1. Atomicity

To demonstrate that all operations within a transaction either succeed or fail together, we'll update two records in the `sets` table.

i. Begin transaction
BEGIN TRANSACTION;

ii. First update within the transaction
UPDATE sets SET name = 'New Name 1' WHERE set_num = '1234-1';

iii. Second update within the transaction
UPDATE sets SET name = 'New Name 2' WHERE set_num = '5678-1';

iv. Commit the transaction
COMMIT;

v. Execute SELECT command to verify the updated records
SELECT * FROM sets WHERE set_num IN ('1234-1', '5678-1');

2. Consistency

i. Rename the existing `sets` table:
ALTER TABLE sets RENAME TO sets_old;

ii. Create a new `sets` table with a `CHECK` constraint (including `img_url` column) - yellow lines keep consistency:

```
CREATE TABLE sets (
   set_num TEXT PRIMARY KEY,
   name TEXT NOT NULL,
   year INTEGER NOT NULL CHECK (year >= 0),
   theme_id INTEGER,
   num_parts INTEGER CHECK (num_parts >= 0),
   img_url TEXT);
```

iii. Copy data from the old table to the new table:
INSERT INTO sets (set_num, name, year, theme_id, num_parts, img_url)
SELECT set_num, name, year, theme_id, num_parts, img_url FROM sets_old;

iv. Drop the old table:
DROP TABLE sets_old;

v. Insert valid data:
INSERT INTO sets (set_num, name, year, theme_id, num_parts, img_url) VALUES ('1234-5678', 'Set 1', 2021, 1, 100, 'http://example.com/img1.jpg');

vi. Attempt to insert data with negative `num_parts` (an error message should occur = violate consistency):
INSERT INTO sets (set_num, name, year, theme_id, num_parts, img_url) VALUES ('5678-123', 'Set 2', 2022, 1, -50, 'http://example.com/img2.jpg');

3. Isolation (We can show this example but we cannot run this example because this example should be executed in a separate session or connection. Therefore, just show the SQL command and explain why this example is Isolation)

i. Insert initial data
INSERT INTO sets (set_num, name, year, theme_id, num_parts, img_url) VALUES ('7890-11', 'Set 3', 2021, 1, 50, 'http://example.com/img3.jpg');

ii. Start the first transaction(data update)
BEGIN TRANSACTION;

-- Update data in the first transaction
UPDATE sets SET num_parts = 200 WHERE set_num = '7890-11';

-- Do not commit yet, keep the transaction open

iii. Start the second transaction in a new connection(data read)
-- Note: This should be executed in a separate session or connection
BEGIN TRANSACTION;

-- Read data in the second transaction
SELECT * FROM sets WHERE set_num = '7890-11';

-- Do not commit yet, keep the transaction open

iv. Commit the first transaction
COMMIT;

v. Commit the second transaction
COMMIT;

vi. Verify the data after both transactions are committed
SELECT * FROM sets WHERE set_num = '7890-11';

4. Durability (This example also has a similar problem to the isolation example. We need to show that data is not lost even after the system is shut down and restarted, but in webSQL IDE if we close the site, the whole DB disappears. Therefore, just show this example and explain why this example satisfied durability.)

i. Start a transaction and insert data:
– BEGIN TRANSACTION;

INSERT INTO sets (set_num, name, year, theme_id, num_parts, img_url) VALUES ('7890-12', 'Durability Test Set', 2024, 1, 150, 'http://example.com/img12.jpg');

-- Do not commit yet, keep the transaction open.

ii. Commit the transaction:
COMMIT;

iii. Close and reopen the SQLite database:

- This step involves closing and reopening the SQL IDE or SQLite CLI.

iv. Verify that the data is permanently stored:
SELECT * FROM sets WHERE set_num = '7890-12';