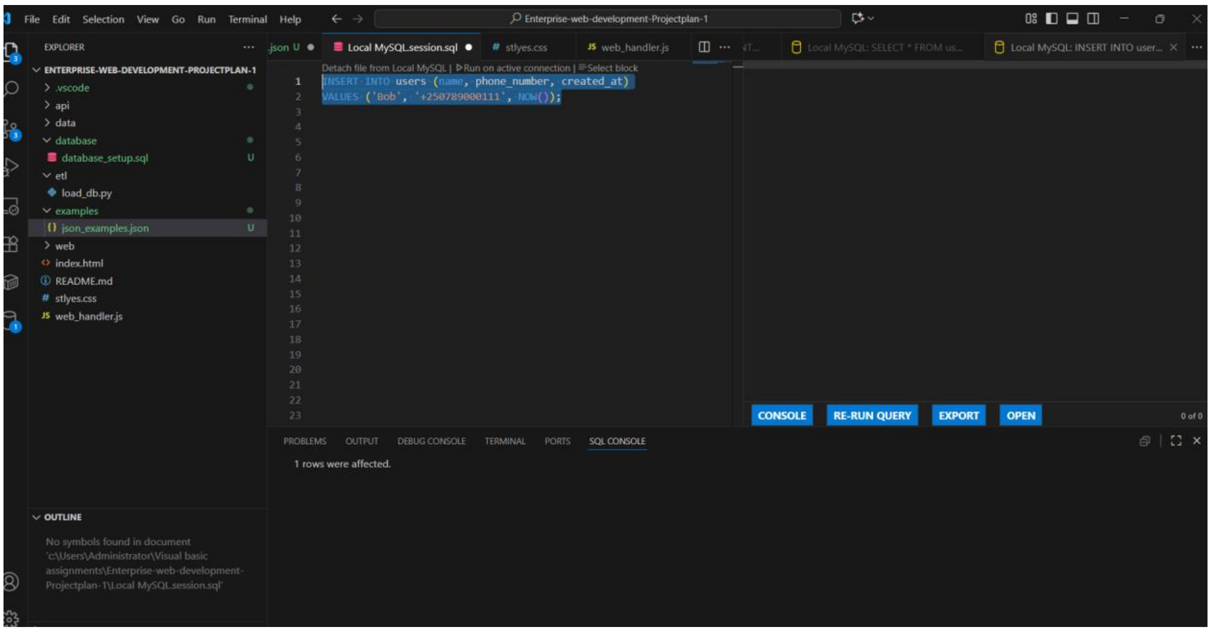
CRUD Tests

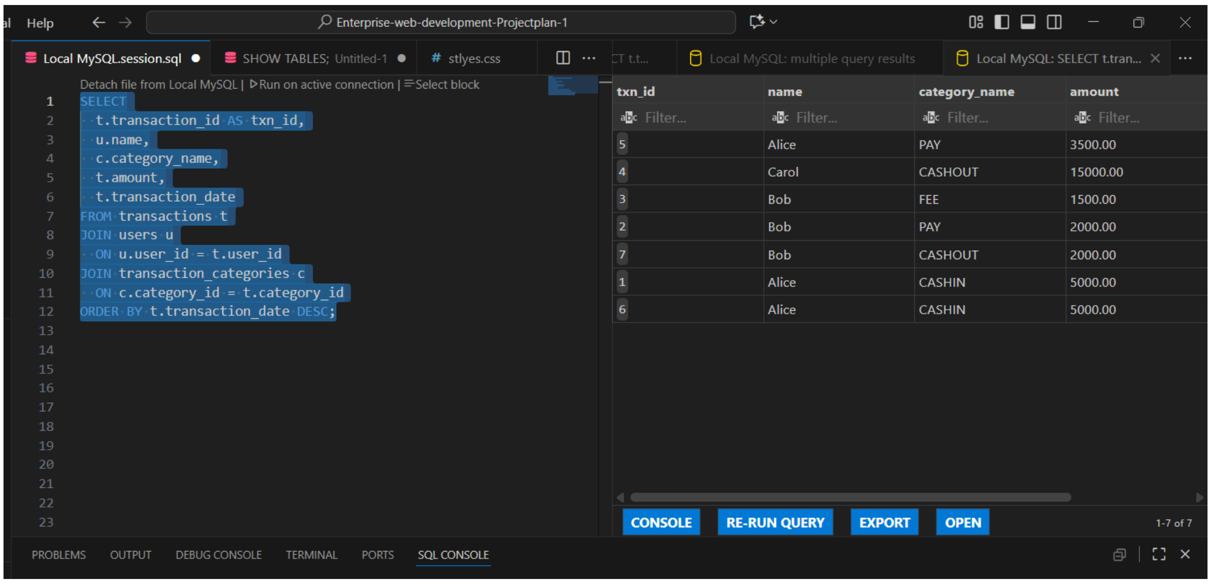
INSERT a user → "1 row affected" (screenshot).



```sql
INSERT INTO users (name, phone_number, created_at)
VALUES ('Bob', '+250789000111', NOW());
```

```
1 rows were affected.
```

SELECT join (users + categories + transactions) → results grid (screenshot).

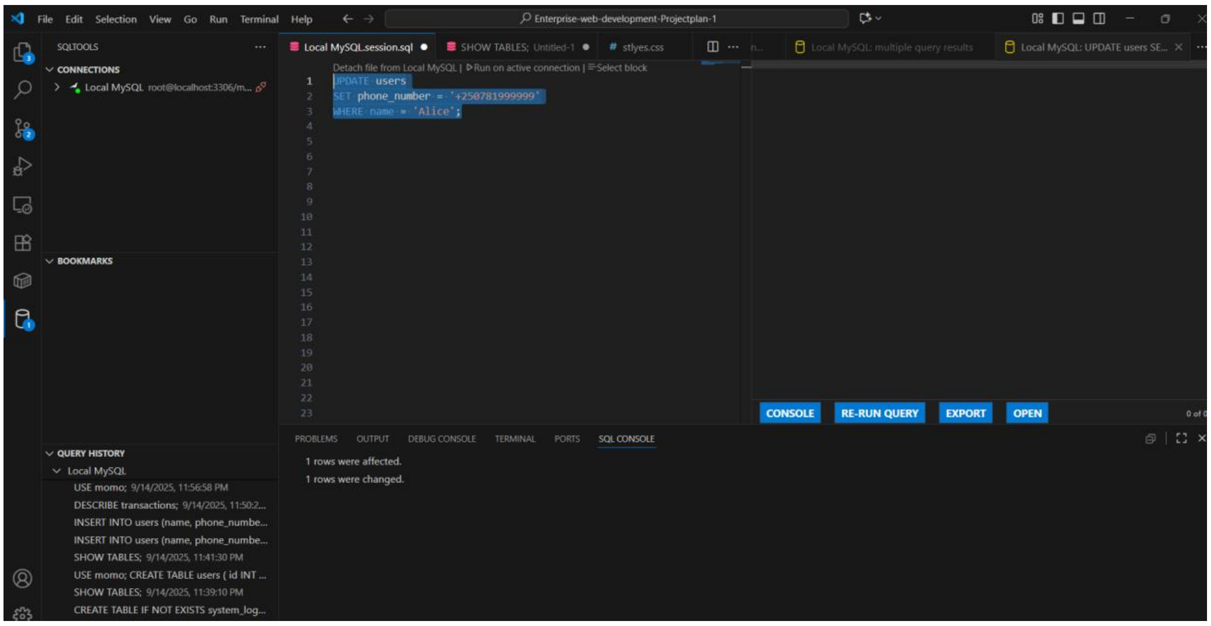


```sql
SELECT
    t.transaction_id AS txn_id,
    u.name,
    c.category_name,
    t.amount,
    t.transaction_date
FROM transactions t
JOIN users u
    ON u.user_id = t.user_id
JOIN transaction_categories c
    ON c.category_id = t.category_id
ORDER BY t.transaction_date DESC;
```

txn_id	name	category_name	amount
5	Alice	PAY	3500.00
4	Carol	CASHOUT	15000.00
3	Bob	FEE	1500.00
2	Bob	PAY	2000.00
7	Bob	CASHOUT	2000.00
1	Alice	CASHIN	5000.00
6	Alice	CASHIN	5000.00

1-7 of 7

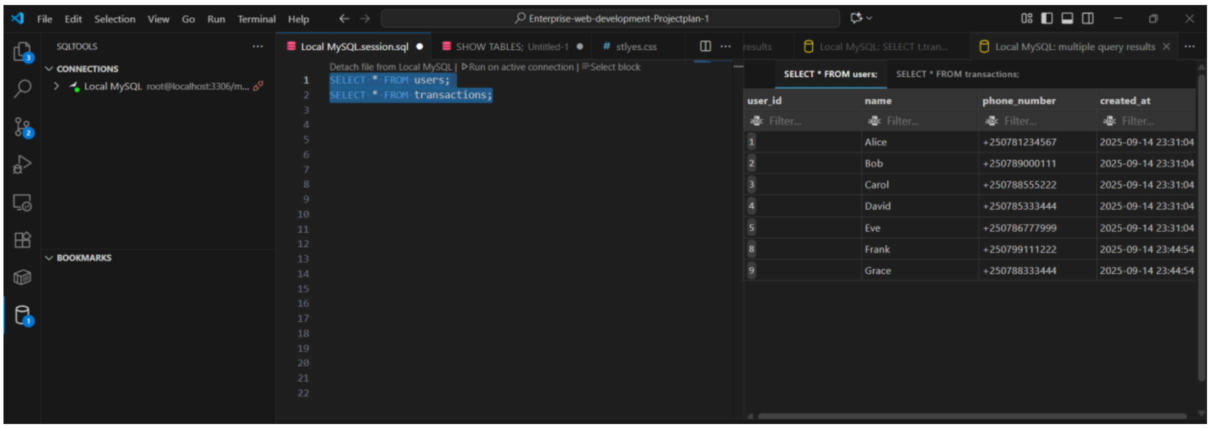
UPDATE a user → "1 row affected" + verify SELECT (screenshot).



```
UPDATE users
SET phone_number = '+250781999999'
WHERE name = 'Alice';
```

```
USE momo; 9/14/2025, 11:56:58 PM
DESCRIBE transactions; 9/14/2025, 11:50:2...
INSERT INTO users (name, phone_numbe...
INSERT INTO users (name, phone_numbe...
SHOW TABLES; 9/14/2025, 11:41:30 PM
USE momo; CREATE TABLE users ( id INT ...
SHOW TABLES; 9/14/2025, 11:39:10 PM
CREATE TABLE IF NOT EXISTS system_log...
```

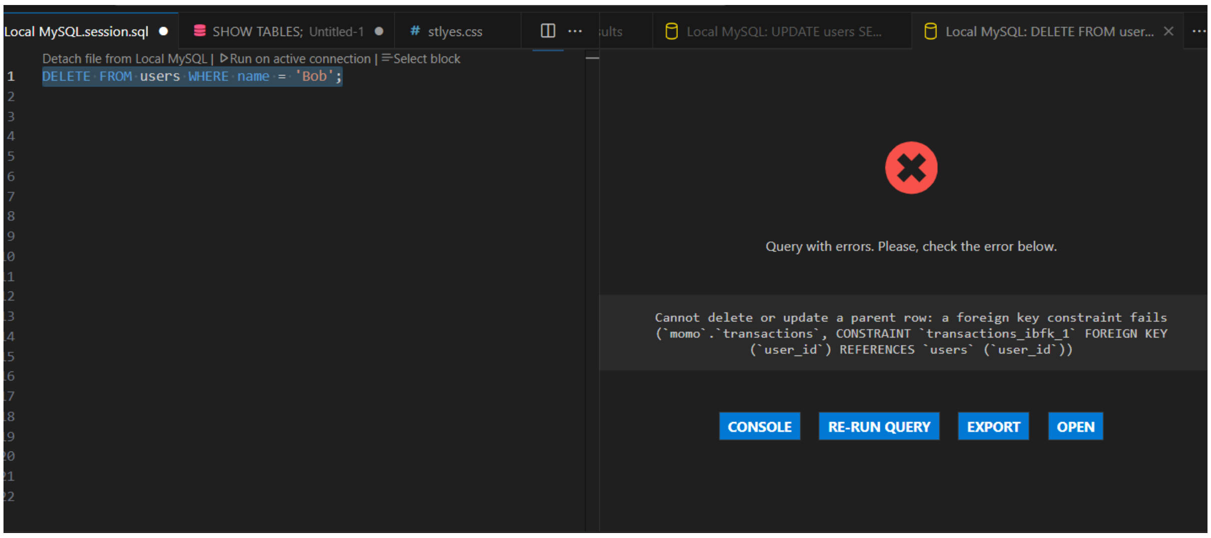
```
1 rows were affected.
1 rows were changed.
```



```
SELECT * FROM users;
SELECT * FROM transactions;
```

user_id	name	phone_number	created_at
1	Alice	+250781234567	2025-09-14 23:31:04
2	Bob	+250789000111	2025-09-14 23:31:04
3	Carol	+250788555222	2025-09-14 23:31:04
4	David	+250785333444	2025-09-14 23:31:04
5	Eve	+250786777999	2025-09-14 23:31:04
8	Frank	+250799111222	2025-09-14 23:44:54
9	Grace	+250788333444	2025-09-14 23:44:54

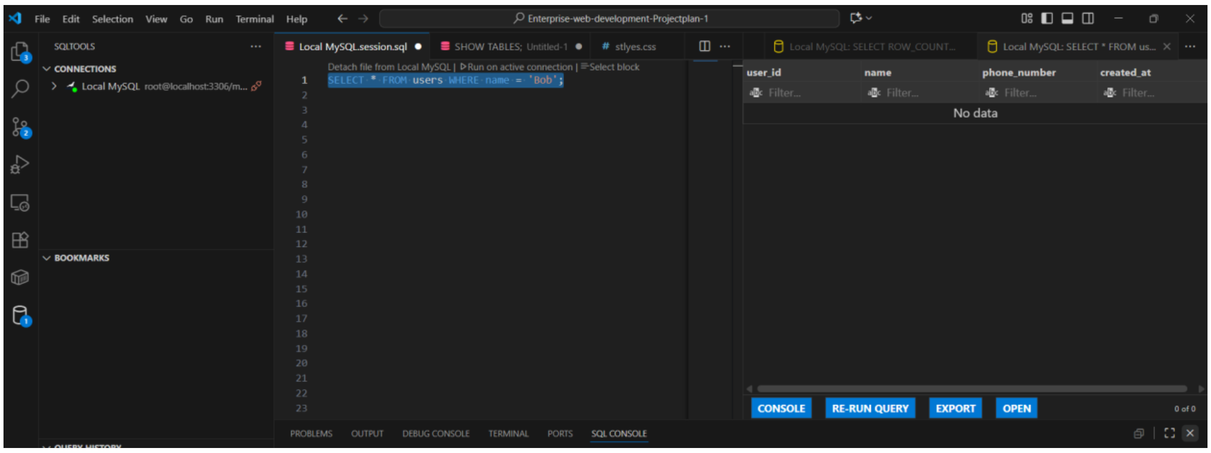
DELETE a user → (screenshot).


Local MySQL.session.sql
SHOW TABLES; Untitled-1
stlyes.css
Local MySQL: UPDATE users SE...
Local MySQL: DELETE FROM user...
Detach file from Local MySQL | Run on active connection | Select block
CONSOLE
RE-RUN QUERY
EXPORT
OPEN

```sql
DELETE FROM users WHERE name = 'Bob';
```

Query with errors. Please, check the error below.

```
Cannot delete or update a parent row: a foreign key constraint fails
(`momo`.`transactions`, CONSTRAINT `transactions_ibfk_1` FOREIGN KEY
(`user_id`) REFERENCES `users` (`user_id`))
```


File Edit Selection View Go Run Terminal Help
Enterprise-web-development-Projectplan-1
SQLTOOLS
CONNECTIONS
Local MySQL root@localhost:3306/m...
BOOKMARKS
Local MySQL.session.sql
SHOW TABLES; Untitled-1
stlyes.css
Local MySQL: SELECT ROW_COUNT...
Local MySQL: SELECT * FROM us...
Detach file from Local MySQL | Run on active connection | Select block
CONSOLE
RE-RUN QUERY
EXPORT
OPEN
0 of 0
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```sql
SELECT * FROM users WHERE name = 'Bob';
```

user_id	name	phone_number	created_at
No data			

3) Mapping: SQL → JSON (put this in README)

Users

users.id → user.id

users.msisdn → user.msisdn

users.name → user.name

users.network → user.network

users.created_at → user.created_at

Transaction_Categories

transaction_categories.id → category.id

transaction_categories.name → category.name

transaction_categories.created_at → category.created_at

Transactions (flat/min)

transactions.id → transaction_min.id

transactions.occurred_at → transaction_min.occurred_at

transactions.amount → transaction_min.amount

transactions.currency → transaction_min.currency

transactions.raw_text → transaction_min.raw_text

transactions.source_file → transaction_min.source_file

transactions.ingested_at → transaction_min.ingested_at

transactions.sender_id → transaction_min.sender_id

transactions.receiver_id → transaction_min.receiver_id

transactions.category_id → transaction_min.category_id

Transactions (nested/full)

transactions.sender_id + join users → transaction_full.sender object

transactions.receiver_id + join users → transaction_full.receiver object (nullable)

transactions.category_id + join transaction_categories → transaction_full.category object

Optional tags (if you have transaction_tags) → transaction_full.tags (array of strings or tag names)

System_Logs

system_logs.* → system_log.* (same field names)

Documentation: Include a brief explanation (200-300 words) justifying your design decisions

We modeled Users, Transaction Categories, Transactions, and System Logs to reflect MoMo SMS processing. users holds unique participants, transaction_categories normalizes business types (CASHIN, CASHOUT, PAY, FEE), transactions is the fact table linking a user and category with amount/timestamp, and system_logs records ETL pipeline events. We enforced referential integrity with foreign keys and added indexes on transactions(user_id, category_id, transaction_date) to speed frequent dashboard queries (by user, by category, by day). A sample many-to-many is demonstrated via transaction_tags to support flexible labeling without denormalizing transactions. JSON design includes both a flat row (transaction_min) and a nested API response (transaction_full) so the frontend can render a transaction and its related entities without extra roundtrips. We included a simple CHECK (amount > 0) to document constraints on monetary values. CRUD tests (insert/read/update/delete) and an FK negative test verify correctness. This schema is intentionally small but extensible (e.g., adding merchants, agents, or splitting sender/receiver later). The ERD and scripts align with Week-2 deliverables and support the next stages: ETL loading and dashboard analytics.