LAPORAN PROYEK

Personal Financial Management System

KELOMPOK 12

Anggota :

Nama	**NIM**
Arnold Daniel Manalu	11323003
James Frans Rizky Tambunan	11323036
Vanesha Ganesya Siahaan	11323037
Jessica Anastasya Purba	11323045

D-III Teknologi Informasi 2023

Fakultas Vokasi
Institut Teknologi Del
2024/2025

DAFTAR ISI

BAB I: PENDAHULUAN

1.1. Latar Belakang

Manajemen keuangan pribadi merupakan hal yang penting untuk kestabilan finansial seseorang. Namun, saat ini banyak orang kesulitan dalam mencatat transaksi keuangan, menganalisis pengeluaran, serta pemasukan mereka. Oleh karena itu, dibutuhkan sistem berbasis database yang dapat membantu dalam pencatatan pemasukan, pengeluaran, dan pengelolaan saldo. Sistem ini juga memanfaatkan teknologi terkini untuk menyajikan data dalam bentuk visualisasi grafis dan memberikan notifikasi peringatan untuk menjaga saldo minimum. Selain itu, sistem ini memungkinkan pengguna untuk melihat saldo mereka dalam mata uang yang berbeda dengan mengakses kurs mata uang terkini, sehingga memudahkan pengelolaan keuangan dalam konteks global.

1.2. Rumusan Masalah

- Kesulitan dalam mencatat dan memantau transaksi keuangan secara akurat dan efisien, yang menyebabkan kesalahan dalam pencatatan dan pemantauan saldo.
- Kurangnya analisis pengeluaran yang jelas dan visualisasi data yang mudah dipahami, sehingga sulit untuk mengambil keputusan finansial yang tepat.
- Kurangnya notifikasi peringatan untuk saldo minimum, yang dapat menyebabkan pengguna tidak menyadari jika saldo mereka mendekati batas.
- Terbatasnya integrasi dengan sistem eksternal, seperti nilai tukar mata uang terkini, yang membuat pengguna kesulitan mengelola pengeluaran dalam mata uang asing.

1.3. Tujuan Penelitian

- Membantu pengguna dalam mencatat dan memantau pemasukan serta pengeluaran secara efisien.
- Menyediakan laporan dan grafik yang menggambarkan pengeluaran bulanan untuk mempermudah analisis oleh user.
- Menerapkan trigger untuk memberikan peringatan saat saldo mencapai batas minimum.
- Melakukan scraping data kurs mata uang terkini dari sumber terpercaya untuk memastikan nilai yang akurat.

1.4. Ruang Lingkup

Proyek ini berfokus pada pengembangan sistem informasi berbasis database menggunakan PostgreSQL untuk mendukung pengelolaan keuangan personal secara efisien. Lingkup pekerjaan yang dilakukan meliputi:

- Perancangan basis data dengan desain ERD
- Operasi data dengan query SQL
- Pembuatan view
- Integritas data
- Transaksi dan Atomicity
- Automasi menggunakan trigger
- Stored procedure dan function
- Hak akses atau Authorization
- Implementasi konsep big data

1.5. Tools

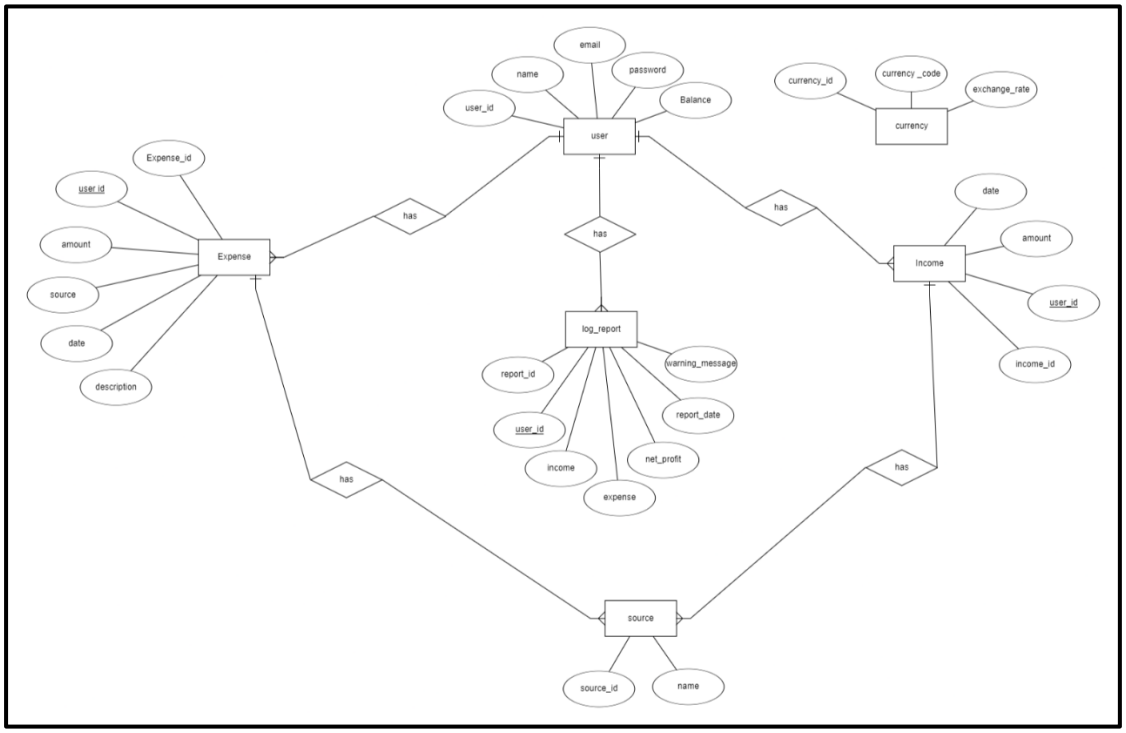
Penggunaan tools pada proyek kami yaitu:

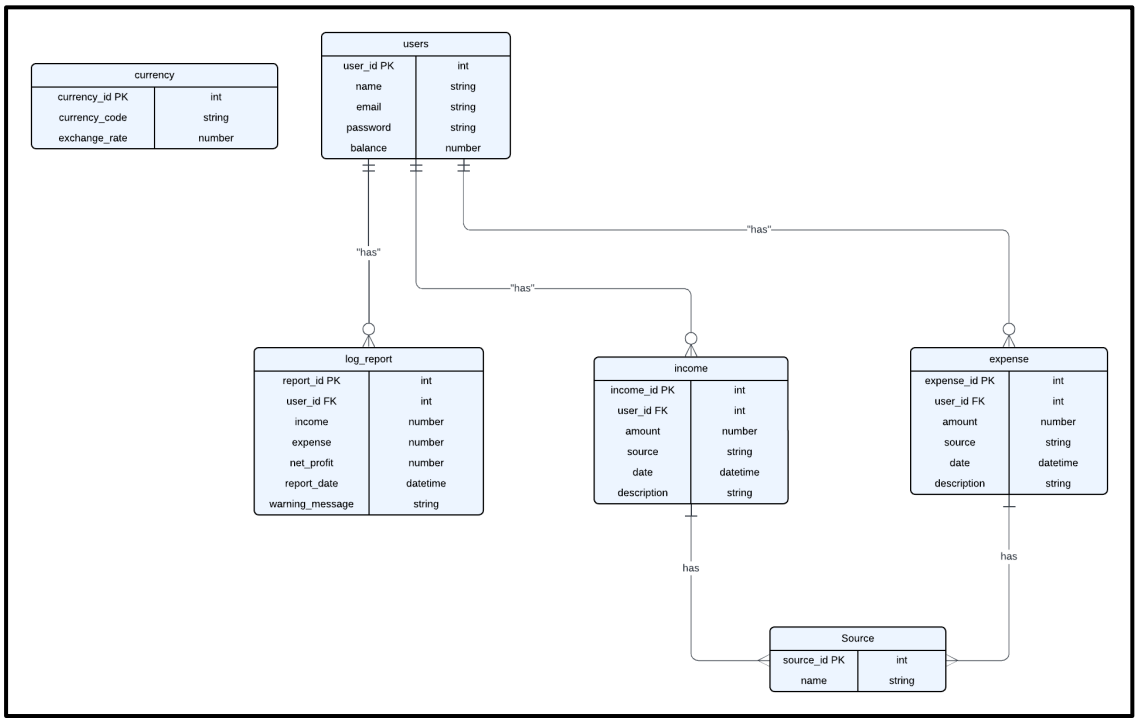
- Trello : untuk membuat workspace
- Googledocs : untuk pembuatan laporan
- Word : untuk membuat laporan
- ERD plus : untuk membuat ERD
- Lucid Chart : untuk membuat ERD

BAB 2 : RANCANGAN BASIS DATA

2.1. Entity-Relationship Diagram (ERD)

ERD di bawah ini menggambarkan struktur utama database sistem manajemen keuangan personal, dimana entity nya Users, Income, Expense, Currency, dan Log Report, Source serta relationship antar entity.





2.2. Sturktur Tabel Database

Berdasarkan ER Diagram tersebut, berikut adalah bentuk tabel-tabel beserta relasinya :

1. User

   Tabel user ini berfungsi sebagai menyimpan mengenai informasi pengguna dalam system . Tabel user memiliki relationship one to many ke tabel income , expense , log_report. Dimana tabel user ini terdiri atas User_id sebagai primary key , name , email, password , dan balance .

2. Income

   Tabel income ini berfungsi untuk menyimpan data pemasukan pengguna kedalam sisitem . Tabel income memiliki relationhip one to many ke tabel user dimana tabel income ini terdiri atas income_id sebagai primary key , user_id sebagai foreign key , amount , source, date , dan description .

3. Expense

   Tabel expense ini berfungsi untuk menyimpan data pengeluaran pengguna kedalam system . tabel expense ini memiliki relationship one to many ke tabel user dimana tabel income ini terdiri atas expense_id sebagai primary key, user_id sebagai foreign key , amount , source , date , description.

4. Log_report

   Tabel Log_report ini berfungsi untuk mencatat ringkasan laporan keuangan pengguna, ermasuk pendapatan, pengeluaran, laba bersih, waktu laporan, dan pesan peringatan untuk analisis keuangan. Table Log_report ini memiliki relationship one to many ke table user dimana terdiri atas report_id sebagai primary key , user_id sebagai foreign key , income, expense , net_profit, report_date, warning_message.

5. Source

   Table source ini berfungsi untuk menyimpan referensi sumber pendapatan dan pengeluaran pengguna. Table source ini memiliki relationship one to many ke table Income dan Expense dimana table source ini terdiri atas source_id sebagai primary key , dan name .

6. Currency

   Tabel currency ini berfungsi untuk mengelola data mata uang dan nilai tukarnya.

   Table ini tidak memiliki relationship ke table manapun atau biasa di sebut dengan table independent . table currency ini terdiri atas currency_id sebagai primary key , currency_code , dan exchange_rate .

2.3. Diagram Relasi Antar Entitas

Berdasarkan diagram ERD, berikut adalah relasi antar entitas dalam sistem:

- User mengelola Income dan Expense, mencatat informasi terkait jumlah, sumber, tanggal, dan deskripsi transaksi.
- Income dan Expense terhubung dengan Source, mencatat asal pendapatan dan tujuan pengeluaran.
- Log_Report terhubung dengan User untuk merangkum aktivitas keuangan pengguna, mencatat total pendapatan, total pengeluaran, laba bersih, dan peringatan masalah keuangan.
- Currency menyimpan informasi tentang mata uang yang digunakan dalam transaksi, mendukung konversi mata uang dalam transaksi pendapatan dan pengeluaran.

BAB 3 : IMPLEMENTASI

3.1. Create Database

```
1   CREATE DATABASE financial_project;
```

Di bagian ini kami membuat database dengan nama financial_project.

3.2. Create Table

3.2.1. Create Table Users

```
3  CREATE TABLE users(
4      user_id SERIAL PRIMARY KEY,
5      name VARCHAR(255) NOT NULL,
6      email VARCHAR(255) NOT NULL UNIQUE,
7      password VARCHAR(255) NOT NULL,
8      balance NUMERIC(12, 2) CHECK (balance >= 0),
9      minimum_balance NUMERIC(12, 2) DEFAULT 50000
10 );
```

query ini untuk membuat tabel users yang berfungsi menyimpan informasi pengguna.

3.2.2. Membuat tabel income

```
20 CREATE TABLE income(
21     income_id SERIAL PRIMARY KEY,
22     user_id INTEGER REFERENCES users(user_id) ON DELETE CASCADE,
23     amount NUMERIC(12, 2) NOT NULL,
24     source INTEGER REFERENCES source(source_id) ON DELETE CASCADE,
25     date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
26     description VARCHAR(255) NOT NULL
27 );
```

Query ini untuk membuat tabel income yang menyimpan data pendapatan pengguna, termasuk jumlah, sumber, tanggal, dan deskripsi, dengan relasi ke tabel users.

3.2.3. Membuat table expense

```
29 CREATE TABLE expense(
30     expense_id SERIAL PRIMARY KEY,
31     user_id INTEGER REFERENCES users(user_id) ON DELETE CASCADE,
32     amount NUMERIC(12, 2) NOT NULL,
33     source INTEGER REFERENCES source(source_id) ON DELETE CASCADE,
34     date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
35     description VARCHAR(255) NOT NULL
36 );
```

Query ini untuk membuat tabel expense yang meyimpan data pengeluaran pengguna , termasuk jumlah , sumber, tanggal, deskripsi demgan relasi ke tanel users.

3.2.4. Membuat tabel currency

```
38 CREATE TABLE currency (
39     currency_id SERIAL PRIMARY KEY,
40     currency_code VARCHAR(10) NOT NULL,
41     exchange_rate DECIMAL(15, 4) NOT NULL
42 );
```

Query ini untuk mencatat data mata uang, termasuk kode mata uang dan nilai tukarnya terhadap mata uang tertentu, dengan kolom unik untuk identifikasi.

3.2.5. Membuat table Log_report

```
44 CREATE TABLE log_report (
45     report_id SERIAL PRIMARY KEY,
46     user_id INTEGER REFERENCES users(user_id),
47     income NUMERIC(12, 2) NOT NULL,
48     expense NUMERIC(12, 2) NOT NULL,
49     net_profit NUMERIC(12, 2) GENERATED ALWAYS AS (income - expense) STORED,
50     report_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
51     warning_message TEXT
52 );
```

Query ini digunakan untuk mencatat ringkasan laporan keuangan pengguna, termasuk total pendapatan, total pengeluaran, laba bersih (dihitung otomatis), tanggal laporan, dan pesan peringatan

3.2.6. Membuat tabel Source

```
54  CREATE TABLE source (
55      source_id INT PRIMARY KEY,
56      name VARCHAR(255) NOT NULL
57  );
```

Query ini digunakan untuk menyimpan daftar sumber pendapatan atau pengeluaran dengan ID unik dan nama sumber.

3.3. Penyisipan Dummy Data

3.3.1. Insert dummy data ke table users

```
12  -- Insert data ke tabel users
13  INSERT INTO users(name, email, password, balance) VALUES
14      ('Vanesa', 'vanesa@gmail.com', 'vanesa123', 100000),
15      ('James', 'james@gmail.com', 'james123', 100000),
16      ('Arnold', 'arnold@gmail.com', 'arnold123', 100000),
17      ('Jessica', 'jessica@gmail.com', 'jessica123', 100000);
18
19  select * from users;
```

Query ini digunakan untuk menambahkan data pengguna baru ke dalam tabel users, yang terdiri dari empat pengguna: Vanesa, James, Arnold, dan Jessica. Setiap pengguna memiliki informasi berupa nama, email, kata sandi, dan saldo awal sebesar 100.000. Setelah data ditambahkan, query SELECT * FROM users digunakan untuk menampilkan seluruh data dalam tabel users untuk memastikan proses penambahan data telah berhasil.

3.3.2. Insert dummy data ke tabel source

```
283 INSERT INTO source (source_id, name) VALUES
284 (1, 'Gaji'),
285 (2, 'Trading'),
286 (3, 'Freelance'),
287 (4, 'Usaha'),
288 (5, 'Proyek'),
289 (6, 'Investasi'),
290 (7, 'Affiliate'),
291 (8, 'Lembur'),
292 (9, 'Dividen'),
293 (10, 'Transportasi'),
294 (11, 'Tagihan'),
295 (12, 'Belanja'),
296 (13, 'Hiburan'),
297 (14, 'Makanan'),
298 (15, 'Jual Barang'),
299 (16, 'Event'),
300 (17, 'Jasa'),
301 (18, 'Dropship'),
302 (19, 'Royalti'),
303 (20, 'Sewa Properti'),
304 (21, 'Hadiah'),
305 (22, 'Pembuatan Website');
```

Query ini digunakan untuk menambahkan data pada tabel source yang gunanya untuk menyimpan referensi sumber pemasukan dan pengeluaran pengguna .

3.3.3. Insert dummy data ke tabel income

```sql
INSERT INTO income(user_id, amount, source, description, date) VALUES
(4, 5000000, 1, 'Gajian bulan Desember', '2023-11-01 09:00:00'),
(1, 250000, 2, 'Hasil dari trading', '2023-11-02 15:30:00'),
(1, 150000, 3, 'Proyek Freelance', '2023-10-03 14:15:00'),
(3, 400000, 4, 'Penjualan Produk', '2023-12-04 11:45:00'),
(2, 500000, 5, 'Pendapatan Proyek', '2023-08-01 13:20:00'),
(2, 160000, 6, 'Keuntungan investasi', '2023-10-02 10:50:00'),
(3, 120000, 7, 'Komisi penjualan online', '2023-02-03 16:30:00'),
(4, 110000, 8, 'Bonus lembur', '2023-03-04 19:00:00'),
(2, 115000, 9, 'Dividen saham', '2023-02-02 09:10:00'),
(3, 450000, 2, 'Hasil dari trading', '2023-01-03 20:00:00'),
(1, 105000, 6, 'keuntungan investasi', '2023-03-01 18:20:00'),
(3, 400000, 2, 'Hasil dari trading', '2023-04-02 12:10:00'),
(2, 3000000, 1, 'Gaji Bulanan', '2023-05-04 09:40:00'),
(4, 200000, 6, 'keuntungan investasi', '2023-06-01 14:50:00'),
(2, 100000, 5, 'Pendapatan Proyek', '2023-05-02 15:25:00'),
(3, 300000, 5, 'Pendapatan Proyek', '2023-06-03 10:40:00'),
(4, 200000, 8, 'Bonus lembur', '2023-04-04 16:00:00'),
(1, 310000, 6, 'Keuntungan investasi', '2023-02-01 11:30:00'),
(3, 4000000, 1, 'Gaji Bulanan', '2023-10-02 17:00:00'),
(2, 140000, 9, 'Diveden Saham', '2023-05-03 08:50:00'),
(4, 320000, 7, 'Komisi penjualan online', '2023-07-04 14:10:00'),
(1, 200000, 3, 'Proyek desain grafis', '2023-07-05 10:20:00'),
(2, 500000, 4, 'Hasil penjualan online', '2023-09-01 12:30:00'),
(3, 250000, 5, 'Pendapatan proyek jangka pendek', '2023-11-15 14:50:00'),
(4, 150000, 8, 'Bonus lembur tambahan', '2023-03-12 19:45:00'),
(1, 300000, 6, 'Hasil investasi saham', '2023-01-05 09:15:00'),
(2, 600000, 1, 'Gaji Bulanan', '2023-04-01 09:00:00'),
(3, 200000, 2, 'Hasil trading forex', '2023-06-05 16:30:00'),
(4, 350000, 7, 'Komisi dari penjualan', '2023-08-10 10:40:00'),
(1, 120000, 5, 'Pendapatan proyek freelance', '2023-09-01 13:20:00'),
(2, 450000, 6, 'Keuntungan dari properti', '2023-12-01 11:50:00'),
(3, 800000, 4, 'Pendapatan usaha online', '2023-01-01 14:00:00'),
(4, 300000, 1, 'Gaji tambahan bulan ini', '2023-02-01 09:30:00'),
(1, 70000, 8, 'Bonus lembur malam', '2023-03-10 19:20:00'),
(2, 160000, 9, 'Hasil investasi reksa dana', '2023-04-15 08:30:00'),
(3, 300000, 2, 'Keuntungan trading crypto', '2023-05-20 18:50:00'),
(4, 250000, 7, 'Komisi produk digital', '2023-06-01 10:00:00'),
(1, 50000, 5, 'Pendapatan proyek kecil', '2023-07-01 11:20:00'),
(2, 200000, 3, 'Proyek pengembangan web', '2023-08-05 13:15:00'),
(3, 100000, 6, 'Hasil investasi emas', '2023-09-10 14:40:00'),
(4, 550000, 1, 'Gaji Bulanan', '2023-10-01 09:50:00'),
(1, 180000, 5, 'Pendapatan proyek penulisan', '2023-11-01 15:10:00'),
(2, 420000, 4, 'Hasil penjualan produk', '2023-11-15 16:20:00'),
(3, 750000, 1, 'Gaji tambahan', '2023-12-01 09:40:00'),
(4, 300000, 7, 'Komisi dari toko online', '2023-12-05 14:30:00'),
(1, 400000, 3, 'Pendapatan desain logo', '2023-12-04 13:20:00'),
(2, 120000, 2, 'Keuntungan trading kecil', '2023-08-01 10:00:00'),
(3, 200000, 6, 'Hasil investasi kecil', '2023-01-20 15:45:00'),
(4, 150000, 5, 'Pendapatan proyek desain', '2023-02-15 18:30:00'),
(1, 300000, 3, 'Proyek aplikasi mobile', '2023-03-20 16:20:00'),
(2, 400000, 1, 'Pendapatan bulanan', '2023-05-01 09:00:00');
```

Query ini digunakan untuk menambahkan data pada tabel income yang gunanya untuk menyimpan data pemasukan pengguna ke dalam sistem .

3.3.4. Insert dummy data ke *tabel* expense

```sql
INSERT INTO expense(user_id, amount, source, description, date) VALUES
(1, 250000, 10, 'Naik taksi', '2023-12-03 08:40:02'),
(1, 40000, 11, 'Tagihan Listrik', '2023-09-01 10:15:20'),
(3, 100000, 12, 'Belanja Bulanan', '2023-10-02 14:20:35'),
(2, 75000, 13, 'Tiket Bioskop', '2023-03-03 18:50:00'),
(3, 120000, 14, 'Makan Siang', '2023-03-04 12:30:25'),
(2, 85000, 7, 'Komisi penjualan online', '2023-10-30 09:00:10'),
(4, 25000, 15, 'Penjualan laptop bekas', '2023-06-02 17:45:50'),
(2, 60000, 19, 'Royalti dari buku yang diterbitkan', '2023-04-01 11:25:40'),
(4, 250000, 4, 'Keuntungan dari penjualan makanan ringan', '2023-03-03 19:20:15'),
(1, 50000, 17, 'Penghasilan dari jasa penulisan artikel', '2023-02-04 08:10:05'),
(2, 90000, 18, 'Keuntungan dari dropship produk kecantikan', '2023-10-03 15:25:35'),
(4, 250000, 1, 'Pendapatan kerja part-time', '2023-12-02 10:40:50'),
(1, 300000, 16, 'Pendapatan dari event organizer', '2023-01-30 14:50:30'),
(1, 45000, 20, 'Hadiah dari lomba coding', '2023-02-01 20:15:20'),
(2, 65000, 6, 'Keuntungan investasi emas', '2023-03-02 16:35:25'),
(4, 180000, 3, 'Proyek desain logo perusahaan', '2023-04-03 13:40:45'),
(2, 150000, 21, 'Pendapatan dari menyewakan rumah', '2023-12-04 09:30:10'),
(3, 70000, 15, 'Penjualan ponsel bekas', '2023-05-02 18:50:00'),
(2, 275000, 1, 'Pendapatan dari pekerjaan kontrak', '2023-06-01 08:25:15'),
(3, 125000, 2, 'Keuntungan dari trading forex', '2023-07-03 07:50:20'),
(4, 100000, 22, 'Penghasilan dari membuat website perusahaan', '2023-08-04 11:00:30'),
(1, 30000, 14, 'Sarapan pagi', '2023-01-05 07:45:00'),
(2, 45000, 10, 'Naik ojek online', '2023-01-10 09:20:00'),
(3, 125000, 12, 'Belanja pakaian', '2023-01-15 13:30:00'),
(4, 65000, 13, 'Beli tiket konser', '2023-01-20 19:15:00'),
(1, 40000, 11, 'Tagihan air', '2023-02-05 08:10:00'),
(2, 90000, 3, 'Desain kartu nama', '2023-02-15 14:25:00'),
(3, 200000, 6, 'Keuntungan reksa dana', '2023-03-01 09:30:00'),
(4, 30000, 14, 'Beli cemilan', '2023-03-05 15:50:00'),
(1, 75000, 15, 'Penjualan kamera bekas', '2023-03-10 10:15:00'),
(2, 40000, 13, 'Beli buku', '2023-03-15 12:45:00'),
(3, 150000, 4, 'Hasil penjualan online shop', '2023-04-10 18:20:00'),
(4, 120000, 2, 'Keuntungan trading saham', '2023-04-15 08:10:00'),
(1, 50000, 10, 'Naik bus antar kota', '2023-05-05 06:50:00'),
(2, 80000, 11, 'Tagihan internet', '2023-05-10 11:00:00'),
(3, 90000, 16, 'Honor sebagai pembicara', '2023-06-20 14:40:00'),
(4, 60000, 14, 'Makan malam di restoran', '2023-07-05 20:15:00'),
(1, 70000, 18, 'Keuntungan penjualan produk', '2023-07-15 16:30:00'),
(2, 140000, 1, 'Pendapatan kerja part-time', '2023-08-01 09:20:00'),
(3, 115000, 7, 'Komisi dari promosi produk', '2023-09-10 10:50:00'),
(4, 200000, 3, 'Proyek editing video', '2023-10-01 13:15:00'),
(1, 90000, 17, 'Penghasilan sebagai fotografer', '2023-10-10 08:00:00'),
(2, 55000, 12, 'Belanja bahan makanan', '2023-10-20 17:30:00'),
(3, 100000, 11, 'Tagihan listrik dan air', '2023-11-05 09:10:00'),
(4, 85000, 4, 'Keuntungan dari penjualan makanan ringan', '2023-11-15 12:00:00'),
(1, 110000, 16, 'Pendapatan dari acara komunitas', '2023-12-01 18:50:00'),
(2, 60000, 10, 'Biaya parkir bulanan', '2023-12-03 07:30:00'),
(3, 150000, 2, 'Hasil investasi crypto', '2023-12-04 20:00:00');
```

Query ini digunakan untuk menambahkan dummy data ke tabel expense yang gunanya untuk menyimpan data pengeluaran pengguna kedalam system.

3.4.Pembuatan Prosedur dan fungsi

3.4.1. Fungsi add_balance

```
117  -- Add Balance
118  -- Fungsi untuk menambah saldo pengguna setiap ada pemasukan
119  CREATE OR REPLACE FUNCTION add_balance()
120  RETURNS TRIGGER AS $$
121  BEGIN
122      UPDATE users
123      SET balance = balance + NEW.amount
124      WHERE user_id = NEW.user_id;
125
126      RETURN NEW;
127  END;
128  $$ LANGUAGE plpgsql;
```

Fungsi ini digunakan untuk menambahkan saldo pengguna di tabel users setiap kali ada data pendapatan baru yang dimasukkan ke tabel income. Fungsi bekerja dengan menambahkan nilai amount dari data baru (NEW.amount) ke kolom balance pengguna (users.balance) yang sesuai dengan user_id

3.4.2. Fungsi balance_warning

```
136  -- Minimum Balance Warning
137  -- Fungsi untuk memberikan peringatan jika saldo di bawah batas minimum
138  CREATE OR REPLACE FUNCTION balance_warning()
139  RETURNS TRIGGER AS $$
140  BEGIN
141      IF (SELECT balance FROM users WHERE user_id = NEW.user_id) < 50000 THEN
142          RAISE NOTICE 'Peringatan: Saldo pengguna % sudah di bawah saldo minimum (50000).', NEW.user_id;
143      END IF;
144
145      RETURN NULL;
146  END;
147  $$ LANGUAGE plpgsql;
```

Fungsi ini digunakan untuk memberikan peringatan jika saldo pengguna di bawah batas minimum (50,000) setelah pengguna menambahkan pengeluaran baru di tabel expense. Fungsi ini akan memeriksa saldo pengguna di tabel users berdasarkan user_id dari data baru (NEW.user_id). Jika saldo kurang dari 50,000, fungsi akan memberikan peringatan menggunakan RAISE NOTICE.

3.4.3. Fungsi total income_expense_for_user

```sql
-- Total Income & Expense for User
-- Fungsi untuk menghitung total pemasukan dan pengeluaran
CREATE OR REPLACE FUNCTION total_income_expense_for_user(p_user_id INTEGER)
RETURNS TABLE(total_income NUMERIC(12, 2), total_expense NUMERIC(12, 2))
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT COALESCE(SUM(amount), 0) INTO total_income
    FROM income
    WHERE user_id = p_user_id;

    SELECT COALESCE(SUM(amount), 0) INTO total_expense
    FROM expense
    WHERE user_id = p_user_id;

    RETURN NEXT;
END;
$$;
```

Fungsi ini digunakan untuk menghitung dan mengembalikan total pendapatan dan total pengeluaran pengguna berdasarkan user_id. Fungsi ini akan menerima parameter p_user_id untuk menentukan pengguna yang akan dihitung. Kemudian, fungsi menjumlahkan nilai amount dari tabel income untuk menghitung total pendapatan dan dari tabel expense untuk menghitung total pengeluaran. Hasilnya dikembalikan dalam format tabel dengan dua kolom: total_income dan total_expense.

3.4.4. Fungsi insert_log_report

```sql
-- Insert Log Report
-- Fungsi untuk memasukkan data ke tabel log_report
CREATE OR REPLACE FUNCTION insert_log_report()
RETURNS VOID AS $$
BEGIN
    DELETE FROM log_report;

    INSERT INTO log_report (user_id, income, expense)
    SELECT
        u.user_id,
        COALESCE(SUM(i.amount), 0) AS income,
        COALESCE(SUM(e.amount), 0) AS expense
    FROM
        users u
    LEFT JOIN
        income i ON u.user_id = i.user_id
    LEFT JOIN
        expense e ON u.user_id = e.user_id
    GROUP BY
        u.user_id;

    UPDATE log_report
    SET warning_message = 'Peringatan: Pengeluaran lebih besar dari pemasukan.'
    WHERE net_profit < 0;
END;
$$ LANGUAGE plpgsql;

-- Pemanggilan fungsi
SELECT * FROM insert_log_report();
```

Fungsi ini bertujuan untuk memperbarui tabel log_report dengan ringkasan pendapatan, pengeluaran, laba bersih, dan memberikan pesan peringatan jika pengeluaran lebih besar daripada pendapatan. Fungsi menghapus data lama di tabel log_report, lalu menghitung total pendapatan dan pengeluaran setiap pengguna menggunakan query dengan LEFT JOIN. Nilai yang dihitung dimasukkan ke tabel log_report, dan pesan peringatan ditambahkan untuk pengguna dengan laba bersih yang negatif.

3.4.5. Fungsi show_exchange_rate

```
-- Fungsi untuk menampilkan saldo dan konversi ke mata uang lain
CREATE OR REPLACE FUNCTION show_exchange_rate(user_num INT)
RETURNS VOID AS $$
DECLARE
    user_balance NUMERIC(12, 2);
    exchange_value NUMERIC(15, 2);
    exchange_code VARCHAR(10);
BEGIN
    -- Mendapatkan balance dari tabel users berdasarkan user_id
    SELECT balance INTO user_balance
    FROM users
    WHERE user_id = user_num;

    -- Mengecek jika user tidak ditemukan
    IF user_balance IS NULL THEN
        RAISE NOTICE 'User with ID % not found.', user_num;
        RETURN;
    END IF;

    -- Menampilkan balance user
    RAISE NOTICE 'User balance: Rp.%', user_balance;

    -- Mendapatkan exchange_rate dan currency_code dari tabel currency
    FOR exchange_code, exchange_value IN
        SELECT currency_code, exchange_rate FROM currency
    LOOP
        -- Menampilkan hasil pembagian antara balance dan exchange_rate
        RAISE NOTICE 'For currency %, the exchange rate is %. Balance divided by exchange rate: %.',
            exchange_code,
            exchange_value,
            ROUND(user_balance / exchange_value, 3);
    END LOOP;
END;
$$ LANGUAGE plpgsql;

-- Pemanggilan fungsi untuk menampilkan saldo dan konversi mata uang
DO $$
BEGIN
    PERFORM show_exchange_rate(1);
END;
$$;
```

Fungsi ini digunakan untuk menampilkan saldo pengguna berdasarkan user_id dalam mata uang lokal (Rupiah) serta melakukan konversi saldo tersebut ke berbagai mata uang lain menggunakan nilai tukar dari tabel currency. Fungsi menerima parameter user_num (ID pengguna) untuk mendapatkan saldo (balance) dari tabel users. Jika saldo tidak ditemukan, fungsi memberikan peringatan bahwa pengguna tidak ada. Setelah itu, saldo ditampilkan dalam Rupiah. Fungsi kemudian mengambil nilai tukar (exchange_rate) dan kode mata uang (currency_code) dari tabel currency, lalu menghitung dan menampilkan saldo pengguna yang telah dikonversi ke setiap mata uang, dengan hasil dibulatkan hingga tiga desimal.

3.5.Pembuatan View

3.5.1. View untuk menampilkan saldo pengguna

```
-- Create View untuk menampilkan saldo pengguna
CREATE VIEW monthly_financial_summary AS
SELECT
    user_id,
    DATE_TRUNC('month', date) AS bulan,
    'expense' AS kategori,
    SUM(amount) AS total
FROM expense
GROUP BY user_id, DATE_TRUNC('month', date)

UNION ALL

SELECT
    user_id,
    DATE_TRUNC('month', date) AS bulan,
    'income' AS kategori,
    SUM(amount) AS total
FROM income
GROUP BY user_id, DATE_TRUNC('month', date)

ORDER BY user_id, bulan, kategori;
```

View ini dibuat untuk menampilkan ringkasan keuangan bulanan setiap pengguna, mencakup total pengeluaran (expense) dan pendapatan (income) yang dikelompokkan berdasarkan bulan dan jenis kategori. View ini menggabungkan data dari tabel expense dan income menggunakan UNION ALL. Data dari kedua tabel tersebut dikelompokkan berdasarkan user_id dan bulan transaksi (diperoleh melalui fungsi DATE_TRUNC('month', date)). Setiap kategori diberi label sebagai 'expense' atau 'income', lalu jumlah total (amount) dihitung menggunakan fungsi SUM. Hasilnya diurutkan berdasarkan user_id, bulan, dan kategori.

3.6. Authorization

3.6.1. Authorization untuk Pengguna dengan Akses Terbatas

```
-- Membuat pengguna
CREATE USER "1";
CREATE USER "2";
CREATE USER "3";
CREATE USER "4";

-- Mengaktifkan Row Level Security
ALTER TABLE users ENABLE ROW LEVEL SECURITY;

-- Membuat kebijakan agar pengguna hanya bisa mengakses data yang sesuai dengan user_id mereka
CREATE POLICY user_see_own_data ON users
    FOR SELECT
    USING (user_id = CAST(current_user AS INT));

-- Menghapus kebijakan jika perlu
-- DROP POLICY user_see_own_data ON users;

-- Memberikan akses SELECT kepada pengguna "1"
GRANT SELECT ON users TO "1";

-- Menggunakan peran pengguna "1"
SET ROLE "1";

-- Melakukan SELECT untuk memastikan hanya data dengan user_id sesuai yang muncul
SELECT * FROM users;

-- Kembalikan role ke admin
SET ROLE postgres;
```

Query ini dibuat untuk membatasi akses data di tabel users agar setiap pengguna hanya bisa melihat data sesuai dengan user_id mereka. Pertama, dibuat beberapa pengguna, lalu diaktifkan fitur Row Level Security (RLS) di tabel users.
Kebijakan dibuat untuk memastikan pengguna hanya dapat mengakses baris data dengan user_id yang sama dengan nama pengguna mereka. Setelah itu, hak akses diberikan ke pengguna tertentu, dan role diubah untuk menguji apakah kebijakan berjalan sesuai harapan. Kebijakan ini menjaga privasi data antar pengguna.

3.7.Trigger

3.7.1. Trigger_minimum_balance_warning_expense

```
144  -- Trigger untuk memberikan peringatan jika saldo di bawah minimum setelah pengeluaran
145  CREATE TRIGGER trigger_minimum_balance_warning_expense
146  AFTER INSERT ON expense
147  FOR EACH ROW
148  EXECUTE FUNCTION balance_warning();
```

Trigger ini digunakan untuk memanggil fungsi balance_warning setiap kali ada data pengeluaran baru yang dimasukkan ke tabel expense. Trigger akan aktif setelah operasi INSERT dilakukan pada tabel expense. Untuk setiap baris data baru, fungsi balance_warning akan dijalankan secara otomatis untuk memeriksa apakah saldo pengguna berada di bawah batas minimum. Jika saldo berada di bawah batas, fungsi akan memberikan peringatan

3.7.2. Trigger reduce_balance

```
150  -- Trigger untuk mengurangi saldo sebelum memasukkan data ke tabel expense
151  CREATE TRIGGER trigger_reduce_balance
152  BEFORE INSERT ON expense
153  FOR EACH ROW
154  EXECUTE FUNCTION reduce_balance();
155
```

Trigger trigger_reduce_balance berfungsi untuk memastikan bahwa setiap kali data baru dimasukkan ke dalam tabel expense, fungsi reduce_balance() dijalankan terlebih dahulu. Dengan demikian, sebelum pencatatan pengeluaran dilakukan, saldo pengguna akan diperbarui sesuai dengan jumlah pengeluaran yang akan dicatat, memastikan integritas dan konsistensi data keuangan pengguna.

3.7.3. Trigger_add_balance

```
156  -- Trigger untuk menambah saldo sebelum memasukkan data ke tabel income
157  CREATE TRIGGER trigger_add_balance
158  BEFORE INSERT ON income
159  FOR EACH ROW
160  EXECUTE FUNCTION add_balance();
```

Trigger trigger_add_balance memastikan bahwa setiap kali data baru dimasukkan ke dalam tabel income, fungsi add_balance() dijalankan terlebih dahulu. Yang memungkinkan pembaruan saldo pengguna secara otomatis sebelum pencatatan pendapatan baru, sehingga integritas dan konsistensi data keuangan pengguna tetap terjaga.

3.8. Backup Data

```
-- Backup data dan jalankan pada terminal
pg_dump -U postgres -d financial_project -f "12_BackupSistemManajemenKeuanganPersonal.sql"
```

Perintah ini digunakan untuk membuat backup database bernama financial_project di PostgreSQL. Backup dilakukan menggunakan utilitas pg_dump dengan akses sebagai pengguna postgres. Hasil backup disimpan dalam file bernama 12_BackupSistemManajemenKeuanganPersonal.sql. File tersebut akan berisi seluruh data dan struktur tabel dari database, sehingga dapat digunakan jika database perlu dipulihkan di kemudian hari. Backup ini penting untuk mengamankan data dari risiko kehilangan atau kerusakan.

3.9. Restore Data

```
-- Restore data dan jalankan pada terminal
psql -U postgres -d financial_project -f "12_BackupSistemManajemenKeuanganPersonal.sql"
```

Query ini digunakan untuk mengembalikan (restore) data dari file backup ke dalam database financial_project. Utilitas psql digunakan untuk menjalankan perintah SQL, dan dengan opsi -U postgres, proses restore dilakukan dengan akses pengguna postgres. Opsi d financial_project menunjukkan bahwa data akan dipulihkan ke database financial_project. File yang berisi cadangan data, yaitu 12_BackupSistemManajemenKeuanganPersonal.sql, diinput menggunakan opsi -f. Dengan perintah ini, seluruh data yang ada dalam file backup akan dipulihkan dan dimasukkan kembali ke dalam database yang dituju, sehingga database tersebut kembali seperti semula saat cadangan dibuat.

3.10. Ekspor Data

Query ini digunakan untuk mengekspor data dari tabel monthly_financial_summary ke file CSV. Data yang diambil dengan query SELECT * akan disalin ke file financial.csv yang disimpan di lokasi yang ditentukan. Dengan menggunakan opsi DELIMITER ',', setiap kolom dalam file CSV akan dipisahkan dengan koma, sementara CSV HEADER memastikan bahwa nama kolom akan ditambahkan di baris pertama file. Ini memungkinkan data dalam tabel tersebut untuk disimpan dalam format yang lebih mudah dibaca atau digunakan di luar database.

BAB 4 : PENUTUP

4.1. Kesimpulan

Proyek Personal Financial Management System telah berhasil dirancang dan diimplementasikan sesuai dengan tujuan yang ditetapkan. Sistem ini memudahkan pengguna dalam mencatat pemasukan dan pengeluaran, serta menyediakan laporan dan grafik analisis keuangan. Fitur notifikasi saldo minimum membantu pengguna untuk memantau keuangan mereka dengan lebih baik. Selain itu, sistem juga mendukung konversi mata uang dengan data kurs terkini, memudahkan transaksi internasional.

Dengan teknologi berbasis database, sistem ini menyimpan data dengan aman dan memudahkan akses untuk memantau kondisi keuangan. Proyek ini diharapkan menjadi solusi efektif dan efisien dalam pengelolaan keuangan pribadi, membantu pengguna membuat keputusan finansial yang lebih baik dan terorganisir.