Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Report
on
TransactionHistory_db

[Code No: COMP 232]
(For partial fulfillment of 2nd year/2nd Semester in Computer Science)

Submitted by

Gaurab Khanal(19)

Submitted to:

Mr. Sanjog Sigdel
Department of Computer Science and Engineering

Submission Date: 2024-12-31

Contents

Introduction 4

ER Diagram 4

SQL Queries Executed 5

1. Creation of the Database 5
2. To use the database 5
3. Creation of the Table 5
   - UsersTable 5
   - TransactionTable 5
   - ProductTable 6
   - TransactionHistory Table 6
4. DESC TABLE 6
   - desc UserTable 6
   - Desc TransactionTable 7
   - desc productTable 7
   - DESC transactionhistory 8
5. Insert the values in the table 8
   - Insert into UserTable 8
   - Insert into TransactionHistory 9
   - Insert into ProductTable 9
   - Insert into TransactionTable 9
6. Selection 10
   - select * from UserTable 10
   - select * from TransactionTable 10
   - select * from ProductTable 11
   - select * from TransactionHistory 11
7. Projection 12
   - Projection on TransactionTable 12
   - Projection on TransactionHistory 12
8. Cartesian Product 13
9. Inner Join 14
10. Left Join 15

11. Right Join 15
12. Normalization 16
- Tables in 1st Normalization Form 16
- Tables in 2nd Normalization Form 16
- Tables in 3rd Normalization Form 17
13. Transaction management 18
- Transaction with error scenario 19
- Sucessful Transaction without Rollback scenario 19
- Successful Transaction with commit and Rollback 19

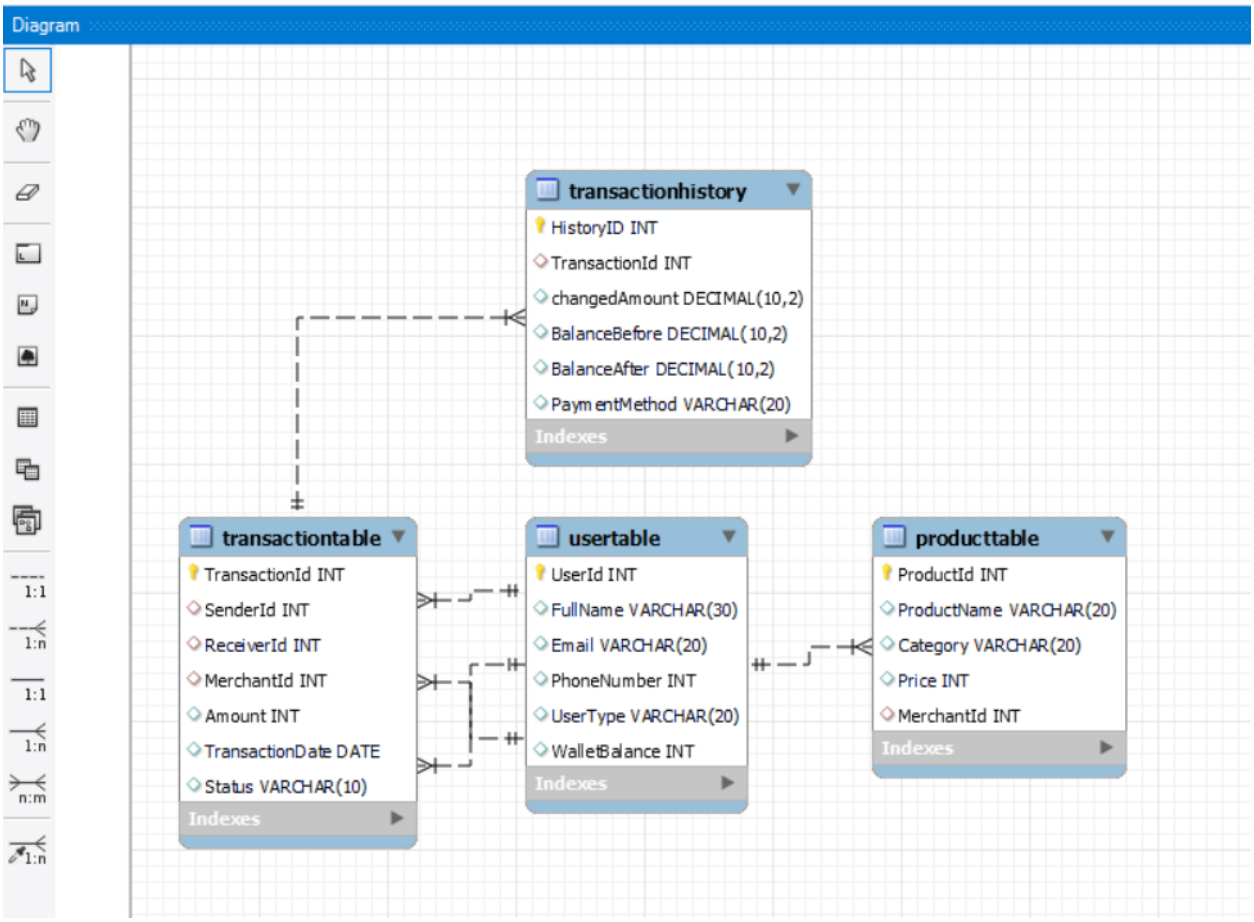
Conclusion 20

Introduction

This project demonstrates the implementation of database normalization, transaction management, and various SQL operations using MySQL Workbench. The database is modeled for a wallet application, incorporating tables for users, transactions, products, and transaction history. The objective is to showcase database operations such as normalization, transaction management, and SQL joins to ensure data consistency and integrity.

ER Diagram

The following Entity-Relationship (ER) diagram represents the database schema and relationships among tables:



SQL Queries Executed

1. Creation of the Database:

   -create database TransactionHistory_db;

2. To use the database:

   -use database TransactionHistory_db;

3. Creation of the Table:

   There are 4 types of tables in the TransactionHistory_db. Each table is interrelated to each other's. The creation of the tables are shown below:

   - **UsersTable**:

     ```
     create table UserTable(
     UserId int auto_increment primary key,
     FullName varchar(30),
     Email Varchar(20),
     PhoneNumber int,
     UserType varchar(20),
     WalletBalance int);
     ```

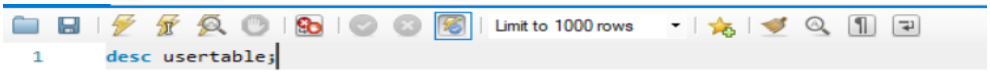
   - **TransactionTable**:

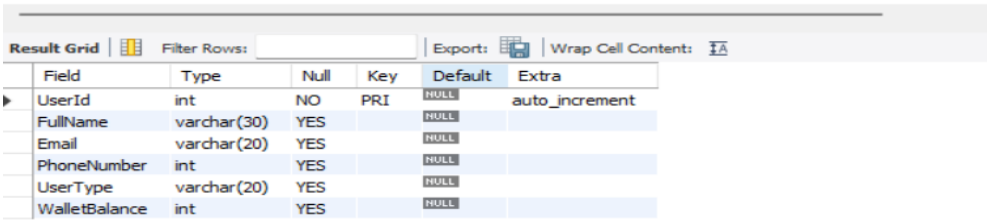
     ```
     create table TransactionTable(
     TransactionId int auto_increment Primary key,
     SenderId int,
     ReceiverId int,
     MerchantId int,
     Amount int,
     TransactionDate Date,
     Status varchar(10),
     foreign key(SenderId) references UserTable(UserId),
     foreign key(ReceiverId) references UserTable(UserId),
     foreign key(MerchantId) references UserTable(UserId));
     ```

- **ProductTable**:
  create table ProductTable(
  ProductId int auto_increment primary key,
  ProductName varchar(20),
  Category varchar(20),
  Price int,
  MerchantId int,
  foreign key(MerchantId) references UserTable(UserId));

- **TransactionHistory Table**:
  create table TransactionHistory(
  HistoryID int auto_increment primary Key,
  TransactionID int,
  changedAmount decimal(10,2),
  BalanceBefore decimal(10,2),
  BalanceAfter decimal(10,2),
  PaymentMethod varchar(20));

4. DESC TABLE;
   - **desc UserTable**;





Field	Type	Null	Key	Default	Extra
UserId	int	NO	PRI	NULL	auto_increment
FullName	varchar(30)	YES		NULL	
Email	varchar(20)	YES		NULL	
PhoneNumber	int	YES		NULL	
UserType	varchar(20)	YES		NULL	
WalletBalance	int	YES		NULL	

- **Desc TransactionTable**;

```
desc TransactionTable;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
TransactionId	int	NO	PRI	NULL	auto_increment
SenderId	int	YES	MUL	NULL	
ReceiverId	int	YES	MUL	NULL	
MerchantId	int	YES	MUL	NULL	
Amount	int	YES		NULL	
TransactionDate	date	YES		NULL	
Status	varchar(10)	YES		NULL	

- **desc productTable**;

```
desc productTable;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
ProductId	int	NO	PRI	NULL	auto_increment
ProductName	varchar(20)	YES		NULL	
Category	varchar(20)	YES		NULL	
Price	int	YES		NULL	
MerchantId	int	YES	MUL	NULL	

- **DESC transactionhistory**;

```
7 •  DESC transactionhistory;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
HistoryID	int	NO	PRI	NULL	auto_increment
TransactionID	int	YES		NULL	
changedAmount	decimal(10,2)	YES		NULL	
BalanceBefore	decimal(10,2)	YES		NULL	
BalanceAfter	decimal(10,2)	YES		NULL	
PaymentMethod	varchar(20)	YES		NULL	

5. Insert the values in the table.

- **Insert into UserTable**:

```
INSERT INTO UserTable
(`UserId`, `FullName`, `Email`, `PhoneNumber`, `UserType`,
`WalletBalance`)
VALUES
('1', 'Gaurab Khanal', 'gaurabkhanal0555@gmail.com', '9800000000', 'Sender',
'1000'),
('2', 'Neo Shakya', 'neoshakya@gmail.com', '9800000001', 'Receiver', '2000'),
('3', 'Pragyan Shrestha', 'pragyanshrestha@gmail.com', '9800000002', 'Sender',
'3000'),
('4', 'Kathmandu University', 'ku.edu.np', '071577259', 'Merchant', '20000'),
('5', 'ITTI', 'itti.com.np', '071577260', 'Merchant', '30000'),
('6', 'Mudita', 'mudita.com.np', '071577261', 'Merchant', '40000');
```

- **Insert into TransactionHistory**:
  INSERT INTO TransactionHistory
  (HistoryID, TransactionID, ChangeAmount, BalanceBefore, BalanceAfter, PaymentMethod)
  VALUES
  ('301', '101', '-500.00', '1500.50', '1000.50', 'Credit Card'),
  ('302', '102', '-1500.00', '2000.00', '500.00', 'esewa'),
  ('303', '103', '-5000.00', '20000.00', '15000.00', 'Debit Card'),
  ('304', '104', '-1000.00', '1500.50', '500.50', 'Credit Card'),
  ('305', '105', '-200.00', '2000.00', '1800.00', 'khalti');

- **Insert into ProductTable**:
  INSERT INTO ProductTable
  (ProductID, ProductName, Category, Price, MerchantID)
  VALUES
  ('1001', 'Smartphone', 'Electronics', '799.99', '4'),
  ('1002', 'Laptop', 'Electronics', '1200.00', '4'),
  ('1003', 'Headphones', 'Electronics', '199.99', '5'),
  ('1004', 'Washing Machine', 'Home Appliances', '450.00', '5'),
  ('1005', 'Refrigerator', 'Home Appliances', '850.00', '6');

- **Insert into TransactionTable**:
  INSERT INTO TransactionTable
  (TransactionID, SenderID, ReceiverID, MerchantID, Amount, transactionDate, Status)
  VALUES
  ('101', '1', '2', NULL, '500', '2024-12-25', 'Completed'),
  ('102', '3', '2', NULL, '1500', '2024-12-26', 'Completed'),
  ('103', '4', NULL, '5', '5000', '2024-12-27', 'Completed'),
  ('104', '1', '3', NULL, '1000', '2024-12-28', 'Pending'),
  ('105', '3', NULL, '5','200', '2024-12-29', 'Completed');

6. Selection:

- **select * from UserTable;**

```
select * from usertable;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

UserId	FullName	Email	PhoneNumber	UserType	WalletBalance
1	Gaurab Khanal	gaurab@gmail.com	9800000	Sender	1000
2	Neo Shakya	neoshakya@gmail.com	9800000	Receiver	2000
3	Pragyan Shrestha	pragyan@gmail.com	980000	Sender	3000
4	Kathmandu University	ku.edu.np	71577	Merchant	20000
5	ITTI	itti.com.np	715772	Merchant	30000
6	Mudita	mudita.com.np	71577	Merchant	40000
NULL	NULL	NULL	NULL	NULL	NULL

- **select * from TransactionTable;**

```
select * from transactiontable;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

TransactionId	SenderId	ReceiverId	MerchantId	Amount	TransactionDate	Status
101	1	2	NULL	500	2024-12-25	Completed
102	3	2	NULL	1500	2024-12-26	Completed
103	4	NULL	5	5000	2024-12-27	Completed
104	1	3	NULL	1000	2024-12-28	Pending
105	3	NULL	5	2000	2024-12-29	Completed
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **select * from ProductTable**;

```
1 •    select * from ProductTable;
```

esult Grid | Filter Rows: | Edit: | Export/Import:

ProductId	ProductName	Category	Price	MerchantId
1001	Smartphone	Electronics	800	4
1002	Laptop	Electronics	1200	4
1003	Headphones	Electronics	200	5
1004	Washing Machine	Home Appliances	450	5
1005	Refrigerator	Home Appliances	850	6
NULL	NULL	NULL	NULL	NULL

- **select * from TransactionHistory**;

```
1    select * from TransactionHistory;
```

sult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

HistoryID	TransactionID	changedAmount	BalanceBefore	BalanceAfter	PaymentMethod
301	101	-500.00	1500.50	1000.50	Credit Card
302	102	-1500.00	2000.00	500.00	esewa
303	103	-5000.00	20000.00	15000.00	Debit Card
304	104	-1000.00	1500.50	500.50	Credit Card
305	105	-200.00	2000.00	1800.00	khalti
NULL	NULL	NULL	NULL	NULL	NULL

7. Projection:

- **Projection on TransactionTable**:

```
SELECT TransactionID, SenderID, Amount
FROM TransactionTable;
```

sult Grid | Filter Rows: | Export:

TransactionID	SenderID	Amount
101	1	500
102	3	1500
103	4	5000
104	1	1000
105	3	2000

- **Projection on TransactionHistory**:

```
SELECT TransactionID, PaymentMethod
FROM TransactionHistory;
```

esult Grid | Filter Rows: | Export: | Wra

TransactionID	PaymentMethod
101	Credit Card
102	esewa
103	Debit Card
104	Credit Card
105	khalti

8. Cartesian Product:

- Using the union operation between usertable and transactiontable:

```sql
SELECT TransactionID AS ID
FROM TransactionTable
UNION
SELECT UserId AS ID
FROM UserTable;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID
101
104
102
105
103
1
2
3
4
5
6

- using the exception operation:

```sql
SELECT UserId
FROM UserTable
EXCEPT
SELECT SenderID
FROM TransactionTable;
```

Result Grid | Filter Rows:

UserId
2
5
6

9. Inner Join:

- Joining TransactionTable and TransactionHistory

```
SELECT t.TransactionID, t.Amount, th.PaymentMethod
FROM TransactionTable t
INNER JOIN TransactionHistory th ON t.TransactionID = th.TransactionID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	Amount	PaymentMethod
101	500	Credit Card
102	1500	esewa
103	5000	Debit Card
104	1000	Credit Card
105	2000	khalti

- Joining TransactionTable , UserTable and TransactionHistory

```
SELECT t.TransactionID, u.FullName AS Sender, t.Amount, th.PaymentMethod
FROM TransactionTable t
INNER JOIN UserTable u ON t.SenderID = u.UserId
INNER JOIN TransactionHistory th ON t.TransactionID = th.TransactionID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	Sender	Amount	PaymentMethod
101	Gaurab Khanal	500	Credit Card
102	Pragyan Shrestha	1500	esewa
103	Kathmandu University	5000	Debit Card
104	Gaurab Khanal	1000	Credit Card
105	Pragyan Shrestha	2000	khalti

10. Left Join:

- left Joining TransactionTable , UserTable and TransactionHistory

```sql
SELECT t.TransactionID, u.FullName AS Sender, t.Amount, th.PaymentMethod
FROM TransactionTable t
LEFT JOIN UserTable u ON t.SenderID = u.UserId
LEFT JOIN TransactionHistory th ON t.TransactionID = th.TransactionID;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	Sender	Amount	PaymentMethod
101	Gaurab Khanal	500	Credit Card
102	Pragyan Shrestha	1500	esewa
103	Kathmandu University	5000	Debit Card
104	Gaurab Khanal	1000	Credit Card
105	Pragyan Shrestha	2000	khalti

11. Right Join:

- Right Join between TransactionTable and TransactionHistory:

Limit to 1000 rows

```sql
SELECT
    t.TransactionID,
    u.FullName AS Sender,
    t.Amount,
    th.PaymentMethod
FROM
    TransactionTable t
RIGHT JOIN
    UserTable u ON t.SenderID = u.UserId
RIGHT JOIN
    TransactionHistory th ON t.TransactionID = th.TransactionID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TransactionID	Sender	Amount	PaymentMethod
101	Gaurab Khanal	500	Credit Card
102	Pragyan Shrestha	1500	esewa
103	Kathmandu University	5000	Debit Card
104	Gaurab Khanal	1000	Credit Card
105	Pragyan Shrestha	2000	khalti

12. Normalization:

- **Tables in 1st Normalization Form**: To be in 1NF all the columns contain atomic values (no repeating groups, and each column contains a single value).

UserId	FullName	Email	PhoneNumber	UserType	WalletBalance	TransactionID	SenderID	ReceiverID	MerchantID	Amount	TransactionDate	Status	HistoryID	ChangeAmount	BalanceBefore	BalanceAfter	PaymentMethod	ProductID	ProductName	Category	Price	ProductMerchantI
1	Gaurab Khanal	gaurabkhanal0555@gmail.com	9800000000	Sender	1000	101	1	2	NULL	500	2024-12-25	Completed	301	500.00	1500.50	1000.50	Credit Card	NULL	NULL	NULL	NULL	NULL
2	Neo Shakya	neoshakya@gmail.com	9800000001	Receiver	2000	102	3	2	NULL	1500	2024-12-26	Completed	302	-1500.00	2000.00	500.00	esewa	NULL	NULL	NULL	NULL	NULL
3	Pragyan Shrestha	pragyanshrestha@gmail.com	9800000002	Sender	3000	104	1	3	NULL	1000	2024-12-28	Pending	304	-1000.00	1500.50	500.50	Credit Card	NULL	NULL	NULL	NULL	NULL
4	Kathmandu Univ.	ku.edu.np	071577259	Merchant	20000	103	4	NULL	5	5000	2024-12-27	Completed	303	-5000.00	20000.00	15000.00	Debit Card	1001	Smartphone	Electronics	799.99	4
5	ITTI	itti.com.np	071577260	Merchant	30000	105	3	NULL	5	200	2024-12-29	Completed	305	200.00	2000.00	1800.00	khalti	1002	Laptop	Electronics	1200	4
6	Mudita	mudita.com.np	071577261	Merchant	40000	NULL	NULL	NULL	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1003	Headphones	Electronics	199.99	5

- **Tables in 2nd Normalization Form**: **2NF** requires that the table is in **1NF** and that all non-key attributes are fully dependent on the primary key (i.e., there should be no partial dependencies). The tables in 2nd normal from are:

  - User Table:

UserId	FullName	Email	PhoneNumber	UserType	WalletBalance
1	Gaurab Khanal	gaurab@gmail.com	9800000	Sender	1000
2	Neo Shakya	neoshakya@gmail.com	9800000	Receiver	2000
3	Pragyan Shrestha	pragyan@gmail.com	980000	Sender	3000
4	Kathmandu University	ku.edu.np	71577	Merchant	20000
5	ITTI	itti.com.np	715772	Merchant	30000
6	Mudita	mudita.com.np	71577	Merchant	40000
NULL	NULL	NULL	NULL	NULL	NULL

  - Combined TransactionTable and TransactionHistory Table:Here ChangeAmount only depends upon HistoryId but not on both TransactionId and HistoryID.

TransactionID	SenderID	ReceiverID	MerchantID	Amount	transactionDate	Status	HistoryID	ChangedAmount	BalanceBefore	BalanceAfter
101	1	2	NULL	500	2024-12-25	Completed	301	-500.00	1500.50	1000.50
102	3	2	NULL	1500	2024-12-26	Completed	302	-1500.00	2000.00	500.00
103	4	NULL	5	5000	2024-12-27	Completed	303	-5000.00	20000.00	15000.00
104	1	3	NULL	1000	2024-12-28	Pending	304	-1000.00	1500.50	500.50
105	3	NULL	5	2000	2024-12-29	Completed	305	-200.00	2000.00	1800.00

- Product Table:

ProductId	ProductName	Category	Price	MerchantId
1001	Smartphone	Electronics	800	4
1002	Laptop	Electronics	1200	4
1003	Headphones	Electronics	200	5
1004	Washing Machine	Home Appliances	450	5
1005	Refrigerator	Home Appliances	850	6
NULL	NULL	NULL	NULL	NULL

- **Tables in 3rd Normalization Form**: **3NF** requires that the table is in **2NF** and that there are no transitive dependencies (i.e., non-key attributes should not depend on other non-key attributes).If we divided the combined tables into two tables then all the tables are in 3NF.The tables in 3rd normal from are:
  - User Table:

UserId	FullName	Email	PhoneNumber	UserType	WalletBalance
1	Gaurab Khanal	gaurab@gmail.com	9800000	Sender	1000
2	Neo Shakya	neoshakya@gmail.com	9800000	Receiver	2000
3	Pragyan Shrestha	pragyan@gmail.com	980000	Sender	3000
4	Kathmandu University	ku.edu.np	71577	Merchant	20000
5	ITTI	itti.com.np	715772	Merchant	30000
6	Mudita	mudita.com.np	71577	Merchant	40000
NULL	NULL	NULL	NULL	NULL	NULL

  - TransactionTable:

TransactionId	SenderId	ReceiverId	MerchantId	Amount	TransactionDate	Status
101	1	2	NULL	500	2024-12-25	Completed
102	3	2	NULL	1500	2024-12-26	Completed
103	4	NULL	5	5000	2024-12-27	Completed
104	1	3	NULL	1000	2024-12-28	Pending
105	3	NULL	5	2000	2024-12-29	Completed
NULL	NULL	NULL	NULL	NULL	NULL	NULL

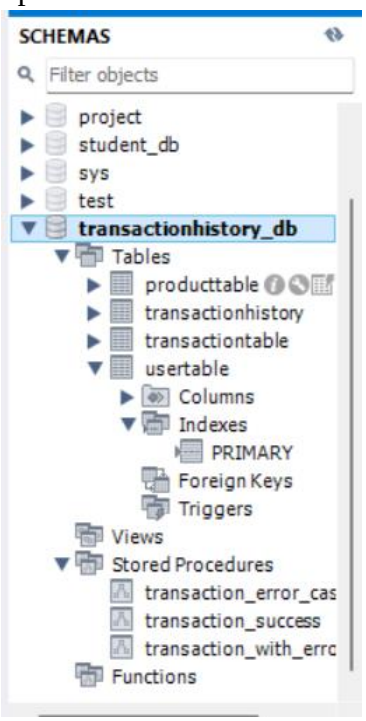
  - Product Table:

ProductId	ProductName	Category	Price	MerchantId
1001	Smartphone	Electronics	800	4
1002	Laptop	Electronics	1200	4
1003	Headphones	Electronics	200	5
1004	Washing Machine	Home Appliances	450	5
1005	Refrigerator	Home Appliances	850	6
NULL	NULL	NULL	NULL	NULL

- TransactionHistory Table:

HistoryID	TransactionId	changedAmount	BalanceBefore	BalanceAfter	PaymentMethod
301	101	-500.00	1500.50	1000.50	Credit Card
302	102	-1500.00	2000.00	500.00	esewa
303	103	-5000.00	20000.00	15000.00	Debit Card
304	104	-1000.00	1500.50	500.50	Credit Card
305	105	-200.00	2000.00	1800.00	khalti
NULL	NULL	NULL	NULL	NULL	NULL

13. Transaction management: Transaction management in MySQL is crucial for ensuring the **integrity, consistency, and reliability** of data when performing multiple related operations.It is stored in stored procedures.



Here are some transaction management operations:

- **Transaction with error scenario**:

Name: transaction_error_case

The name of the routine is parsed automatically from the DD statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `transaction_error_case`()
BEGIN


    START TRANSACTION;

    -- Deduct balance from Gaurab Khanal
    UPDATE UserTable
    SET WalletBalance = WalletBalance - 100
    WHERE FullName = 'Gaurab Khanal';

    -- Attempt to add balance to a non-existent user
    UPDATE UserTable
    SET WalletBalance = WalletBalance + 100
    WHERE FullName = 'NonExistent';

    COMMIT;
END
```

- **Sucessful Transaction without Rollback scenario**:

Name: transaction_success

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `transaction_success`()
begin
start transaction;
update UserTable set WalletBalance=WalletBalance-100 where FullName='Gaurab Khanal';

update UserTable set WalletBalance=WalletBalance+100 where FullName='Neo Shakya';

commit;
end
```

- **Successful Transaction with commit and Rollback**:

Name: transaction_with_error_handling

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `transaction_with_error_handling`()
begin

declare exit handler for sqlexception
begin
rollback;
select 'transaction failed,rollback executed' as message;
end;
start transaction;
update UserTable set WalletBalance=WalletBalance-100 where FullName='Gaurab Khanal';

update UserTable set WalletBalance=WalletBalance+100 where FullName='Neo Shakya';

commit;
end
```

Conclusion

This project successfully demonstrates database normalization, transaction management, and advanced SQL operations in MySQL. The implementation ensures data integrity, efficient querying, and error handling, making it robust for applications like digital wallets like Esewa Khalti which performs operations like sending receiving or buying anything.