Transactions

- ## Non Conflicting Transactions

1. **Serial Transactions - Updating the same data twice**

    START TRANSACTION;
    START TRANSACTION;
    *SELECT * FROM User;*
    UPDATE User
    SET pin_code = 123456
    WHERE id = 1;
    COMMIT;
    *SELECT * FROM User;*
    UPDATE User
    SET pin_code = 123457
    WHERE id = 1;
    COMMIT;

2. **Serial Transactions - Reading and accessing updated data from previous Transaction**

    START TRANSACTION;
    START TRANSACTION;
    *SELECT * FROM User;*
    UPDATE User
    SET password = '123456'
    WHERE id = 1;
    COMMIT;
    *SELECT * FROM User WHERE password LIKE '123456';*
    COMMIT;

3. **Isolation - Working in 2 separate tables**

    START TRANSACTION;
    START TRANSACTION;
    *SELECT * FROM User;*
    UPDATE User

SET password = '103456'
WHERE id = 20;
COMMIT;
*SELECT * FROM Product;*
UPDATE Product
SET name ='lewis vitton'
WHERE price =100;
COMMIT;

4. **Consistency - Transaction reading data ie updated by previous Transaction**

   START TRANSACTION;
   START TRANSACTION;
   DELETE FROM Product
   WHERE price< 20;
   COMMIT;
   *SELECT * FROM Product*
   WHERE price =100;
   COMMIT;

● Conflicting Transactions

1. **Write - Write Conflict**

   START TRANSACTION;
   UPDATE User SET house_num = 'A_999' WHERE id = 1;
   *SELECT * FROM User;*
   START TRANSACTION;
   UPDATE User SET house_num = 'A_888' WHERE id = 1;
   *SELECT * FROM User;*
   COMMIT;
   COMMIT;

2. **Dirty - Read Conflict**

   set SQL_SAFE_UPDATES=0;
   start transaction; -- t1
   start transaction; -- t2

```
update Brand set name ="a" where id=1; -- t2
select * from Brand; -- t1
commit; -- t2
commit; -- t1
```
