example.	749	132800
4	Joe Martinez	843-833-6925	8786 Munoz Valley, Bronx, N	smiller@example.n	660	168300
5	Matthew Mej	664-850-6977	4742 Ray Squares, Queens, N	wyattmichelle@exa	645	108200
6	Jacob Clark	588-476-4598	9981 Chavez Village, Buffalo	jacqueline19@exar	652	54200
7	James Mayo	780-935-6155	526 Amanda Gardens, Cheek	gabriellecameron@	648	113700
8	Derek Clark	469-964-8019	2341 Lydia Valley, Syracuse,	jason76@example.r	793	60000
9	Justin Baker	842-363-7916	8349 Thomas Dam, Troy, NY	zhurst@example.cc	684	152000

2. **Cards.csv** records each cards identical information

card_id	customer_id	card_type_id	card_number	expiration_da	credit_limit	current_bala	issue_date
1	1	1	5433-0365-4145-8685	26-Sep	4300	98.97	2/3/24
2	1	4	6444-0196-5569-8169	25-Aug	46900	612.55	7/24/24
3	2	2	6011-3561-5951-4846	25-Sep	5000	231.39	10/6/22
4	3	4	6529-9468-0443-6995	26-Mar	35500	1669.15	2/19/24
5	3	3	3721-4895-1343-3200	28-Feb	16600	177.37	9/25/22
6	4	1	5367-6320-1632-8708	25-Nov	2700	85.02	8/28/22
7	5	1	6011-9579-8687-2774	28-Jan	4200	140.19	6/17/24
8	6	1	5143-4558-1223-6231	28-Jun	3500	48.05	6/29/22
9	7	4	3490-9670-5466-8893	28-Feb	26600	1107.75	11/17/22
10	7	3	3456-2729-8069-9016	28-Mar	14100	342.38	12/24/23

3. **Credit_card_types.csv** records the different types of credit card we offer

card_type_id	name	credit_score_	credit_score_	credit_limit_	credit_limit_	annual_fee	rewards_rate
1	Basic	580	669	1000	5000	0	0.005
2	Gold	670	739	5000	10000	95	0.01
3	Platinum	740	799	10000	25000	195	0.02
4	Diamond	800	850	25000	50000	495	0.03
5	Black	800	850	50000	100000	695	0.05

4. **Transactions.csv** records real-time transaction history from April 1st to 4th in 2025

	transaction_	card_id	merchant_name	timestamp	amount	location	transaction_type	related_transaction_id
1	510	21	Newman, Miranda and	4/4/25 7:51	327.9	1220 Davis Inlet, Yonkers, NY 12579	p2p_transfer	
2	511	49	Crawford PLC	4/4/25 8:05	-254.7	585 Lisa Coves, Troy, NY 17113	cancellation	396
3	512	10	Boyle-Smith	4/4/25 8:09	141.25	9751 Tina Neck, White Plains, NY 11364	atm_withdrawal	
4	513	29	Rodriguez-Mills	4/4/25 8:13	173.26	1170 Morris Row, Niagara Falls, NY 14205	purchase	
5	514	5	Blair Ltd	4/4/25 8:49	-432.53	1154 Atkins Forge, Hoodview, FL 32975	refund	
6	515	3	Ward PLC	4/4/25 8:51	405.3	9440 Joel Park, Bronx, NY 16247	bill_payment	
7	516	28	Bailey and Sons	4/4/25 9:06	-329.86	7111 Daniels Lodge, New York, NY 15313	refund	
8	517	21	Hall LLC	4/4/25 9:29	-369.08	629 Kelly Dam, Michaelshire, FL 36185	refund	
9	518	38	Gibson-Freeman	4/4/25 9:40	264.22	8398 Neal Wells, New York, NY 14290	wire_transfer	
10	519	33	Clay, Freeman and Wea	4/4/25 9:49	247.87	682 Finley Road, Binghamton, NY 18361	purchase	
11	520	44	Hanson PLC	4/4/25 9:56	7440.9	935 Laura View, Yonkers, NY 18021	wire_transfer	
12	521	11	Ortiz-Hudson	4/4/25 10:37	276.11	122 Derrick Flats, Staten Island, NY 17169	p2p_transfer	
13	522	18	Garcia, Daniels and Du	4/4/25 10:40	100.16	107 Susan Points, Cheektowaga, NY 18772	bill_payment	
14	523	32	Smith Ltd	4/4/25 10:50	435.81	5 Stacie Island, New York, NY 19004	wire_transfer	

Requirements and Grading Rubric (100 points)

Note: Pandas package is not allowed for processing data

0. Go Over the Stream Processing Example

Repository is shared through GitLab, please go over the example and make sure you have Kafka set up correctly.
Demo on this example is not required.

1. Stream Layer (40 points)

- Using Kafka for stream processing is required.
- The **stream producer** sends one day's transactions to Kafka (10 points)
    - Producer takes data from **transactions.csv** and sends the transactions to consumer.

- The **stream consumer** processes these transactions in **time sequence** (20 points)
  - Consumer takes transaction from producer and process data at real-time
    - Have to process data points with time sequence
    - **Collect all data and process in batch will receive 0 credit for this part**
  - Validate transactions and update pending credit card balances
    - Decline the transaction if:
      - One transaction ≥ 50% of the card's credit limit
      - Merchant location is too far away from the user's address (the function to calculate the distance will be given)
      - Accumulated pending balance (including the existing balance) exceeds the credit card limit
      - Print real-time log messages to terminal for all declined transactions
    - Update pending credit card balances
      - All approved transactions have "pending" status
      - All denied transactions have "declined" status
      - All refunds or cancelled transactions need to reflect on the pending balance

transaction_	card_id	merchant_name	timestamp	amount	location	transaction_	related_trans	status	pending_balance
1	54	Briggs, Mcclain and Simmo	4/1/25 1:05	476.85	3454 Dodson	p2p_transfer		declined	1272.93
2	5	Morris, Campbell and Owe	4/1/25 1:21	-315.12	6383 Randy C	refund		pending	15679.56
3	28	Fernandez and Sons	4/1/25 1:24	242	4506 Joshua	bill_payment		pending	9019.42
4	32	Santos-Christian	4/1/25 1:50	-114.16	4730 Morgan	refund		pending	3871.08
5	4	Weber, Wheeler and Gonza	4/1/25 2:16	169.1	3288 Michae	wire_transfer		pending	14257.77
6	42	Bright-Francis	4/1/25 2:28	-177.55	1924 Jimmy S	refund		pending	4843.58
7	14	Kennedy-Schmidt	4/1/25 2:28	203.25	725 Kevin Pra	purchase		pending	4339.4
8	50	Mendez-Graham	4/1/25 2:35	294.56	2436 William	p2p_transfer		pending	6386.68
9	40	Davis-Bowen	4/1/25 2:59	129.15	1902 Adriana	wire_transfer		pending	18904.25

  - Submit the stream processed file as **results/stream_transactions.csv** (10 points)

2. Batch Layer (25 points)

- Process the stream transaction file **stream_transactions.csv** after stream process
- Approve all pending transactions and save results as **results/batch_transactions.csv** (5 points)
- Update the current balance in **cards.csv** and save as **results/cards_updated.csv**

- Calculate the current user credit score based on the total credit usage across all cards
  - Function for calculating the credit score will be given
  - Update the user credit score in **customers.csv** and save as **results/customers_updated.csv** (10 points)
  - Reduce the user credit limit if credit card score drop and save to **results/cards_updated.csv** (10 points)

3. Serving Layer (10 points)

- Put initial data in MySQL
- Stream layer reads data from MySQL
- After all jobs are finished, update the data into MySQL
- Show user query in demo

3. Project Submission and Documentation (15 points)

- Create merge request for your implementation and pass pipeline check (5 points)
- Write a README file that includes (5 points)
  - Brief introduction to the project and the dataset you are working on
  - List of required packages
  - Instructions for running the code and expected outputs
    - Please be very specific on how to run the code for this project
    - You may choose to have `main.py` or not
  - Use proper Markdown formatting
- Code Requirements (5 points)
  - Data processing functions should be member functions of different classes
  - Format code with `black` before submitting
  - **If Serving layer is implemented, add `.env.example` for easier code setup**

5. Demo Questions (10 points)

- Clearly state what you did for this project
- Answer questions regarding the project

Important Notes

- **Project demo and implementation submission deadline: May 1st in class**
  - **After 1:45pm, the implementation part of grade is locked, your CSV files will be graded using the last commit made before class ends**

- **ReadMe Deadline: May 2nd 1am** (only minor changes with code accepted with this deadline)
- Students who do not attend the project demo day will receive only 50% of their Project 4 grade
- Partial credit will be given for CSV output files that partially match the expected format