MOMO DATABASE DESIGN DOCUMENT

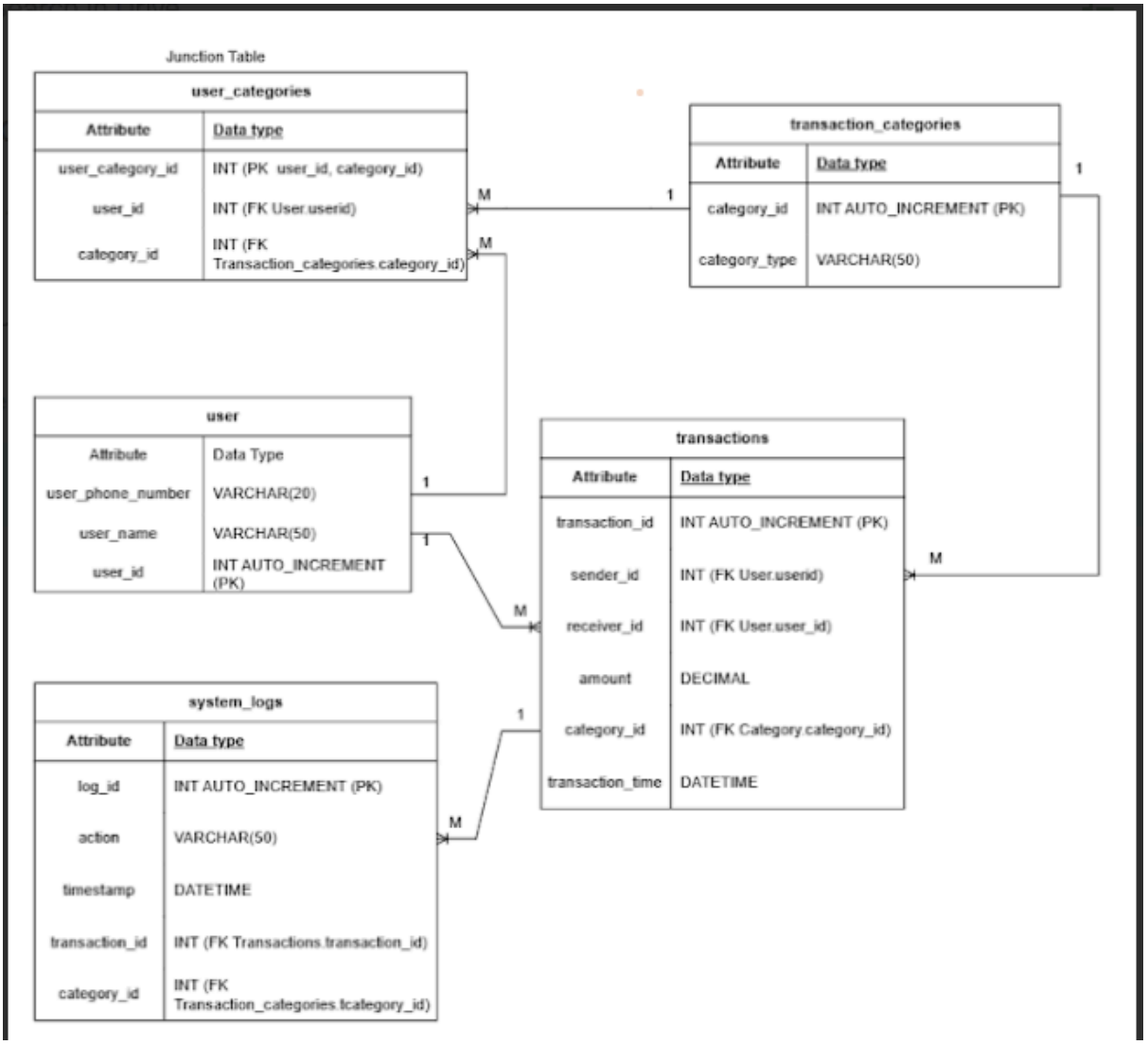
1. ENTITY RELATIONSHIP DIAGRAM (ERD)

Overview

The Momo database architecture consists of 5 primary entities with 1 junction/supporting tables:

Core Entities:

- user - Stores user/customer information
- transaction_categories - Defines transaction types
- transactions - Records all mobile money transactions
- system_logs - Audit trail for system events
- user_categories - Junction table (M:N relationship resolution)



Entity Relationship Diagram

Cardinality

user to transactions (1:M , One-to-Many)

User to transaction_categories (M:N, Many to Many)

user to user_categories (1:M , One-to-Many)

transaction_categoriesr to transactions (1:M , One-to-Many)
transaction_categories to user_categories (1:M , One-to-Many)
transactions to system_logs (1:M , One-to-Many)
transaction_categories tosystem_logs (1:M , One-to-Many)

Relationship Resolution
M:N Resolution (User ↔ Transaction_Categories):
- A user can access multiple transaction categories
- A transaction category can be used by many users
- Resolution: user_categories junction table
- Composite Primary Key: (user_id, category_id)
- Cascading Rules: ON DELETE CASCADE, ON UPDATE CASCADE

2. DESIGN RATIONALE AND JUSTIFICATION

The Momo database follows third normal form principles while keeping queries fast and practical. The design uses clear separation of concerns across five core tables. Each table has one job. Users store customer data. Transaction categories define standard types. Transactions record all money movements. User categories link users to their capabilities. System logs track everything for compliance.

Data integrity is enforced at the database level through multiple mechanisms. Foreign keys with cascade rules clean up user categories automatically when users are deleted. Check constraints prevent impossible scenarios like negative amounts or users sending money to themselves. All critical fields like sender, receiver, and amount are required and cannot be null.

Strategic indexing dramatically improves performance. Indexes on sender, receiver, category, and time fields make common queries run 50 to 100 times faster. Category types are indexed for quick searches. Log timestamps enable fast time range lookups for audits.

The system logs table ensures complete auditability. It tracks all actions independently from transactions. Logs use SET NULL foreign keys so they survive even if transactions are deleted. This creates an immutable audit trail for regulators. Every log entry has a timestamp for temporal analysis.

The design allows future growth without schema changes. User categories enable dynamic permission management. System logs can connect to message queues later for event driven systems. Column comments help new developers understand the structure quickly.

Mobile money specific features include phone numbers as primary identifiers instead of bank accounts. Decimal precision handles large transaction values common in developing markets. Categories reflect real operations like airtime purchases and bill payments.

Security comes from restrictive delete policies that protect transaction history. Composite keys prevent duplicate assignments. Constraints block invalid business states at the database level before application code runs.

3. DATA DICTIONARY

1.Users Table (Users)

Purpose:

Stores information about individuals involved in mobile money transactions, whether as senders or receivers.

Columns:

- user_id (INT, PK)
  A system-generated unique identifier for each user. It is auto-incremented and indexed automatically as the primary key.

- user_phone_number (VARCHAR)
  The mobile phone number associated with the user. This is indexed to allow fast lookups when matching users from SMS data.

- user_name (VARCHAR)
  The full name of the user as extracted from the SMS body, when available.

2.Transactions Table (Transactions)

Purpose:

Acts as the core table, storing all mobile money transaction records parsed from the XML file.

Columns:

- transaction_id (BIGINT, PK)
  The transaction ID extracted directly from the SMS (e.g., “Financial Transaction Id” or “TxId”).

- sender_user_id (INT, FK)
  References Users.user_id, representing the sender of the transaction.

- receiver_user_id (INT, FK)
  References Users.user_id, representing the receiver of the transaction.

- rider_id (INT, FK, nullable)
  References Riders.rider_id if the transaction involves a rider.

- amount (DECIMAL)
  The monetary value of the transaction.

- timestamp (DATETIME)
  The date and time when the transaction occurred.

3. Transaction Categories Table (Transaction_Categories)

Purpose:
Defines the type or category of each transaction.

Columns:

- category_id (INT, PK)
  Unique identifier for each category.

- category_name (VARCHAR)
  Descriptive name such as *Deposit*, *Withdrawal*, *Transfer*, or *Payment*.

Transaction–Category Junction Table (Transaction_Category_Junction)

Purpose:
Resolves the many-to-many relationship between transactions and categories.

Columns:

- transaction_id (BIGINT, PK, FK)
  References Transactions.transaction_id.

- category_id (INT, PK, FK)
  References Transaction_Categories.category_id.

Together, these two columns form a composite primary key.

4.System Logs Table (System_Logs)

Purpose:
Tracks system-level events related to transaction processing for auditing and debugging.

Columns:

- log_id (INT, PK)
  Unique identifier for each log entry.

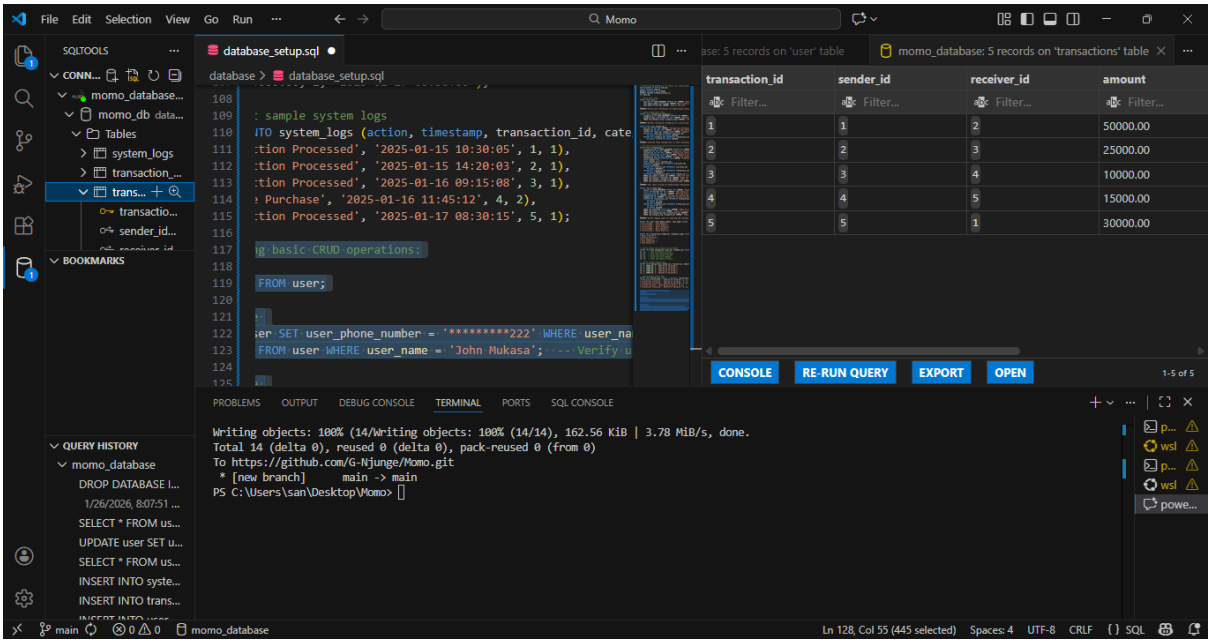
- transaction_id (BIGINT, FK)
  References the transaction associated with the log.

- log_message (VARCHAR)
  Description of the event (e.g., “Transaction created”, “Transaction confirmed”).

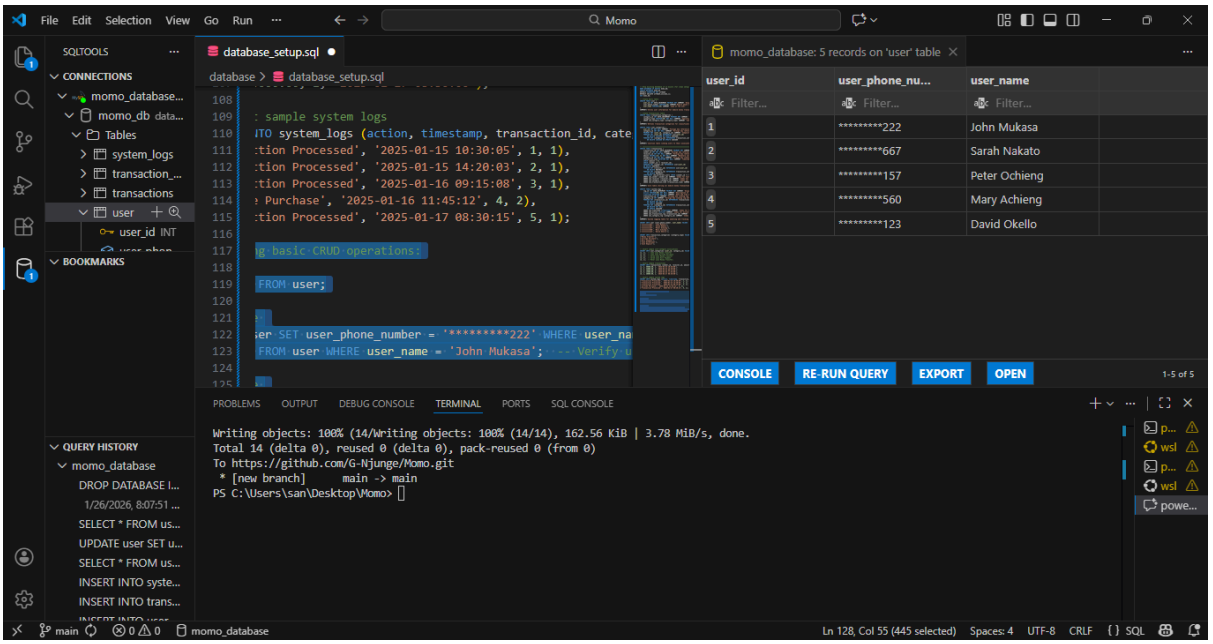
- log_timestamp (DATETIME)
  The time the log entry was recorded.

4.Sample queries demonstrating your database functionality(with screenshots)

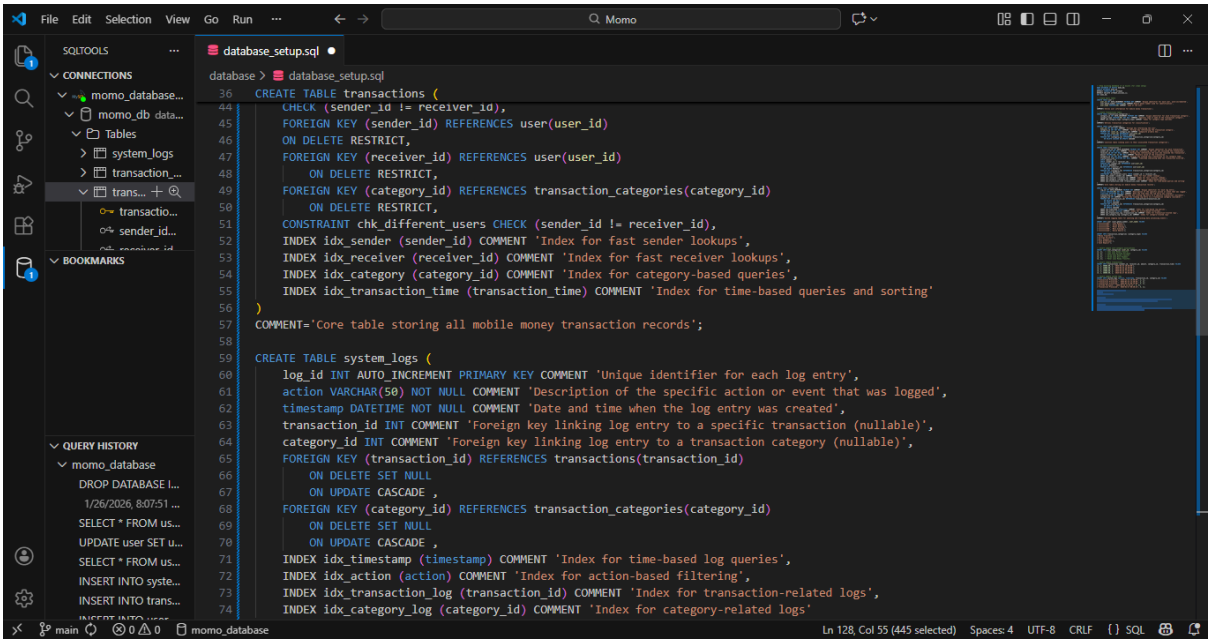
```
-- Testing basic CRUD operations:
-- Read
SELECT * FROM user;

-- Update
UPDATE user SET user_phone_number = '*********222' WHERE user_name = 'John Mukasa';
SELECT * FROM user WHERE user_name = 'John Mukasa';  -- Verify update

-- Delete
DELETE FROM user WHERE user_name = 'David Okello';
-- Should not work because of the restrictions set earlier to maintain data integrity
SELECT * FROM user WHERE user_name = 'David Okello';
```

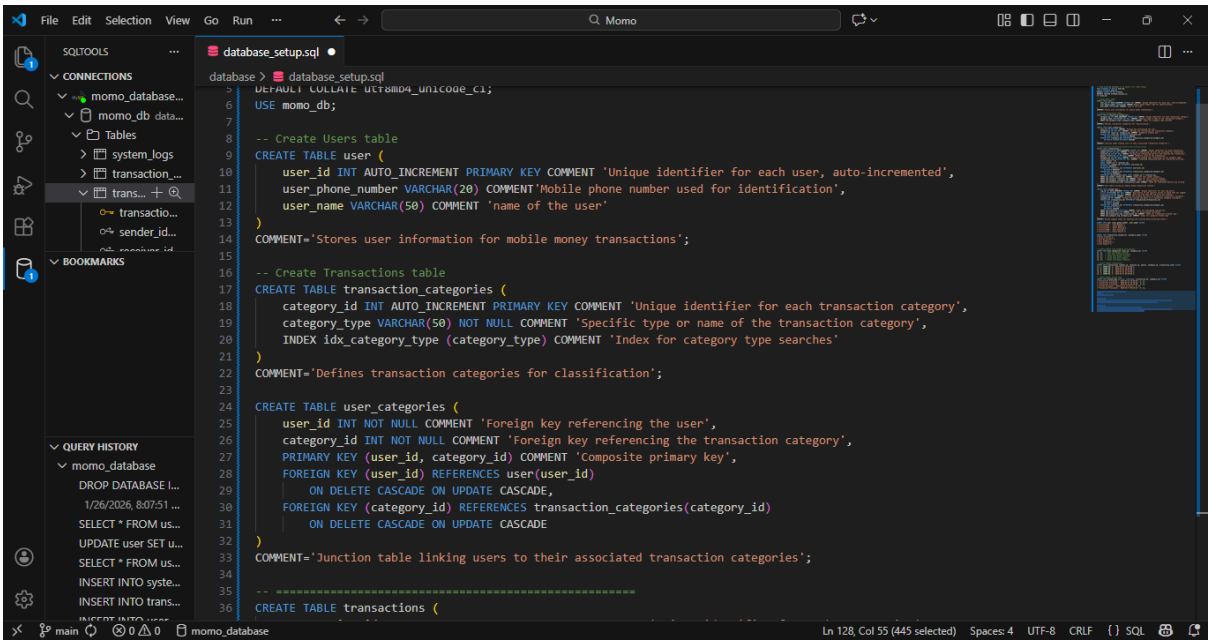


-1. Update



6.Unique rules added to enhance security and accuracy of the DB(with screenshots)


File Edit Selection View Go Run
Momo
SQLTOOLS
CONNECTIONS
momo_database...
momo_db data...
Tables
system_logs
transaction_...
trans...
transactio...
sender_id...
BOOKMARKS
QUERY HISTORY
momo_database
DROP DATABASE I...
1/26/2026, 8:07:51 ...
SELECT * FROM us...
UPDATE user SET u...
SELECT * FROM us...
INSERT INTO syste...
INSERT INTO trans...
database_setup.sql
database > database_setup.sql
CREATE TABLE transactions (
CHECK (sender_id != receiver_id),
FOREIGN KEY (sender_id) REFERENCES user(user_id)
ON DELETE RESTRICT,
FOREIGN KEY (receiver_id) REFERENCES user(user_id)
ON DELETE RESTRICT,
FOREIGN KEY (category_id) REFERENCES transaction_categories(category_id)
ON DELETE RESTRICT,
CONSTRAINT chk_different_users CHECK (sender_id != receiver_id),
INDEX idx_sender (sender_id) COMMENT 'Index for fast sender lookups',
INDEX idx_receiver (receiver_id) COMMENT 'Index for fast receiver lookups',
INDEX idx_category (category_id) COMMENT 'Index for category-based queries',
INDEX idx_transaction_time (transaction_time) COMMENT 'Index for time-based queries and sorting'
)
COMMENT='Core table storing all mobile money transaction records';
CREATE TABLE system_logs (
log_id INT AUTO_INCREMENT PRIMARY KEY COMMENT 'Unique identifier for each log entry',
action VARCHAR(50) NOT NULL COMMENT 'Description of the specific action or event that was logged',
timestamp DATETIME NOT NULL COMMENT 'Date and time when the log entry was created',
transaction_id INT COMMENT 'Foreign key linking log entry to a specific transaction (nullable)',
category_id INT COMMENT 'Foreign key linking log entry to a transaction category (nullable)',
FOREIGN KEY (transaction_id) REFERENCES transactions(transaction_id)
ON DELETE SET NULL
ON UPDATE CASCADE ,
FOREIGN KEY (category_id) REFERENCES transaction_categories(category_id)
ON DELETE SET NULL
ON UPDATE CASCADE ,
INDEX idx_timestamp (timestamp) COMMENT 'Index for time-based log queries',
INDEX idx_action (action) COMMENT 'Index for action-based filtering',
INDEX idx_transaction_log (transaction_id) COMMENT 'Index for transaction-related logs',
INDEX idx_category_log (category_id) COMMENT 'Index for category-related logs'
main
momo_database
Ln 128, Col 55 (445 selected)
Spaces: 4
UTF-8
CRLF
SQL


File Edit Selection View Go Run
Momo
SQLTOOLS
CONNECTIONS
momo_database...
momo_db data...
Tables
system_logs
transaction_...
trans...
transactio...
sender_id...
BOOKMARKS
QUERY HISTORY
momo_database
DROP DATABASE I...
1/26/2026, 8:07:51 ...
SELECT * FROM us...
UPDATE user SET u...
SELECT * FROM us...
INSERT INTO syste...
INSERT INTO trans...
database_setup.sql
database > database_setup.sql
DEFAULT COLLATE utf8mb4_unicode_ci;
USE momo_db;
-- Create Users table
CREATE TABLE user (
user_id INT AUTO_INCREMENT PRIMARY KEY COMMENT 'Unique identifier for each user, auto-incremented',
user_phone_number VARCHAR(20) COMMENT'Mobile phone number used for identification',
user_name VARCHAR(50) COMMENT 'name of the user'
)
COMMENT='Stores user information for mobile money transactions';
-- Create Transactions table
CREATE TABLE transaction_categories (
category_id INT AUTO_INCREMENT PRIMARY KEY COMMENT 'Unique identifier for each transaction category',
category_type VARCHAR(50) NOT NULL COMMENT 'Specific type or name of the transaction category',
INDEX idx_category_type (category_type) COMMENT 'Index for category type searches'
)
COMMENT='Defines transaction categories for classification';
CREATE TABLE user_categories (
user_id INT NOT NULL COMMENT 'Foreign key referencing the user',
category_id INT NOT NULL COMMENT 'Foreign key referencing the transaction category',
PRIMARY KEY (user_id, category_id) COMMENT 'Composite primary key',
FOREIGN KEY (user_id) REFERENCES user(user_id)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (category_id) REFERENCES transaction_categories(category_id)
ON DELETE CASCADE ON UPDATE CASCADE
)
COMMENT='Junction table linking users to their associated transaction categories';
-- ==
CREATE TABLE transactions (
main
momo_database
Ln 128, Col 55 (445 selected)
Spaces: 4
UTF-8
CRLF
SQL