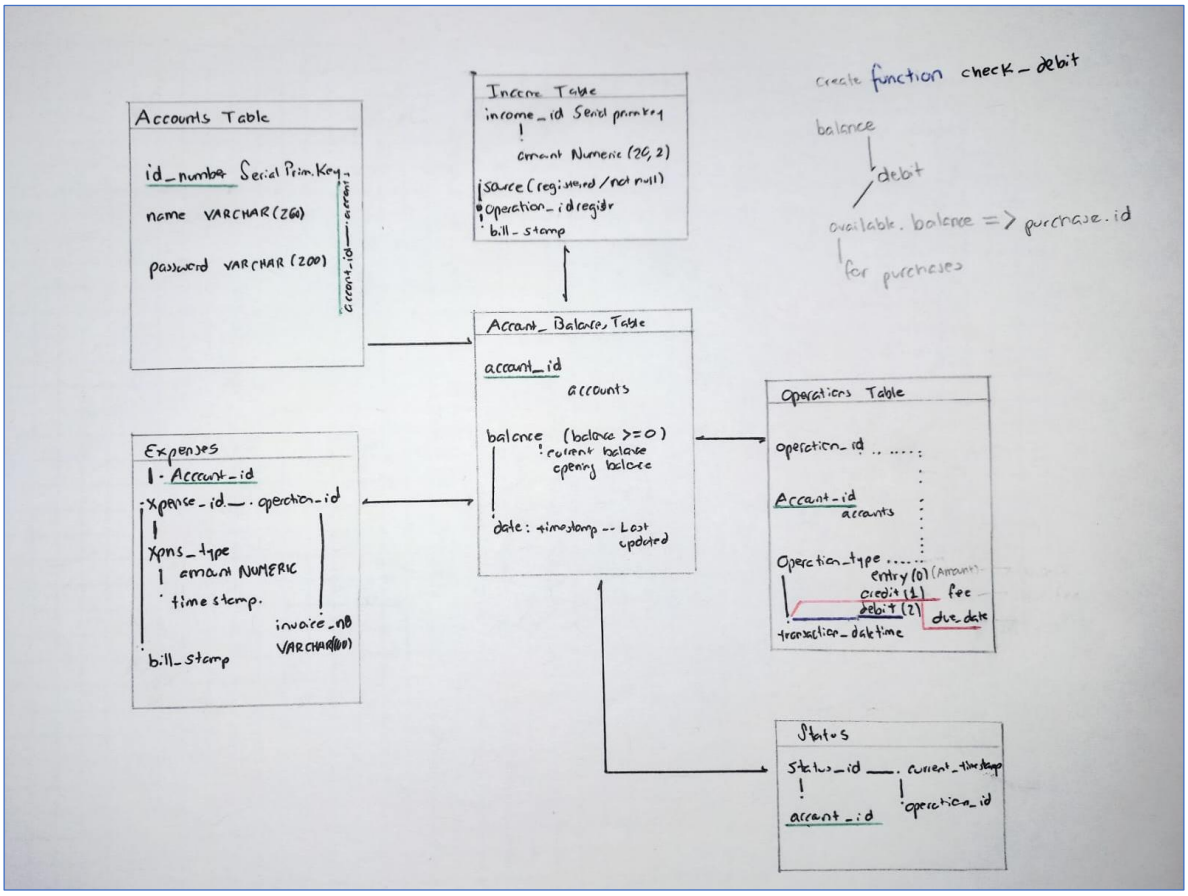
Santiago Cruz Todd
Tarea N° 3

Actividad:




```
-- 1. Accounts Table
CREATE TABLE accounts (
    id_number PRIMARY KEY,  -- User's unique identifier
    name VARCHAR(256) NOT NULL,    -- User's name
    password VARCHAR(256) NOT NULL -- Password for the bank account
);

-- 2. Account Balances Table
CREATE TABLE account_balances (
    account_id INTEGER REFERENCES accounts(id_number) PRIMARY KEY,
    balance NUMERIC(20, 2) NOT NULL CHECK (balance >= 0), -- Current balance
    opening_balance NUMERIC(20, 2) NOT NULL,  -- Opening balance when the account was created
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- Last updated
```

```
);

-- 3. Operations Table (Credit/Debit)
CREATE TABLE operations (
    operation_id PRIMARY KEY,
    account_id INTEGER REFERENCES accounts(id_number) ON DELETE CASCADE, -- Linked to the accounts table
    operation_type VARCHAR(10) NOT NULL CHECK (operation_type IN ('debit', 'credit')), -- Credit or Debit
    amount NUMERIC(20, 2) NOT NULL CHECK (amount > 0.0),  -- Amount involved in the transaction
    fee NUMERIC(20, 2), -- Bank fee for credit (if applicable)
    debit_allowed BOOLEAN DEFAULT TRUE, -- Flag to check if debit is allowed based on balance
    due_at DATE,  -- Payment due date for credits
    transaction_datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- Date and time of the transaction
);

-- 4. Status Table (Transaction Logs)
CREATE TABLE status (
    status_id PRIMARY KEY,
    account_id INTEGER REFERENCES accounts(id_number),
    operation_id INTEGER REFERENCES operations(operation_id),
    status_message VARCHAR(512) NOT NULL, -- Message such as "Debit rejected due to insufficient funds"
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP -- When the status was recorded
);

-- 5. Expenses Table
CREATE TABLE expenses (
    expense_id PRIMARY KEY,
    account_id INTEGER REFERENCES accounts(id_number),
    operation_id INTEGER REFERENCES operations(operation_id),
    expense_type VARCHAR(50),  -- Type of expense, e.g., purchase, transfer
    amount NUMERIC(20, 2) NOT NULL, -- Expense amount
    expense_datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- Date of the expense
);

-- 6. Incomes Table
CREATE TABLE incomes (
    income_id SERIAL PRIMARY KEY,
```

```
    account_id INTEGER REFERENCES accounts(id_number),
    amount NUMERIC(20, 2) NOT NULL, -- Income amount
    source VARCHAR(256) NOT NULL, -- From which company or entity
    income_datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP  -- Date the income
was received
);

-- 7. Trigger Function to Update Account Balances After Operations
CREATE FUNCTION update_balance() RETURNS TRIGGER AS $$
BEGIN
    -- Update the balance for the account after each operation (debit/credit)
    IF (NEW.operation_type = 'debit') THEN
        UPDATE account_balances
        SET balance = balance - NEW.amount, updated_at = CURRENT_TIMESTAMP
        WHERE account_id = NEW.account_id;
    ELSIF (NEW.operation_type = 'credit') THEN
        UPDATE account_balances
        SET balance = balance + (NEW.amount - NEW.fee), updated_at =
CURRENT_TIMESTAMP
        WHERE account_id = NEW.account_id;
    END IF;
    RETURN NEW;
END;

-- 8. Trigger to Automatically Update Balance After Insert
CREATE TRIGGER trigger_update_balance
AFTER INSERT ON operations
FOR EACH ROW
EXECUTE PROCEDURE update_balance();

-- 9. Trigger Function to Check Debit Balance Before Inserting Operation
CREATE FUNCTION check_debit_balance() RETURNS TRIGGER AS $$
BEGIN
    -- Check if the balance is enough for a debit transaction
    IF (NEW.operation_type = 'debit') THEN
        DECLARE available_balance NUMERIC(20, 2);
        IF available_balance < purchase.id THEN
            RAISE EXCEPTION 'Insufficient funds for debit operation.';
        END IF;
    END IF;
END;

-- 10. Trigger to Check Balance Before Debit Operation
CREATE TRIGGER trigger_check_debit
```

```
BEFORE INSERT ON operations
FOR EACH ROW
EXECUTE PROCEDURE check_debit_balance();
```