Rakamin Academy X id/x partners

# Designing a data warehouse and implementing stored procedures

## ID/X Partners - Data Engineer

Presented by
Sayid Mufaqih

Rakamin Academy

📍 Bangunrejo, Lampung tengah

✉ sayidmufaqih03@gmail.com

in sayidmufaqih

# Sayid Mufaqih

## Teacher

I’m currently a teacher but have passion in building databases and implementing ETL process through various technology.

# Courses and Certification

| | |
|---|---|
| **Google Data Analytics Specialization \| Link** | **January, 2024** |
| **Business Analytics with Excel: Elementary to Advanced \| Link** | **December, 2023** |
| **KPMG-AU Data Analytics Job Simulation \| Link** | **November, 2023** |
| **Programming for Everybody (Getting started with python) \| Link** | **September, 2023** |

# About Company

ID/X Partners, a company that collaborates with Rakamin Academy, offers virtual internship experiences in the fields of data engineering, data science, and software engineering. Their data engineering internship program allows participants to build and manage data warehouses for e-commerce clients, thereby improving problem-solving skills and the use of SQL in data processing

# Project Portfolio

## Background :

- One of the clients of ID/X Partners company which operates in the banking industry, has a need to create a *Data Warehouse* from several different data sources stored in their system.

## Data :

- sample.bak
- Transaction_exel.xlsx
- Transaction_csv.csv

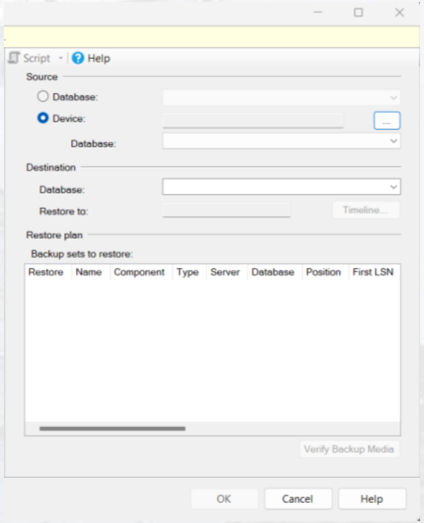
## Problem Statement :

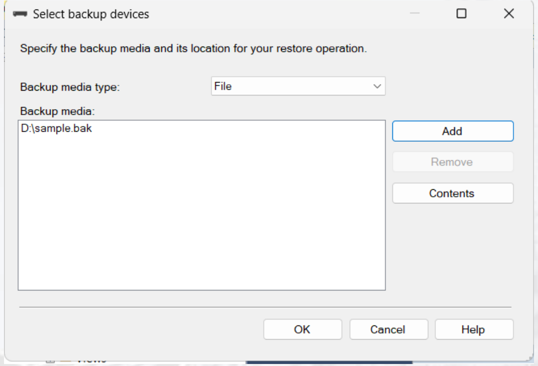
- Difficulty in extracting data from various sources (excel, csv, database) simultaneously so that reporting and data analysis always experience delays

**Project explanation video here!**

# 1. Data Warehouse Creation

## 1. Restore database ‘sample.bak in SSMS’

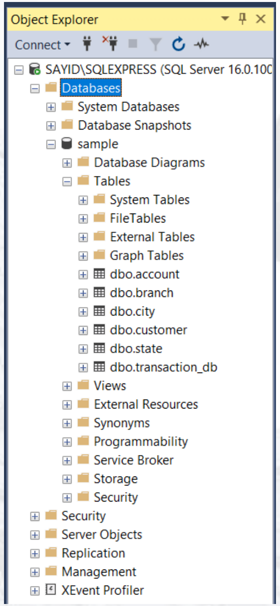



Select backup devices

Specify the backup media and its location for your restore operation.

Backup media type: File

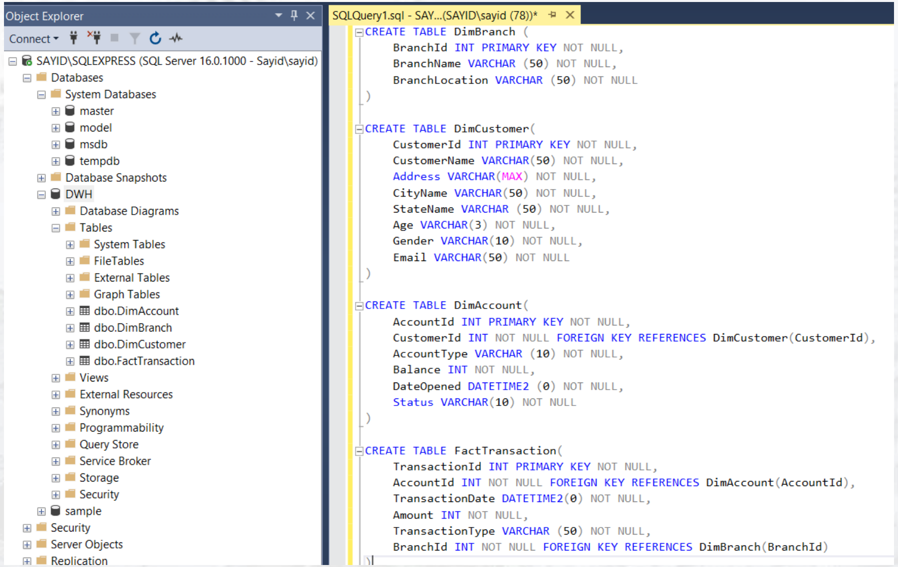
Backup media:
D:\sample.bak


Object Explorer

```
⊟ SAYID\SQLEXPRESS (SQL Server 16.0.100
  ⊟ Databases
    ⊞ System Databases
    ⊞ Database Snapshots
    ⊟ sample
      ⊞ Database Diagrams
      ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊞ dbo.account
        ⊞ dbo.branch
        ⊞ dbo.city
        ⊞ dbo.customer
        ⊞ dbo.state
        ⊞ dbo.transaction_db
      ⊞ Views
      ⊞ External Resources
      ⊞ Synonyms
      ⊞ Programmability
      ⊞ Service Broker
      ⊞ Storage
      ⊞ Security
  ⊞ Security
  ⊞ Server Objects
  ⊞ Replication
  ⊞ Management
  ⊞ XEvent Profiler
```

Database ‘sample’ was sucessfully restored

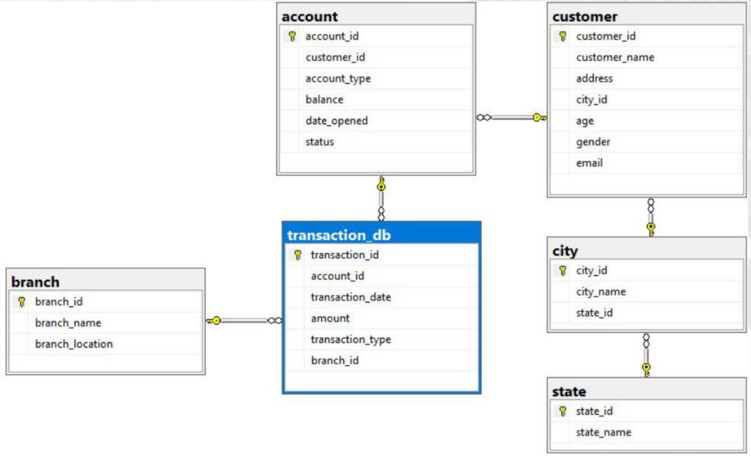
# 2. Create new database named DWH as *Data Warehouse*



```sql
CREATE TABLE DimBranch (
    BranchId INT PRIMARY KEY NOT NULL,
    BranchName VARCHAR (50) NOT NULL,
    BranchLocation VARCHAR (50) NOT NULL
)

CREATE TABLE DimCustomer(
    CustomerId INT PRIMARY KEY NOT NULL,
    CustomerName VARCHAR(50) NOT NULL,
    Address VARCHAR(MAX) NOT NULL,
    CityName VARCHAR(50) NOT NULL,
    StateName VARCHAR (50) NOT NULL,
    Age VARCHAR(3) NOT NULL,
    Gender VARCHAR(10) NOT NULL,
    Email VARCHAR(50) NOT NULL
)

CREATE TABLE DimAccount(
    AccountId INT PRIMARY KEY NOT NULL,
    CustomerId INT NOT NULL FOREIGN KEY REFERENCES DimCustomer(CustomerId),
    AccountType VARCHAR (10) NOT NULL,
    Balance INT NOT NULL,
    DateOpened DATETIME2 (0) NOT NULL,
    Status VARCHAR(10) NOT NULL
)

CREATE TABLE FactTransaction(
    TransactionId INT PRIMARY KEY NOT NULL,
    AccountId INT NOT NULL FOREIGN KEY REFERENCES DimAccount(AccountId),
    TransactionDate DATETIME2(0) NOT NULL,
    Amount INT NOT NULL,
    TransactionType VARCHAR (50) NOT NULL,
    BranchId INT NOT NULL FOREIGN KEY REFERENCES DimBranch(BranchId)
)
```

Create three dimention table :
**DimAccount, DimCustomer, DimBranch** and
one fact table:
**FactTransaction**

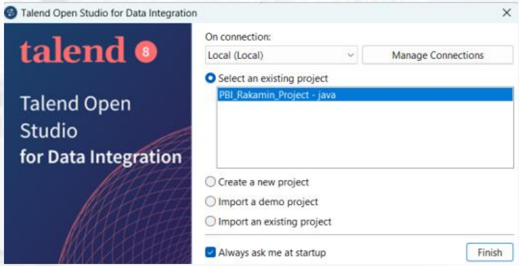
Scheme in DWH Database



Rakamin Academy

# 2. Connect and add data to Talend

## 1. Make project in Talend


Talend Open Studio for Data Integration

talend 8

Talend Open Studio for Data Integration

On connection:

Local (Local) | Manage Connections

● Select an existing project

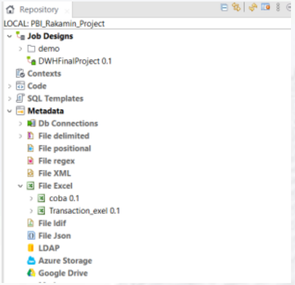
PBI_Rakamin_Project - java

○ Create a new project

○ Import a demo project

○ Import an existing project

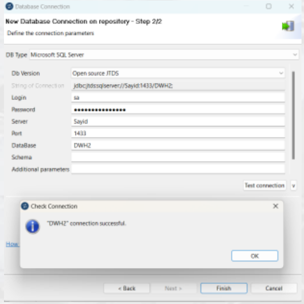
☑ Always ask me at startup


Repository

LOCAL: PBI_Rakamin_Project

- Job Designs
  - demo
  - DWHFinalProject 0.1
- Contexts
- Code
- SQL Templates
- Metadata
  - Db Connections
  - File delimited
  - File positional
  - File regex
  - File XML
  - File Excel
    - coba 0.1
    - Transaction_exel 0.1
  - File ldif
  - File Json
  - LDAP
  - Azure Storage
  - Google Drive

## 2. Make connection for database sample and DWH


Database Connection

New Database Connection on repository - Step 2/2

Define the connection parameters

DB Type: Microsoft SQL Server

Db Version: Open source JTDS

String of Connection: jdbc:jtds:sqlserver://Sayid:1433/DWH2;

Login: sa

Password: •••••••••••••••

Server: Sayid

Port: 1433

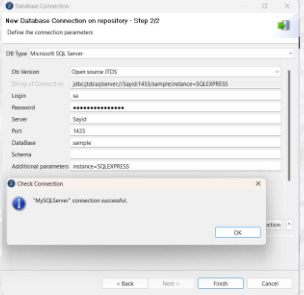
DataBase: DWH2

Schema:

Additional parameters:

Check Connection

"DWH2" connection successful.


Database Connection

New Database Connection on repository - Step 2/2

Define the connection parameters

DB Type: Microsoft SQL Server

Db Version: Open source JTDS

String of Connection: jdbc:jtds:sqlserver://Sayid:1433/sample;instance=SQLEXPRESS

Login: sa

Password: •••••••••••••••

Server: Sayid

Port: 1433

DataBase: sample

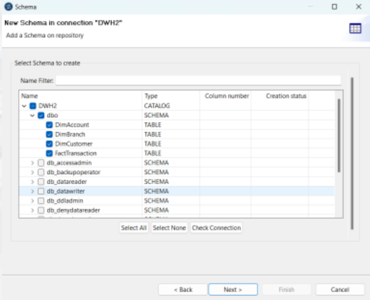
Schema:

Additional parameters: instance=SQLEXPRESS

Check Connection

"MySQLServer" connection successful.

## 3. Make schema in connection and set the table name, data type, etc. as appropriate


Schema

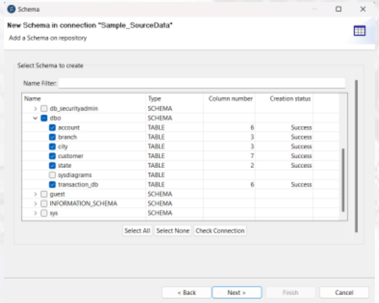
New Schema in connection "DWH2"

Add a Schema on repository

Select Schema to create

Name Filter:

| Name | Type | Column number | Creation status |
|---|---|---|---|
| ■ DWH2 | CATALOG | | |
| ☑ dbo | SCHEMA | | |
| ☑ DimAccount | TABLE | | |
| ☑ DimBranch | TABLE | | |
| ☑ DimCustomer | TABLE | | |
| ☑ FactTransaction | TABLE | | |
| ☐ db_accessadmin | SCHEMA | | |
| ☐ db_backupoperator | SCHEMA | | |
| ☐ db_datareader | SCHEMA | | |
| ☐ db_datawriter | SCHEMA | | |
| ☐ db_ddladmin | SCHEMA | | |
| ☐ db_denydatareader | SCHEMA | | |


Schema

New Schema in connection "Sample_SourceData"

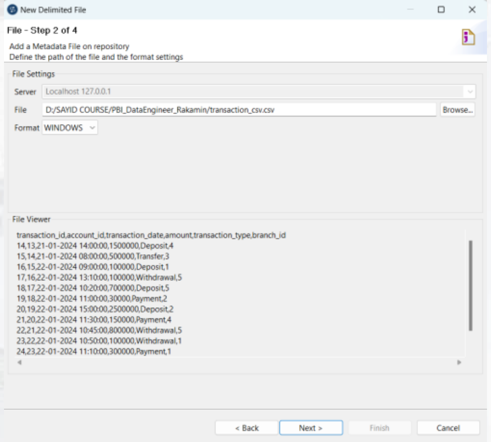
Add a Schema on repository

Select Schema to create

Name Filter:

| Name | Type | Column number | Creation status |
|---|---|---|---|
| ☐ db_securityadmin | SCHEMA | | |
| ■ dbo | SCHEMA | | |
| ☑ account | TABLE | 6 | Success |
| ☑ branch | TABLE | 3 | Success |
| ☑ city | TABLE | 3 | Success |
| ☑ customer | TABLE | 7 | Success |
| ☑ state | TABLE | 2 | Success |
| ☐ sysdiagrams | TABLE | | |
| ☑ transaction_db | TABLE | 6 | Success |
| ☐ guest | SCHEMA | | |
| ☐ INFORMATION_SCHEMA | SCHEMA | | |
| ☐ sys | SCHEMA | | |

# 4. Add a metadata file from Excel and CSV



File - Step 2 of 4

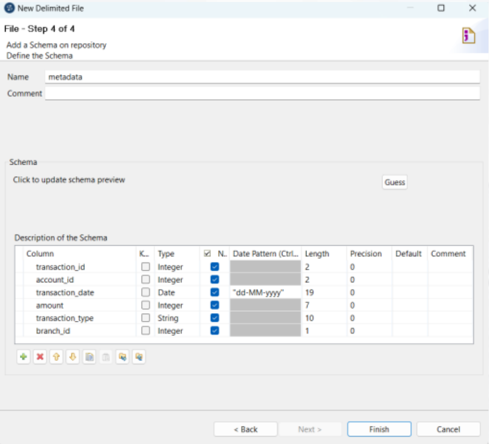
Add a Metadata File on repository
Define the path of the file and the format settings

File Settings

Server: Localhost 127.0.0.1
File: D:/SAYID COURSE/PBI_DataEngineer_Rakamin/transaction_csv.csv
Browse...
Format: WINDOWS

File Viewer

```
transaction_id,account_id,transaction_date,amount,transaction_type,branch_id
14,13,21-01-2024 14:00:00,1500000,Deposit,4
15,14,21-01-2024 08:00:00,500000,Transfer,3
16,15,22-01-2024 09:00:00,100000,Deposit,1
17,16,22-01-2024 13:10:00,100000,Withdrawal,5
18,17,22-01-2024 10:20:00,700000,Deposit,5
19,18,22-01-2024 11:00:00,30000,Payment,2
20,19,22-01-2024 15:00:00,2500000,Deposit,2
21,20,22-01-2024 11:30:00,150000,Payment,4
22,21,22-01-2024 10:45:00,800000,Withdrawal,5
23,22,22-01-2024 10:50:00,100000,Withdrawal,1
24,23,22-01-2024 11:10:00,300000,Payment,1
```



File - Step 4 of 4

Add a Schema on repository
Define the Schema

Name: metadata
Comment:

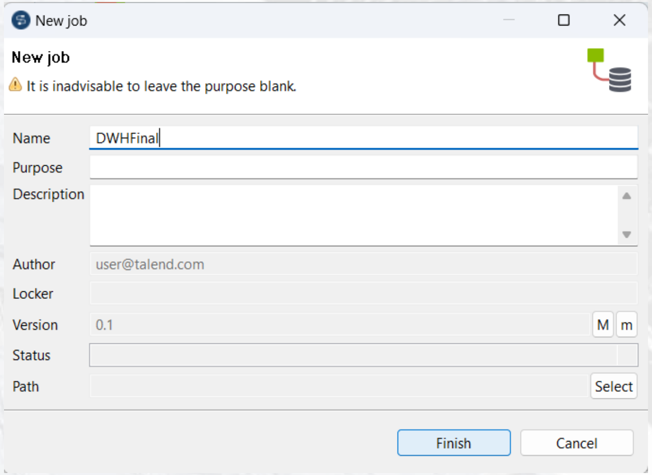
Schema

Click to update schema preview

Description of the Schema

| Column | K.. | Type | ☑ N.. | Date Pattern (Ctrl.. | Length | Precision | Default | Comment |
|---|---|---|---|---|---|---|---|---|
| transaction_id | □ | Integer | ☑ | | 2 | 0 | | |
| account_id | □ | Integer | ☑ | | 2 | 0 | | |
| transaction_date | □ | Date | ☑ | "dd-MM-yyyy" | 19 | 0 | | |
| amount | □ | Integer | ☑ | | 7 | 0 | | |
| transaction_type | □ | String | ☑ | | 10 | 0 | | |
| branch_id | □ | Integer | ☑ | | 1 | 0 | | |

# 5. Make new job in Talend

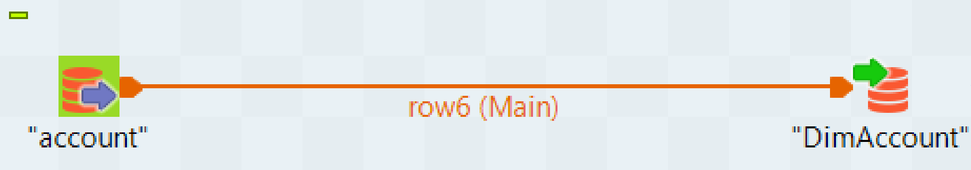


New job

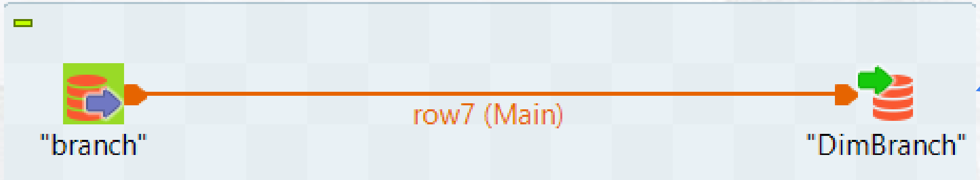
⚠ It is inadvisable to leave the purpose blank.

Name: DWHFinal
Purpose:
Description:
Author: user@talend.com
Locker:
Version: 0.1 M m
Status:
Path:

# 3. Create ETL Job for Dimension Table

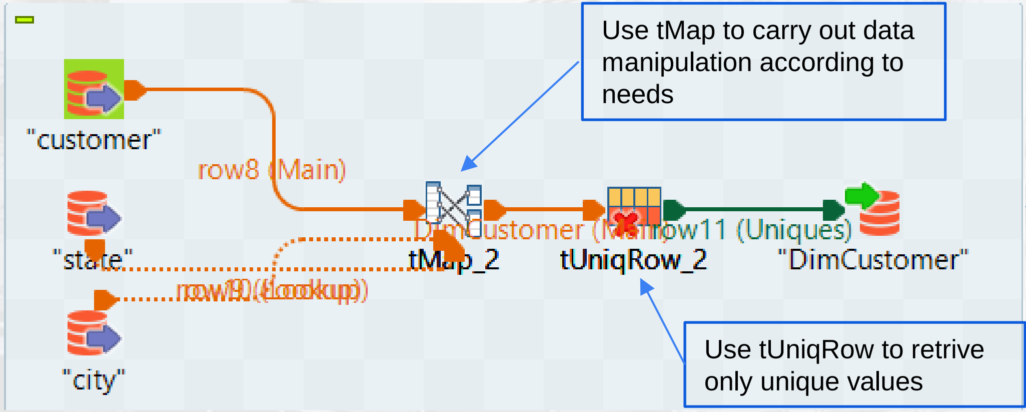


Transfer data from source (account) to dimention table (DimAccount)



Transfer data from source (branch) to dimention table (DimBranch)

DimAccount and DimBranch table columns follow the columns of the existing table from the source.



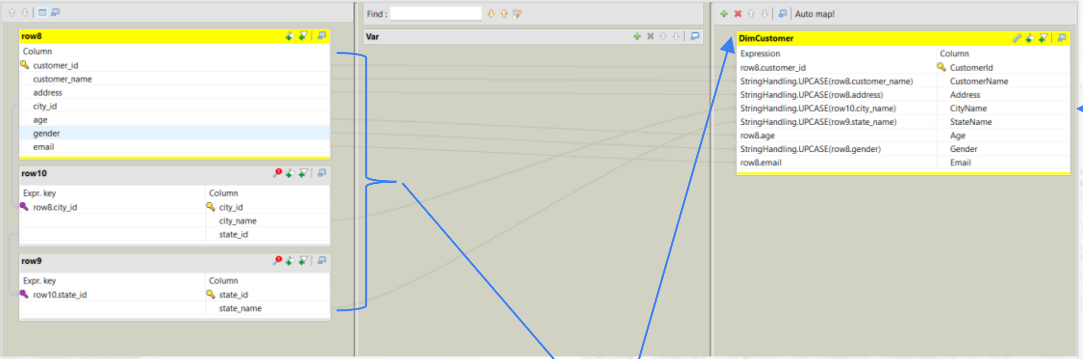
Use tMap to carry out data manipulation according to needs

Use tUniqRow to retrive only unique values

Transfer data from source (customer,state,city) to dimention table (DimAccount)

DimCustomer is formed from the combination of the customer, city and state tables to retrieve the CityName and StateName columns.

# tMap used to transform data in DimCustomer



**row8**

| Column |
|---|
| customer_id |
| customer_name |
| address |
| city_id |
| age |
| gender |
| email |

**row10**

| Expr. key | Column |
|---|---|
| row8.city_id | city_id |
| | city_name |
| | state_id |

**row9**

| Expr. key | Column |
|---|---|
| row10.state_id | state_id |
| | state_name |

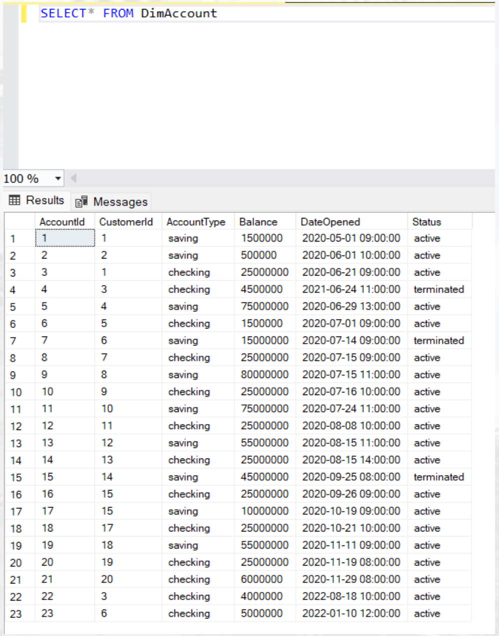
**DimCustomer**

| Expression | Column |
|---|---|
| row8.customer_id | CustomerId |
| StringHandling.UPCASE(row8.customer_name) | CustomerName |
| StringHandling.UPCASE(row8.address) | Address |
| StringHandling.UPCASE(row10.city_name) | CityName |
| StringHandling.UPCASE(row9.state_name) | StateName |
| row8.age | Age |
| StringHandling.UPCASE(row8.gender) | Gender |
| row8.email | Email |

For DimCustomer tables, format the column stored is CustomerID, CustomerName,Address,CityName,StateName,Age,Gender, Email. All data from that column is changed be capitalized except for columns CustomerID, Age and Email

DimCustomer is formed from the combination of the customer, city and state tables to retrieve the CityName and StateName columns.

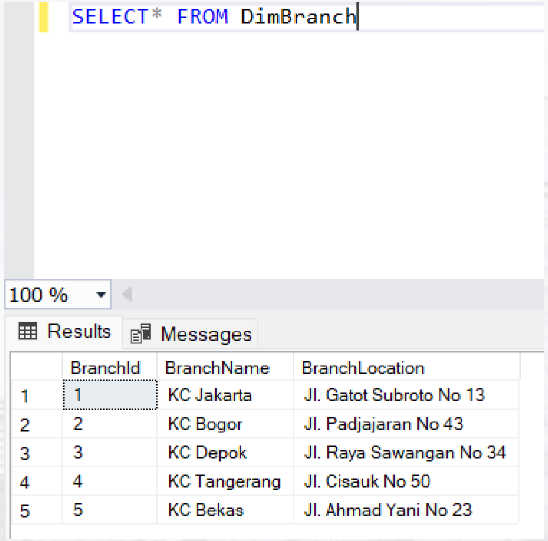
**Result** : DimAccount table in DWH database



```
SELECT* FROM DimAccount
```

| | AccountId | CustomerId | AccountType | Balance | DateOpened | Status |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | saving | 1500000 | 2020-05-01 09:00:00 | active |
| 2 | 2 | 2 | saving | 500000 | 2020-06-01 10:00:00 | active |
| 3 | 3 | 1 | checking | 25000000 | 2020-06-21 09:00:00 | active |
| 4 | 4 | 3 | checking | 4500000 | 2021-06-24 11:00:00 | terminated |
| 5 | 5 | 4 | saving | 75000000 | 2020-06-29 13:00:00 | active |
| 6 | 6 | 5 | checking | 1500000 | 2020-07-01 09:00:00 | active |
| 7 | 7 | 6 | saving | 15000000 | 2020-07-14 09:00:00 | terminated |
| 8 | 8 | 7 | checking | 25000000 | 2020-07-15 09:00:00 | active |
| 9 | 9 | 8 | saving | 80000000 | 2020-07-15 11:00:00 | active |
| 10 | 10 | 9 | checking | 25000000 | 2020-07-16 10:00:00 | active |
| 11 | 11 | 10 | saving | 75000000 | 2020-07-24 11:00:00 | active |
| 12 | 12 | 11 | checking | 25000000 | 2020-08-08 10:00:00 | active |
| 13 | 13 | 12 | saving | 55000000 | 2020-08-15 11:00:00 | active |
| 14 | 14 | 13 | checking | 25000000 | 2020-08-15 14:00:00 | active |
| 15 | 15 | 14 | saving | 45000000 | 2020-09-25 08:00:00 | terminated |
| 16 | 16 | 15 | checking | 25000000 | 2020-09-26 09:00:00 | active |
| 17 | 17 | 15 | saving | 10000000 | 2020-10-19 09:00:00 | active |
| 18 | 18 | 17 | checking | 25000000 | 2020-10-21 10:00:00 | active |
| 19 | 19 | 18 | saving | 55000000 | 2020-11-11 09:00:00 | active |
| 20 | 20 | 19 | checking | 25000000 | 2020-11-19 08:00:00 | active |
| 21 | 21 | 20 | checking | 6000000 | 2020-11-29 08:00:00 | active |
| 22 | 22 | 3 | checking | 4000000 | 2022-08-18 10:00:00 | active |
| 23 | 23 | 6 | checking | 5000000 | 2022-01-10 12:00:00 | active |

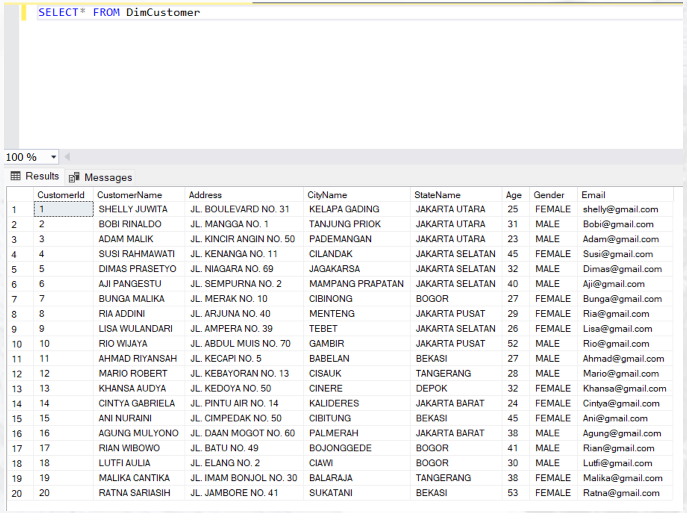
**Result** : DimBranch table in DWH database



```
SELECT* FROM DimBranch
```

| | BranchId | BranchName | BranchLocation |
|---|---|---|---|
| 1 | 1 | KC Jakarta | Jl. Gatot Subroto No 13 |
| 2 | 2 | KC Bogor | Jl. Padjajaran No 43 |
| 3 | 3 | KC Depok | Jl. Raya Sawangan No 34 |
| 4 | 4 | KC Tangerang | Jl. Cisauk No 50 |
| 5 | 5 | KC Bekas | Jl. Ahmad Yani No 23 |

**Result** : DimCustomer table in DWH database

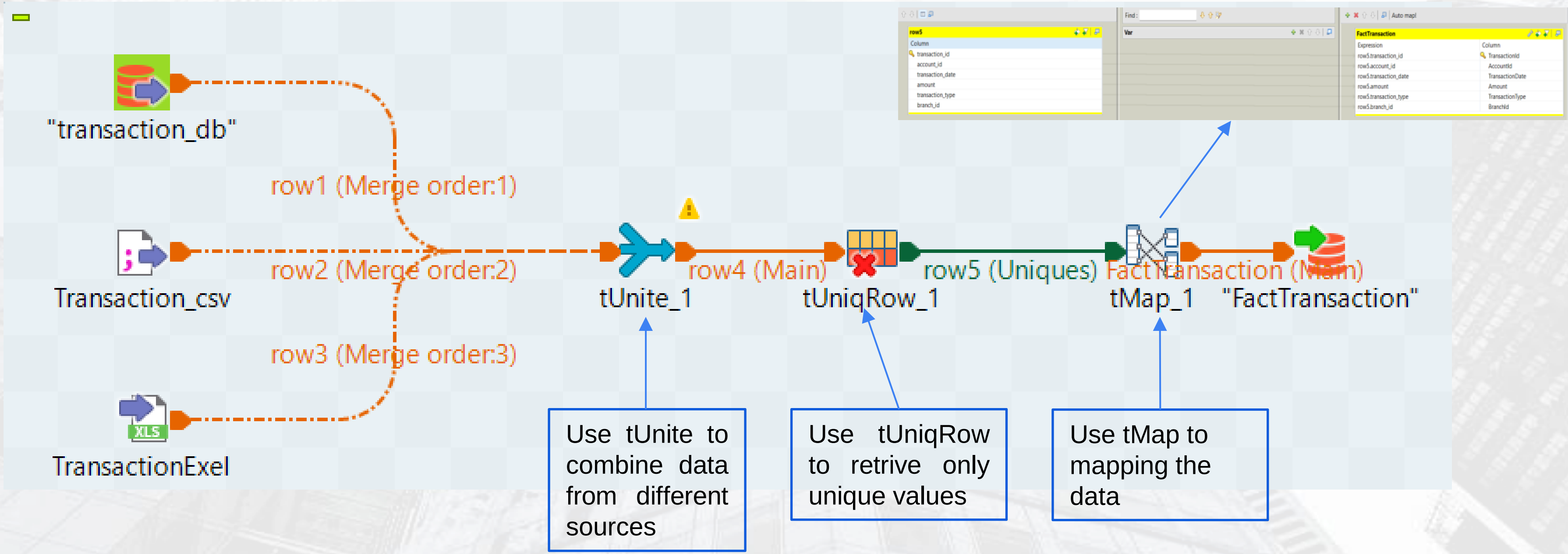


```
SELECT* FROM DimCustomer
```

| | CustomerId | CustomerName | Address | CityName | StateName | Age | Gender | Email |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | SHELLY JUWITA | JL. BOULEVARD NO. 31 | KELAPA GADING | JAKARTA UTARA | 25 | FEMALE | shelly@gmail.com |
| 2 | 2 | BOBI RINALDO | JL. MANGGA NO. 1 | TANJUNG PRIOK | JAKARTA UTARA | 31 | MALE | Bobi@gmail.com |
| 3 | 3 | ADAM MALIK | JL. KINCIR ANGIN NO. 50 | PADEMANGAN | JAKARTA UTARA | 23 | MALE | Adam@gmail.com |
| 4 | 4 | SUSI RAHMAWATI | JL. KENANGA NO. 11 | CILANDAK | JAKARTA SELATAN | 45 | FEMALE | Susi@gmail.com |
| 5 | 5 | DIMAS PRASETYO | JL. NIAGARA NO. 69 | JAGAKARSA | JAKARTA SELATAN | 32 | MALE | Dimas@gmail.com |
| 6 | 6 | AJI PANGESTU | JL. SEMPURNA NO. 2 | MAMPANG PRAPATAN | JAKARTA SELATAN | 40 | MALE | Aji@gmail.com |
| 7 | 7 | BUNGA MALIKA | JL. MERAK NO. 10 | CIBINONG | BOGOR | 27 | FEMALE | Bunga@gmail.com |
| 8 | 8 | RIA ADDINI | JL. ARJUNA NO. 40 | MENTENG | JAKARTA PUSAT | 29 | FEMALE | Ria@gmail.com |
| 9 | 9 | LISA WULANDARI | JL. AMPERA NO. 39 | TEBET | JAKARTA SELATAN | 26 | FEMALE | Lisa@gmail.com |
| 10 | 10 | RIO WIJAYA | JL. ABDUL MUIS NO. 70 | GAMBIR | JAKARTA PUSAT | 52 | MALE | Rio@gmail.com |
| 11 | 11 | AHMAD RIYANSAH | JL. KECAPI NO. 5 | BABELAN | BEKASI | 27 | MALE | Ahmad@gmail.com |
| 12 | 12 | MARIO ROBERT | JL. KEBAYORAN NO. 13 | CISAUK | TANGERANG | 28 | MALE | Mario@gmail.com |
| 13 | 13 | KHANSA AUDYA | JL. KEDOYA NO. 50 | CINERE | DEPOK | 32 | FEMALE | Khansa@gmail.com |
| 14 | 14 | CINTYA GABRIELA | JL. PINTU AIR NO. 14 | KALIDERES | JAKARTA BARAT | 24 | FEMALE | Cintya@gmail.com |
| 15 | 15 | ANI NURAINI | JL. CIMPEDAK NO. 50 | CIBITUNG | BEKASI | 45 | FEMALE | Ani@gmail.com |
| 16 | 16 | AGUNG MULYONO | JL. DAAN MOGOT NO. 60 | PALMERAH | JAKARTA BARAT | 38 | MALE | Agung@gmail.com |
| 17 | 17 | RIAN WIBOWO | JL. BATU NO. 49 | BOJONGGEDE | BOGOR | 41 | MALE | Rian@gmail.com |
| 18 | 18 | LUTFI AULIA | JL. ELANG NO. 2 | CIAWI | BOGOR | 30 | MALE | Lutfi@gmail.com |
| 19 | 19 | MALIKA CANTIKA | JL. IMAM BONJOL NO. 30 | BALARAJA | TANGERANG | 38 | FEMALE | Malika@gmail.com |
| 20 | 20 | RATNA SARIASIH | JL. JAMBORE NO. 41 | SUKATANI | BEKASI | 53 | FEMALE | Ratna@gmail.com |

# 4. Create ETL Job for Fact Table

All Fact table columns follow the columns of the existing table from the source.



| row5 |
|---|
| Column |
| transaction_id |
| account_id |
| transaction_date |
| amount |
| transaction_type |
| branch_id |

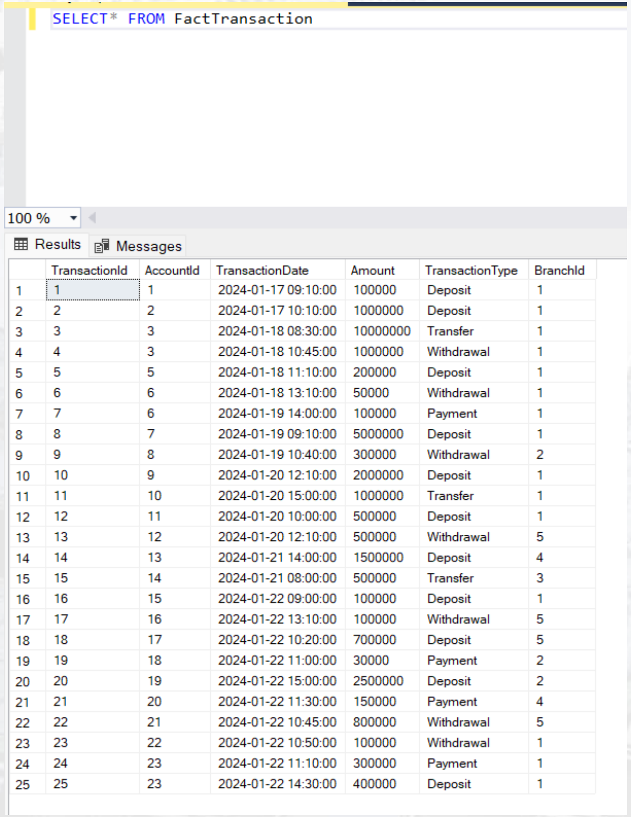
| FactTransaction | |
|---|---|
| Expression | Column |
| row5.transaction_id | TransactionId |
| row5.account_id | AccountId |
| row5.transaction_date | TransactionDate |
| row5.amount | Amount |
| row5.transaction_type | TransactionType |
| row5.branch_id | BranchId |

Use tUnite to combine data from different sources

Use tUniqRow to retrive only unique values

Use tMap to mapping the data

**Result** : FactTransaction table in DWH database



```sql
SELECT* FROM FactTransaction
```

|  | TransactionId | AccountId | TransactionDate | Amount | TransactionType | BranchId |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2024-01-17 09:10:00 | 100000 | Deposit | 1 |
| 2 | 2 | 2 | 2024-01-17 10:10:00 | 1000000 | Deposit | 1 |
| 3 | 3 | 3 | 2024-01-18 08:30:00 | 10000000 | Transfer | 1 |
| 4 | 4 | 3 | 2024-01-18 10:45:00 | 1000000 | Withdrawal | 1 |
| 5 | 5 | 5 | 2024-01-18 11:10:00 | 200000 | Deposit | 1 |
| 6 | 6 | 6 | 2024-01-18 13:10:00 | 50000 | Withdrawal | 1 |
| 7 | 7 | 6 | 2024-01-19 14:00:00 | 100000 | Payment | 1 |
| 8 | 8 | 7 | 2024-01-19 09:10:00 | 5000000 | Deposit | 1 |
| 9 | 9 | 8 | 2024-01-19 10:40:00 | 300000 | Withdrawal | 2 |
| 10 | 10 | 9 | 2024-01-20 12:10:00 | 2000000 | Deposit | 1 |
| 11 | 11 | 10 | 2024-01-20 15:00:00 | 1000000 | Transfer | 1 |
| 12 | 12 | 11 | 2024-01-20 10:00:00 | 500000 | Deposit | 1 |
| 13 | 13 | 12 | 2024-01-20 12:10:00 | 500000 | Withdrawal | 5 |
| 14 | 14 | 13 | 2024-01-21 14:00:00 | 1500000 | Deposit | 4 |
| 15 | 15 | 14 | 2024-01-21 08:00:00 | 500000 | Transfer | 3 |
| 16 | 16 | 15 | 2024-01-22 09:00:00 | 100000 | Deposit | 1 |
| 17 | 17 | 16 | 2024-01-22 13:10:00 | 100000 | Withdrawal | 5 |
| 18 | 18 | 17 | 2024-01-22 10:20:00 | 700000 | Deposit | 5 |
| 19 | 19 | 18 | 2024-01-22 11:00:00 | 30000 | Payment | 2 |
| 20 | 20 | 19 | 2024-01-22 15:00:00 | 2500000 | Deposit | 2 |
| 21 | 21 | 20 | 2024-01-22 11:30:00 | 150000 | Payment | 4 |
| 22 | 22 | 21 | 2024-01-22 10:45:00 | 800000 | Withdrawal | 5 |
| 23 | 23 | 22 | 2024-01-22 10:50:00 | 100000 | Withdrawal | 1 |
| 24 | 24 | 23 | 2024-01-22 11:10:00 | 300000 | Payment | 1 |
| 25 | 25 | 23 | 2024-01-22 14:30:00 | 400000 | Deposit | 1 |

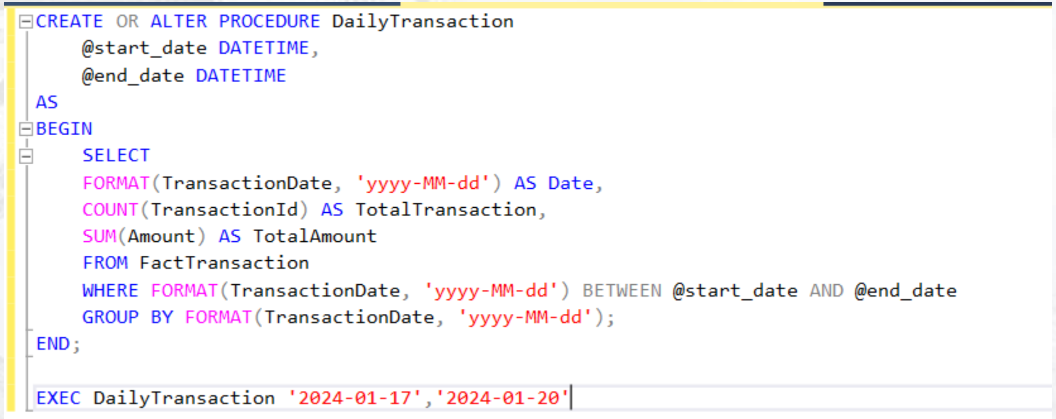
# 5. Create Stored Procedure

**Create two Stored Procedures (SP) with parameters, to help them get a summary of the data quickly. The requested Stored Procedure is:**

1. **DailyTransaction** (to calculate the number transactions along with the total nominal value every day)

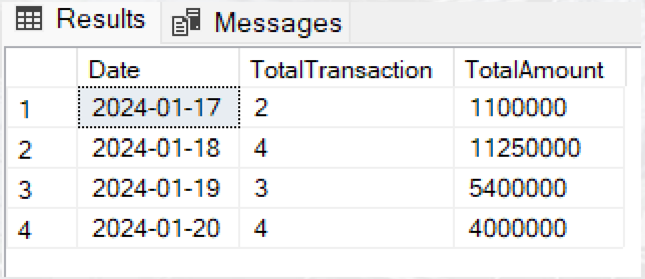
2. **BalancePerCustomer** (to find out remaining balance per customer)

# Stored Procedure : DailyTransaction



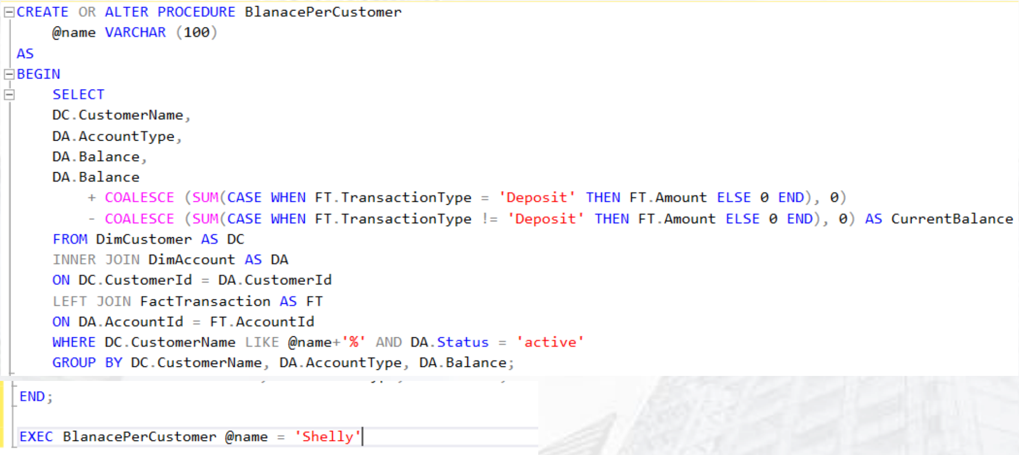
```sql
CREATE OR ALTER PROCEDURE DailyTransaction
    @start_date DATETIME,
    @end_date DATETIME
AS
BEGIN
    SELECT
    FORMAT(TransactionDate, 'yyyy-MM-dd') AS Date,
    COUNT(TransactionId) AS TotalTransaction,
    SUM(Amount) AS TotalAmount
    FROM FactTransaction
    WHERE FORMAT(TransactionDate, 'yyyy-MM-dd') BETWEEN @start_date AND @end_date
    GROUP BY FORMAT(TransactionDate, 'yyyy-MM-dd');
END;

EXEC DailyTransaction '2024-01-17','2024-01-20'
```

# Result : DailyTransaction



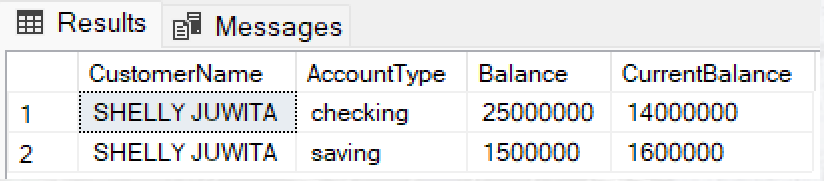
|   | Date | TotalTransaction | TotalAmount |
|---|---|---|---|
| 1 | 2024-01-17 | 2 | 1100000 |
| 2 | 2024-01-18 | 4 | 11250000 |
| 3 | 2024-01-19 | 3 | 5400000 |
| 4 | 2024-01-20 | 4 | 4000000 |

# Stored Procedure : BalancePerCustomer



```
CREATE OR ALTER PROCEDURE BlanacePerCustomer
    @name VARCHAR (100)
AS
BEGIN
    SELECT
    DC.CustomerName,
    DA.AccountType,
    DA.Balance,
    DA.Balance
        + COALESCE (SUM(CASE WHEN FT.TransactionType = 'Deposit' THEN FT.Amount ELSE 0 END), 0)
        - COALESCE (SUM(CASE WHEN FT.TransactionType != 'Deposit' THEN FT.Amount ELSE 0 END), 0) AS CurrentBalance
    FROM DimCustomer AS DC
    INNER JOIN DimAccount AS DA
    ON DC.CustomerId = DA.CustomerId
    LEFT JOIN FactTransaction AS FT
    ON DA.AccountId = FT.AccountId
    WHERE DC.CustomerName LIKE @name+'%' AND DA.Status = 'active'
    GROUP BY DC.CustomerName, DA.AccountType, DA.Balance;
END;

EXEC BlanacePerCustomer @name = 'Shelly'
```

# Result : BalancePerCustomer



| | CustomerName | AccountType | Balance | CurrentBalance |
|---|---|---|---|---|
| 1 | SHELLY JUWITA | checking | 25000000 | 14000000 |
| 2 | SHELLY JUWITA | saving | 1500000 | 1600000 |

## Link Github : Hasil pengerjaan SQL

1. Dimention Table dan FactTransaction
2. Stored procedure DailyTransaction
3. Stored procedure BalancePerCustomer

# Thank You

Rakamin Academy X id/x partners