Name: Ayushi Patel

Roll no.: 20BIT059

Que-1

SELECT

date\_trunc('day', block\_timestamp) AS transaction\_date,

AVG(COUNT(\*)) AS avg\_transactions\_per\_day

FROM

polygon.transactions

GROUP BY

transaction\_date

ORDER BY

transaction\_date;

Que-2

SELECT

receiver AS token\_holder,

SUM(amount) AS total\_op\_tokens

FROM

op\_token\_distributions\_optimism.transfer\_mapping

WHERE

token = 'OP' -- Assuming 'OP' is the symbol of the OP token

GROUP BY

token\_holder

ORDER BY

total\_op\_tokens DESC

LIMIT

10;

Task 2

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int minCoins(vector<int>& coins, int amount) {

int n = coins.size();

// Create a table to store the minimum number of coins needed for each amount

vector<int> dp(amount + 1, INT\_MAX);

// The minimum number of coins needed to make change for 0 is always 0

dp[0] = 0;

// Iterate through all amounts from 1 to the target amount

for (int i = 1; i <= amount; ++i) {

// Iterate through all coin denominations

for (int j = 0; j < n; ++j) {

// Check if the current coin denomination is less than or equal to the current amount

if (coins[j] <= i && dp[i - coins[j]] != INT\_MAX) {

// Update the minimum number of coins needed for the current amount

dp[i] = min(dp[i], dp[i - coins[j]] + 1);

}

}

}

// The result is stored in dp[amount]

return dp[amount] == INT\_MAX ? -1 : dp[amount];

}

int main() {

// Example usage

vector<int> denominations = {1, 5, 10};

int targetAmount = 27;

int result = minCoins(denominations, targetAmount);

if (result != -1) {

cout << "The minimum number of coins needed: " << result << endl;

} else {

cout << "It's not possible to make change for the given amount using the provided denominations." << endl;

}

return 0;

}

Output: -



Que-2

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Job {

int deadline;

int processingTime;

int jobNumber;

};

// Function to compare jobs based on deadlines in descending order

bool compareJobs(const Job& a, const Job& b) {

return a.deadline > b.deadline;

}

void jobScheduling(vector<Job>& jobs) {

int n = jobs.size();

// Sort the jobs based on deadlines in descending order

sort(jobs.begin(), jobs.end(), compare