DATABASE DOCUMENTATION

1. Introduction

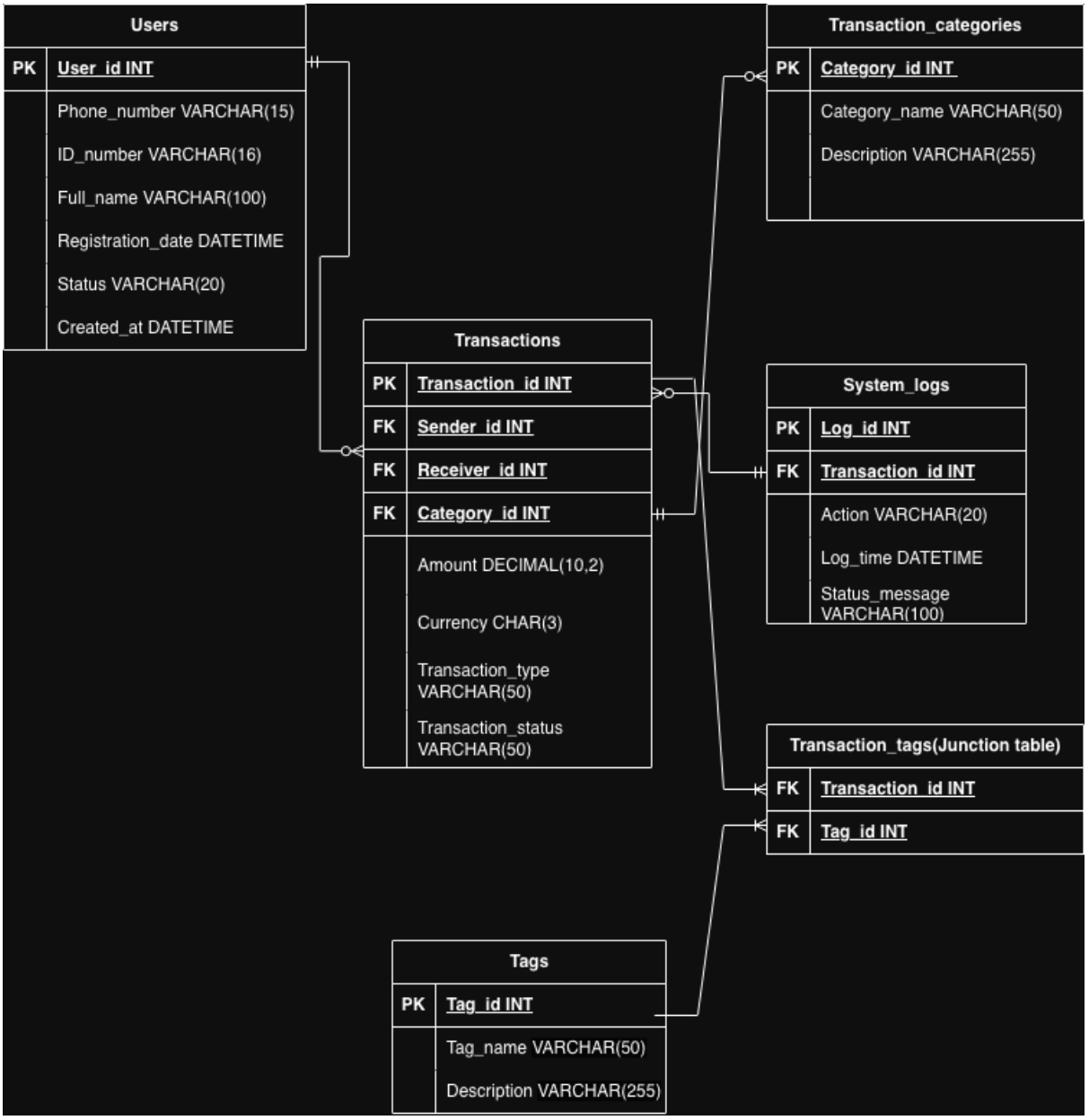
This document describes the design and implementation of a relational database for a Mobile Money (MoMo) SMS transaction system.
The database is designed to store user information, transaction records, transaction categories, tags, and system logs while ensuring data accuracy, security, and integrity.

The goal of this database is to simulate real-world Mobile Money operations such as sending money, cash out, bill payments, and merchant payments

2. Entity Relationship Diagram

Our entity relationship diagram illustrates all the entities and attributes we used for the MoMo SMS data processing system. We used some key entities such as Users, Transaction, Transaction_categories, system_logs, Tags, and transaction_tags. It also indicates all the relationships between the entities.

The diagram below provides an overview of how the database is structured and how different entities are connected.



The diagram also illustrates the relationships between entities, such as the one-to-many relationship between Users and Transactions, where a user can participate in multiple transactions, and the one-to-many relationship between Transaction Categories and Transactions. Primary keys are used to uniquely identify records, while foreign keys establish relationships between entities and ensure data integrity. Overall, the ERD provides a normalized and scalable database design that supports efficient storage, querying, and analysis of MoMo transaction data.

2.2 Entity Description

- **Users:** Stores information about registered Mobile Money users.
- **Transactions:** Stores all financial transactions performed by users.

- **Transaction Categories**: Defines the type of transaction (e.g. Cash Out, Deposit).
- **Tags:** Used to label transactions (e.g. Fraud suspected, High value).
- **Transaction Tags:** A junction table that links transactions with tags.
- **System Logs:** Records transaction actions and status changes.

3. Design Rationale and Justification

The database design follows normalization principles to avoid data redundancy and ensure consistency.

Key Design Decisions:

- Separate tables were created for users, transactions, and categories to improve data organization.
- Foreign keys were used to enforce relationships between tables.
- A many-to-many relationship between transactions and tags was resolved using the transaction_tags table.
- System logs were stored in a separate table to track transaction history without duplicating data

This design makes the database **scalable**, **secure**, and easy to maintain

4. Data Dictionary

4.1 Users Table

Column name	**Data type**	**Description**
user_id	INT	Unique identifier for each user
full_name	VARCHAR(100)	User's full name
phone_number	VARCHAR(100)	Unique phone number
id_number	VARCHAR(100)	National ID number
registration-date	DATETIME	Data user registered
status	VARCHAR(20)	User account status
created_at	DATETIME	Record creation time

4.2 Transactions Table

Column name	Data type	Description
tranaction_id	INT	Unique transaction id
sender_id	INT	User sending money
receiver_id	INT	User receiving money
category_id	INT	Transaction category
amount	DECIMAL(10, 20	Transaction amount
currency	CHAR(3)	Currency used
transaction_type	VARCHAR(50)	Type of transaction
transaction_status	ENUM	Status of transaction
created_at	DATETIME	Time of transaction

4.3 Transaction Categories Table

Column name	Data type	Description
category_id	INT	Unique category ID
category_name	VARCHAR(50)	Name of category
description	VARCHAR(255)	Category description

4.4 Tags Table

Column name	Data type	Description
tag_id	INT	Unique tag ID
tag_name	VARCHAR(50)	Tag name
description	VARCHAR(255)	Tag description

4.5 Transaction Logs Table

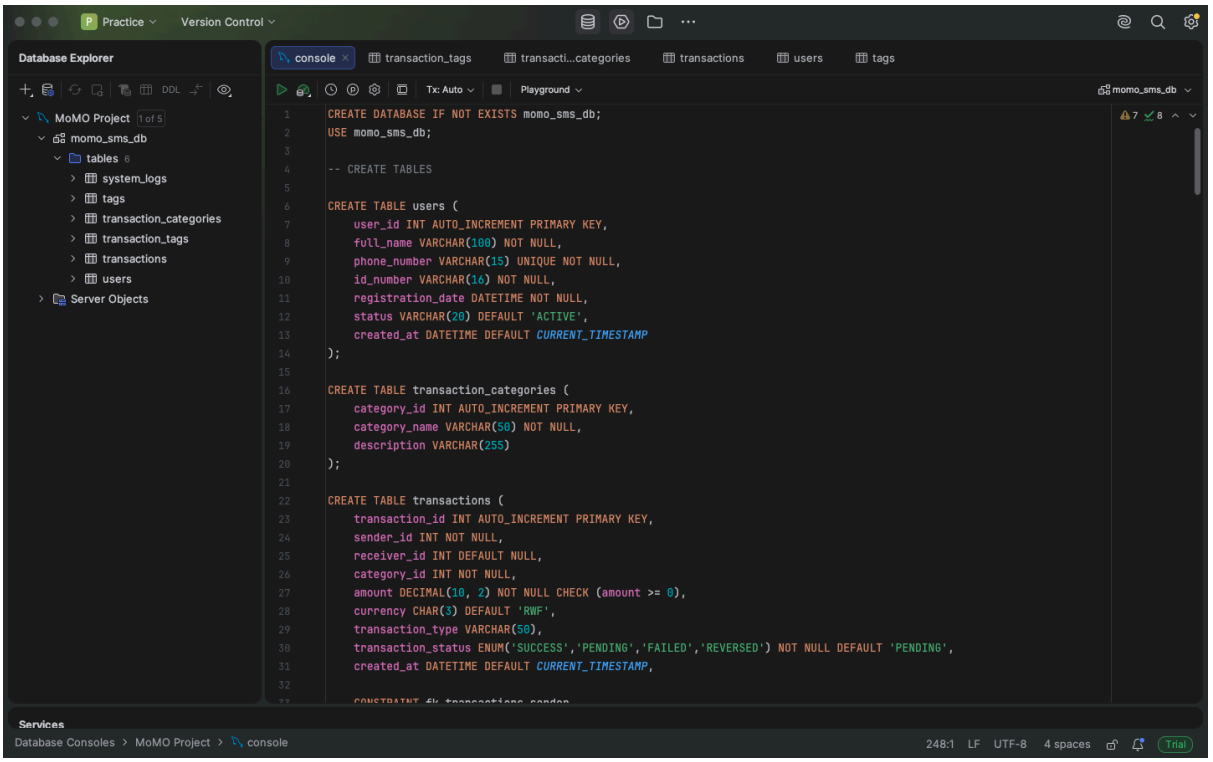
Column name	Data type	Description
transaction_id	INT	Linked transaction
tag_id	INT	Linked Tag

4.6 System Logs Table

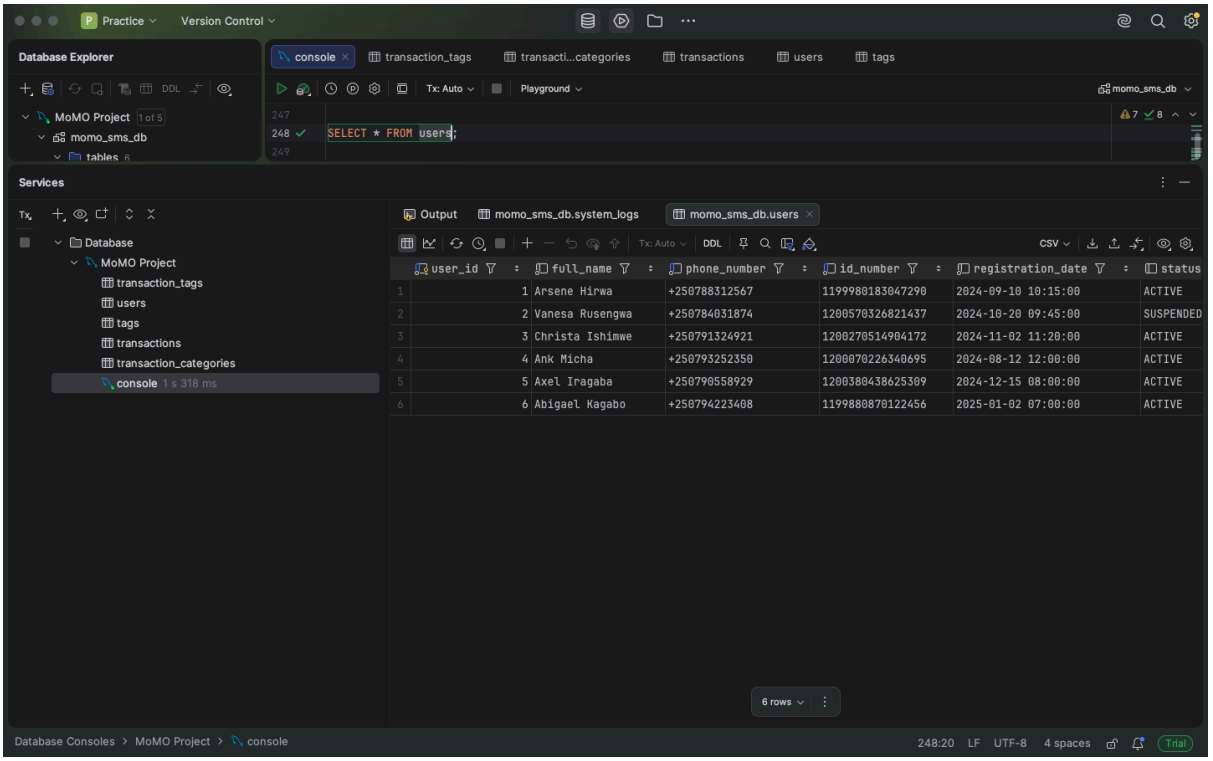
Column name	Data type	Description
log_id	INT	Unique log ID
transaction_id	INT	Related transaction
action	VARCHAR(20)	Action performed
log_time	DATETIME	Time of action
status_message	VARCHAR(255)	Log description

5. CRUD Operations

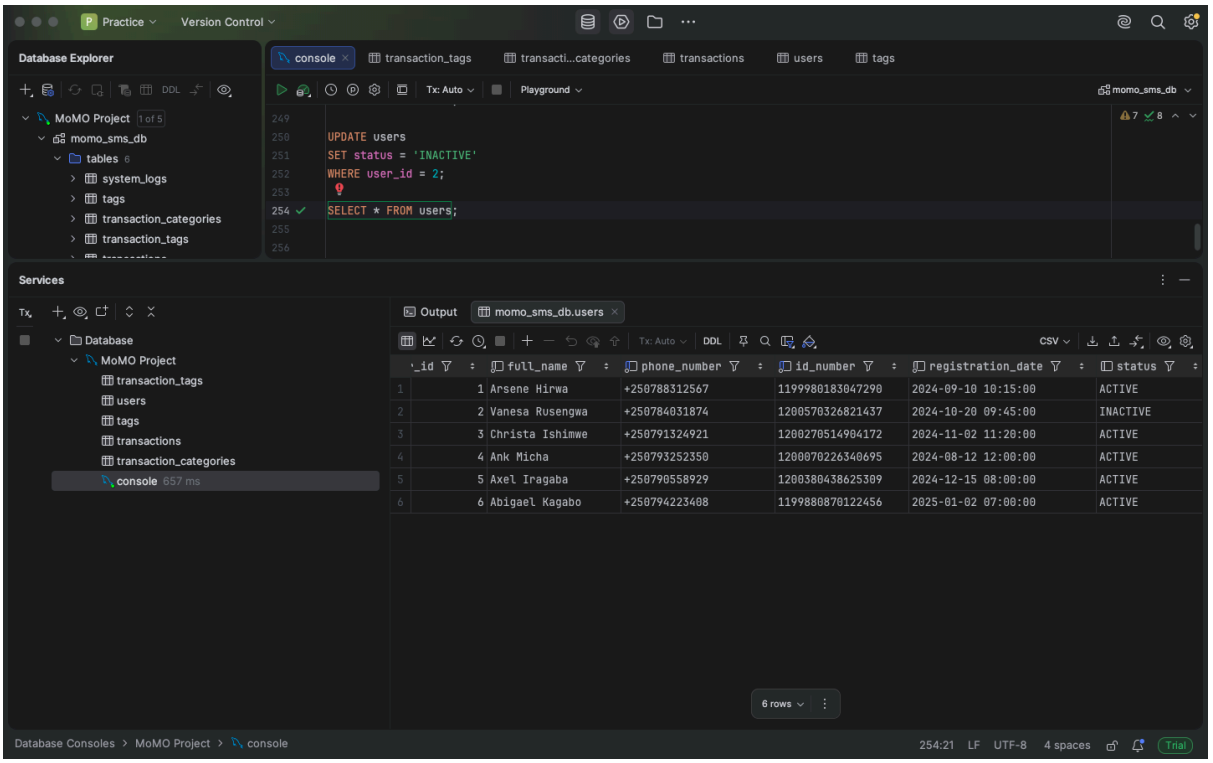
Create:



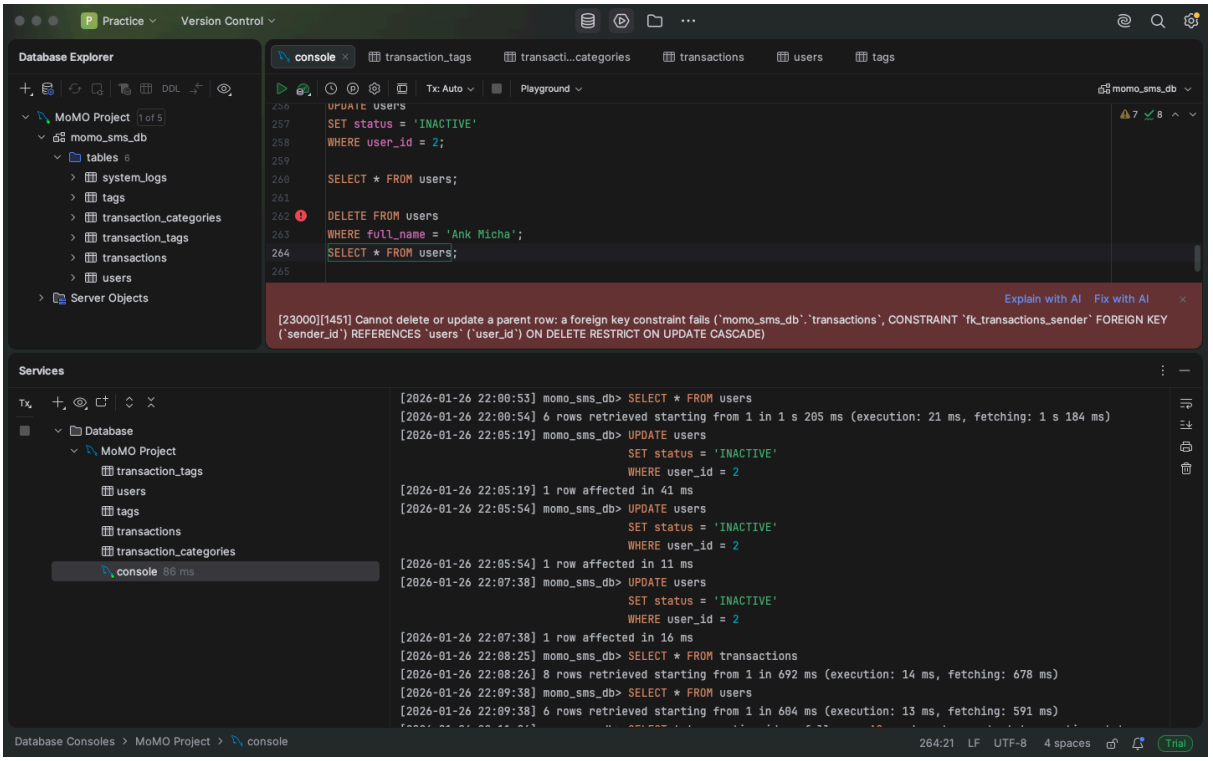
Select:



Update:

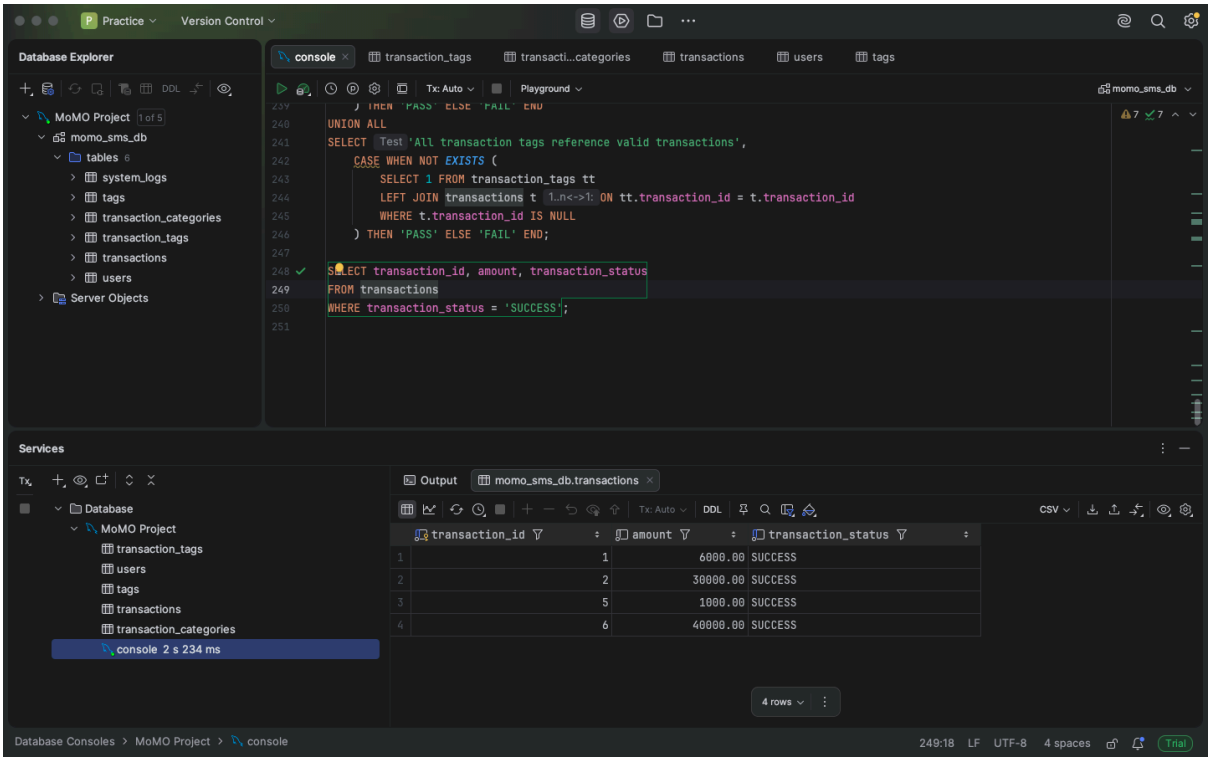

Practice
Version Control
Database Explorer
console
transaction_tags
transacti...categories
transactions
users
tags
MoMO Project 1 of 5
momo_sms_db
tables 6
system_logs
tags
transaction_categories
transaction_tags
Tx: Auto
Playground
momo_sms_db
UPDATE users
SET status = 'INACTIVE'
WHERE user_id = 2;
SELECT * FROM users;
Services
Database
MoMO Project
transaction_tags
users
tags
transactions
transaction_categories
console 657 ms
Output
momo_sms_db.users
DDL
CSV
_id
full_name
phone_number
id_number
registration_date
status
1 1 Arsene Hirwa +250788312567 1199980183047290 2024-09-10 10:15:00 ACTIVE
2 2 Vanesa Rusengwa +250784031874 1200570326821437 2024-10-20 09:45:00 INACTIVE
3 3 Christa Ishimwe +250791324921 1200270514904172 2024-11-02 11:20:00 ACTIVE
4 4 Ank Micha +250793252350 1200070226340695 2024-08-12 12:00:00 ACTIVE
5 5 Axel Iragaba +250790558929 1200380438625309 2024-12-15 08:00:00 ACTIVE
6 6 Abigael Kagabo +250794223408 1199880870122456 2025-01-02 07:00:00 ACTIVE
6 rows
Database Consoles
MoMO Project
console
254:21
LF
UTF-8
4 spaces
Trial

Delete: The database prevented deleting a user because that user is already referenced as a sender in the transaction table.

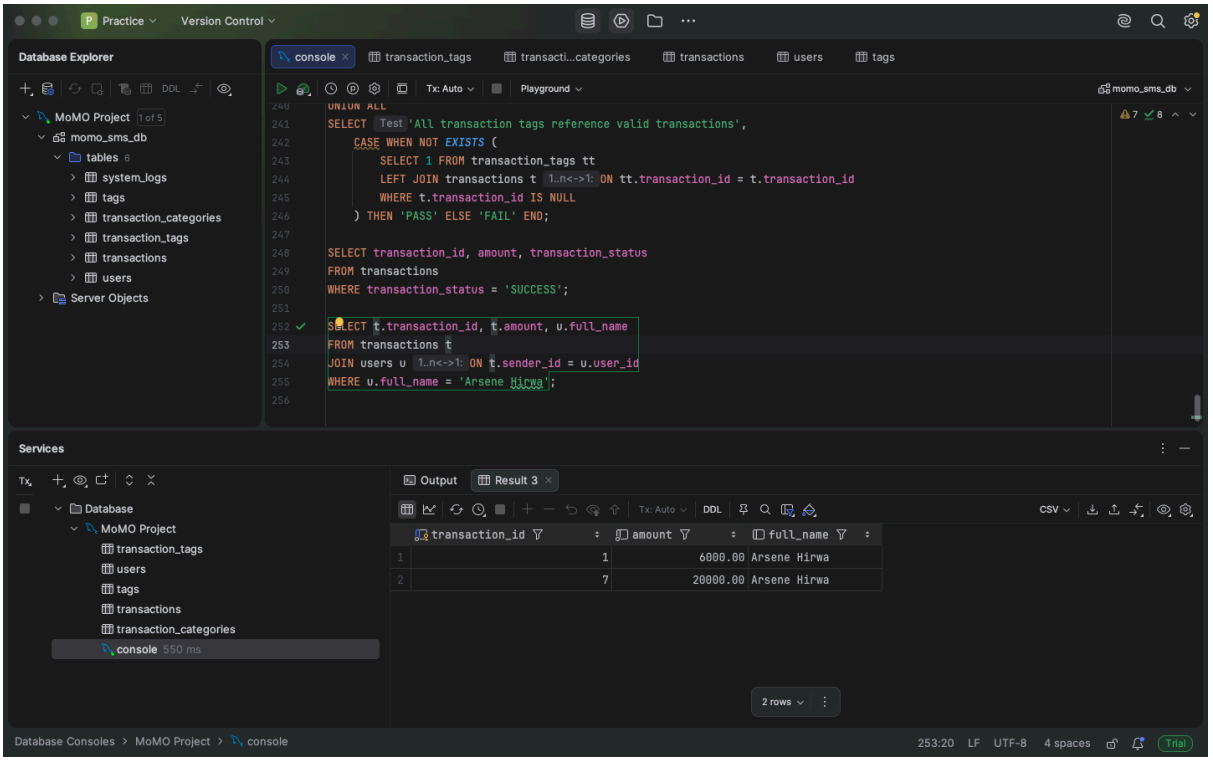


5. Sample Queries Demonstrating Functionality

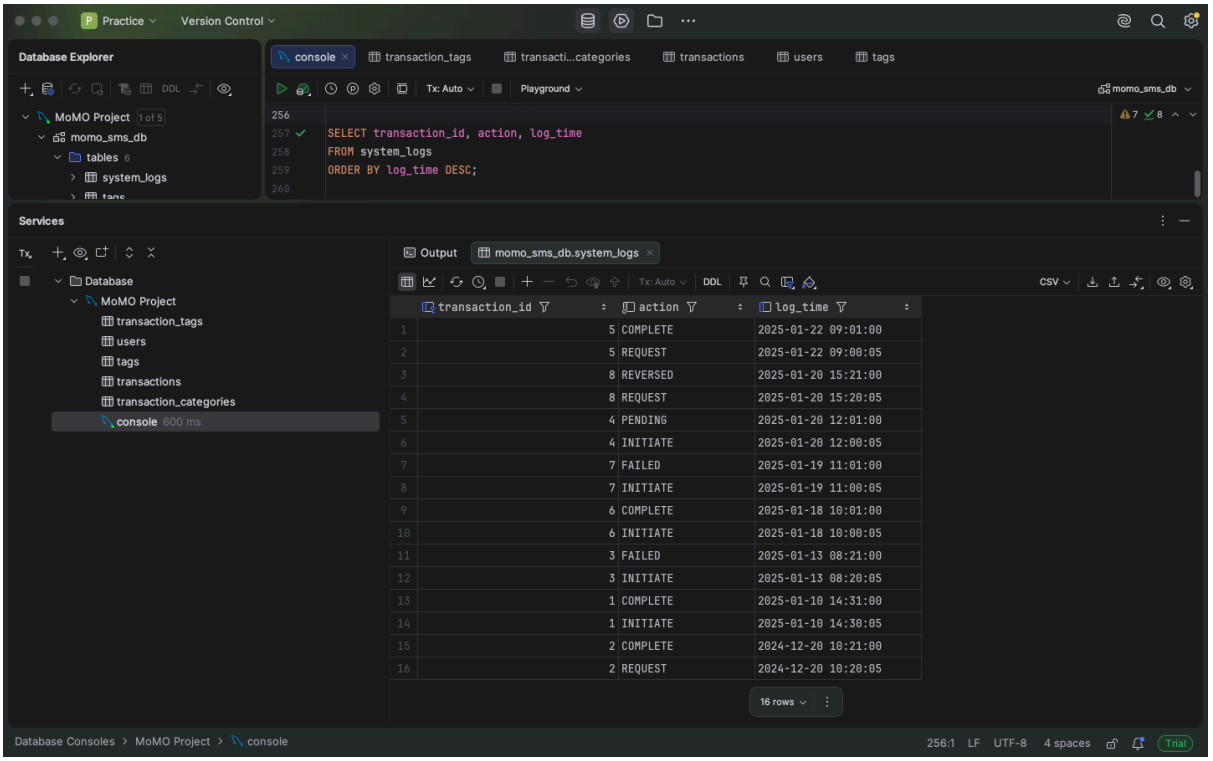
Query One: View all successful transactions



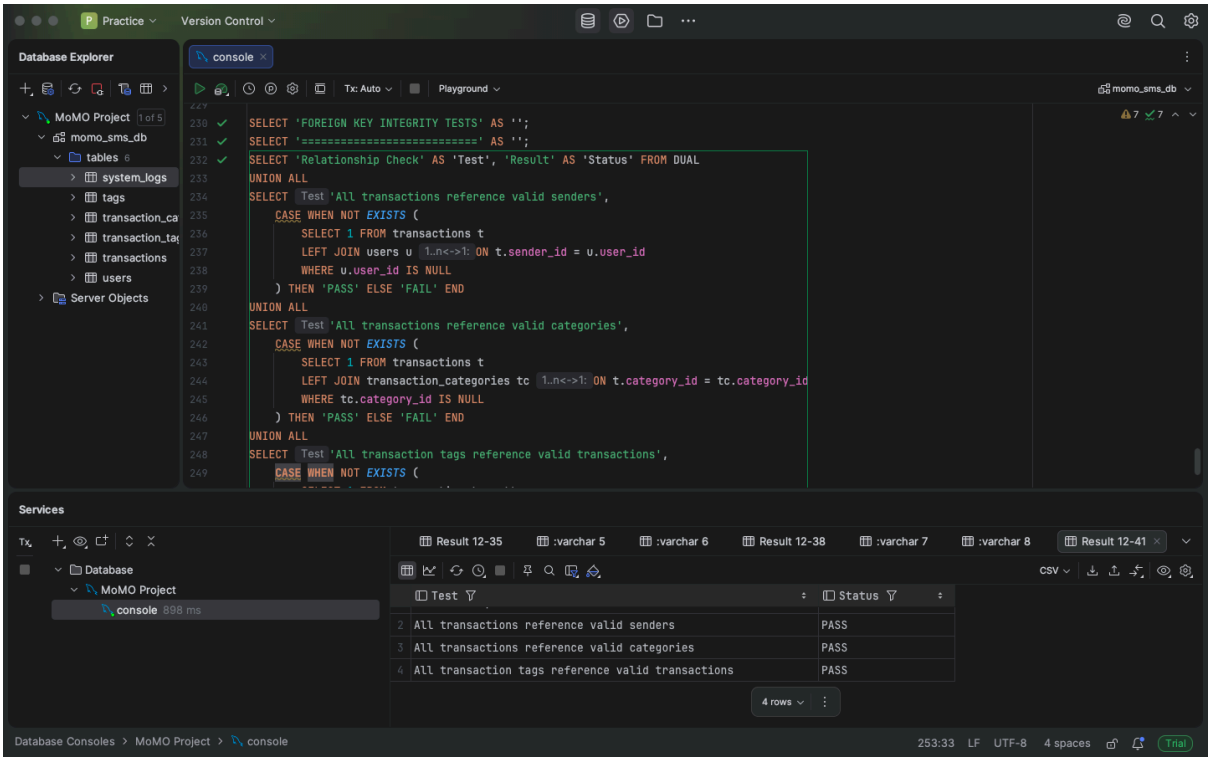
Query Two: View transactions sent by a specific user



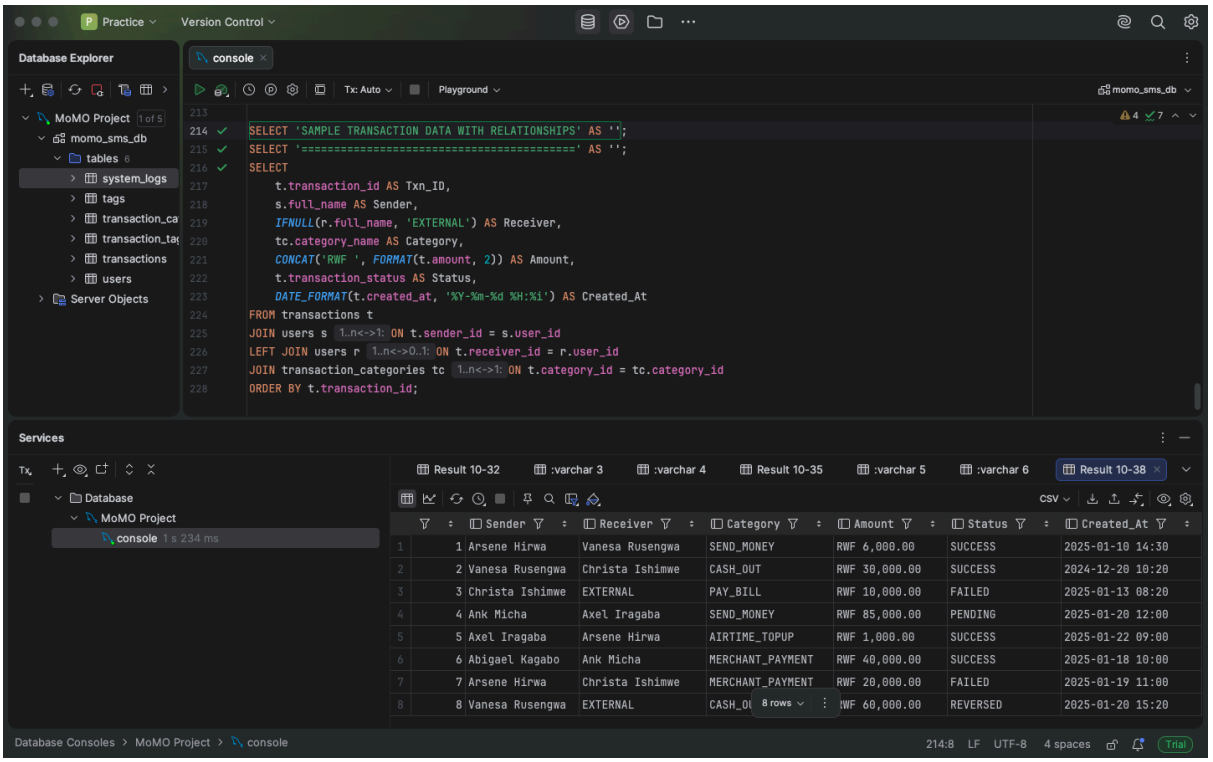
Query Three: View Transaction Logs



Foreign Key Integrity Tests (screenshot)



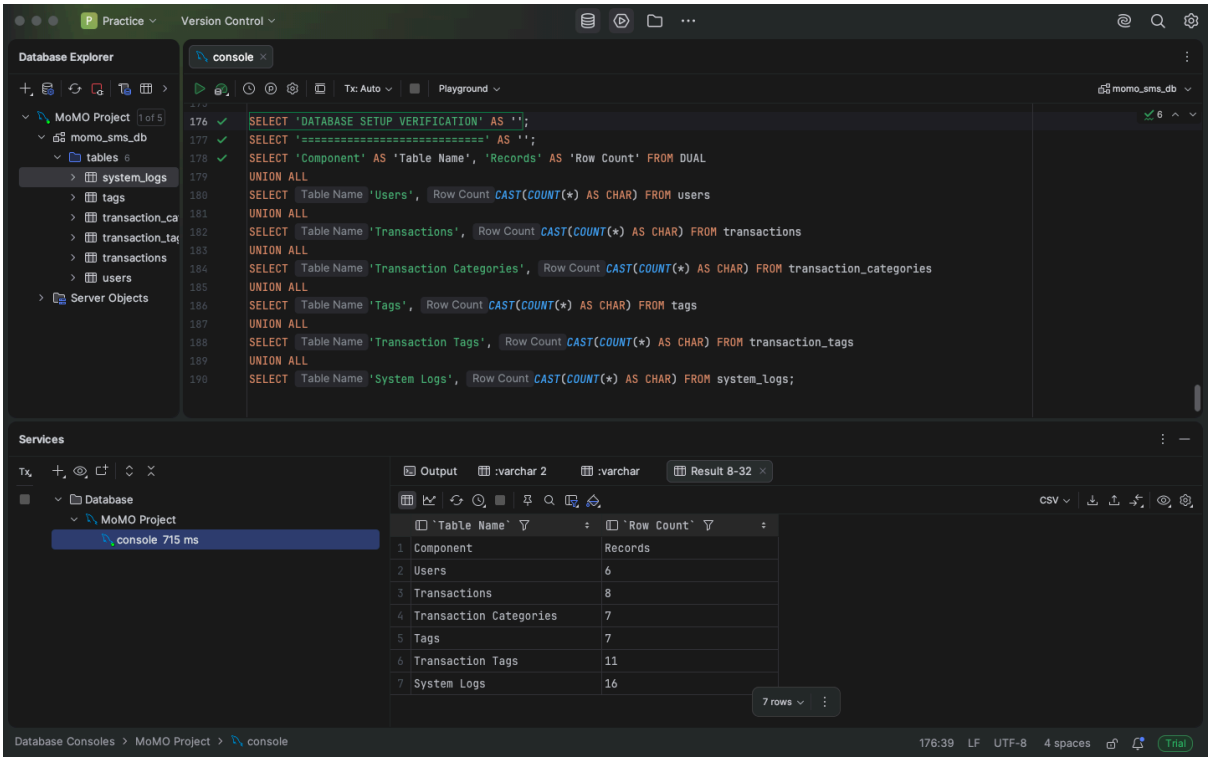
Sample Transaction Query With JOINs(screenshot)



```
SELECT 'SAMPLE TRANSACTION DATA WITH RELATIONSHIPS' AS '';
SELECT '==========' AS '';
SELECT
    t.transaction_id AS Txn_ID,
    s.full_name AS Sender,
    IFNULL(r.full_name, 'EXTERNAL') AS Receiver,
    tc.category_name AS Category,
    CONCAT('RWF ', FORMAT(t.amount, 2)) AS Amount,
    t.transaction_status AS Status,
    DATE_FORMAT(t.created_at, '%Y-%m-%d %H:%i') AS Created_At
FROM transactions t
JOIN users s ON t.sender_id = s.user_id
LEFT JOIN users r ON t.receiver_id = r.user_id
JOIN transaction_categories tc ON t.category_id = tc.category_id
ORDER BY t.transaction_id;
```

	Sender	Receiver	Category	Amount	Status	Created_At
1	Arsene Hirwa	Vanesa Rusengwa	SEND_MONEY	RWF 6,000.00	SUCCESS	2025-01-10 14:30
2	Vanesa Rusengwa	Christa Ishimwe	CASH_OUT	RWF 30,000.00	SUCCESS	2024-12-20 10:20
3	Christa Ishimwe	EXTERNAL	PAY_BILL	RWF 10,000.00	FAILED	2025-01-13 08:20
4	Ank Micha	Axel Iragaba	SEND_MONEY	RWF 85,000.00	PENDING	2025-01-20 12:00
5	Axel Iragaba	Arsene Hirwa	AIRTIME_TOPUP	RWF 1,000.00	SUCCESS	2025-01-22 09:00
6	Abigael Kagabo	Ank Micha	MERCHANT_PAYMENT	RWF 40,000.00	SUCCESS	2025-01-18 10:00
7	Arsene Hirwa	Christa Ishimwe	MERCHANT_PAYMENT	RWF 20,000.00	FAILED	2025-01-19 11:00
8	Vanesa Rusengwa	EXTERNAL	CASH_O	WF 60,000.00	REVERSED	2025-01-20 15:20

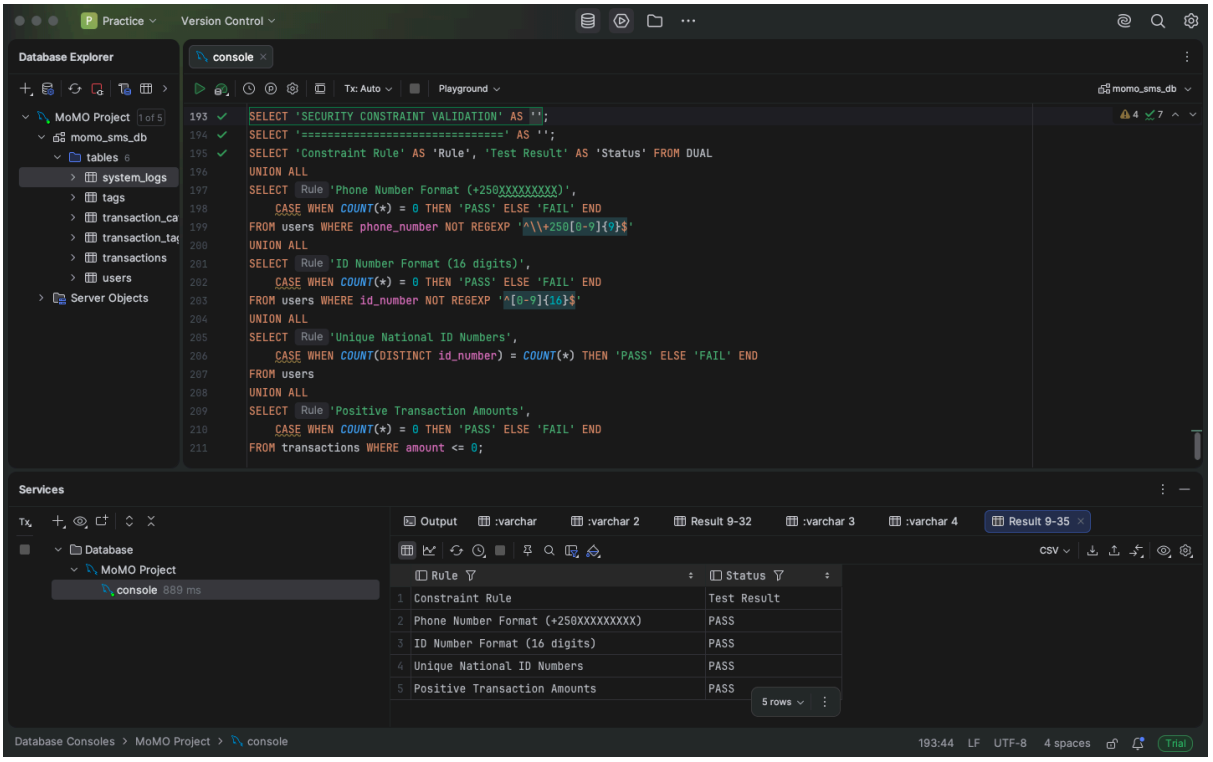
Database Setup Verification(screenshot)



```
SELECT 'DATABASE SETUP VERIFICATION' AS '';
SELECT '==========' AS '';
SELECT 'Component' AS 'Table Name', 'Records' AS 'Row Count' FROM DUAL
UNION ALL
SELECT 'Users', CAST(COUNT(*) AS CHAR) FROM users
UNION ALL
SELECT 'Transactions', CAST(COUNT(*) AS CHAR) FROM transactions
UNION ALL
SELECT 'Transaction Categories', CAST(COUNT(*) AS CHAR) FROM transaction_categories
UNION ALL
SELECT 'Tags', CAST(COUNT(*) AS CHAR) FROM tags
UNION ALL
SELECT 'Transaction Tags', CAST(COUNT(*) AS CHAR) FROM transaction_tags
UNION ALL
SELECT 'System Logs', CAST(COUNT(*) AS CHAR) FROM system_logs;
```

	`Table Name`	`Row Count`
1	Component	Records
2	Users	6
3	Transactions	8
4	Transaction Categories	7
5	Tags	7
6	Transaction Tags	11
7	System Logs	16

Security And Constraints Validation(screenshot)



Conclusion

The Momo SMS database efficiently manages Mobile money transactions by organizing users, transactions, categories, tags, and system logs into a secure and relational structure. With enforced constraints, foreign keys, and indexes, it ensures data integrity, traceability, and reliable reporting. This design models real-world Mobile money operations accurately while remaining scalable, maintainable, and easy to analyze.