Q1. Transactions-Concurrency Consider the table Test (AID, X, Y) following transactions: S: UPDATE TEST SET X:= X + 10 WHERE AID = 1; UPDATE TEST SET Y := Y - 10 WHERE AID = 1; T: UPDATE TEST SET X:= X * 2 WHERE AID = 1; UPDATE TEST SET Y := Y * 2 WHERE AID = 1; U: UPDATE TEST SET Y:= Y + 10 WHERE AID = 1; UPDATE TEST SET X := X - 10 WHERE AID = 1; Assuming initial values of X = 15 and Y = 25, concurrent execution of these three transactions can leave the database in various states. Determine the state of the database (values of X and Y) assuming isolation level serializable (i.e., no dirty reads) for each of S, T, and U. Note: The isolation level is serializable. You do not have to check for appearance of serializability.

Answer:
S,T,U X = 40, Y = 40
S,U,T X = 30, Y = 50
T,S,U X = 30, Y = 50
T,U,S X = 30, Y = 50
U,T,S X = 20, Y = 60
U,S,T X = 30, Y = 50

Q2. Transactions-Representation Consider table Item(name, price) where name is a key, and the following two concurrent transactions. T1: Begin Transaction; S1: Insert Into Item Values ('FCDB',40); S2: Update Item Set price = price + 30 Where name = 'EN'; Commit; T2: Begin Transaction; S3: Select Avg(price) As a1 From Item; S4: Select Max(price) As a2 From Item; Commit; Map these transactions into the internal Read/Write representation.

Answer:
T1:
Begin
Read ('FCDB',40)
Price = Price + 30
Write (Price)
Commit

T2:
Begin
Read item(price)
A.Price = A.Price1+A.Price2+...+A.PriceN
A1 = A.Price/n
Read (A1)
Read item(price)
A2 = max(price)
Read (A2)
Write (A2)
Commit

Note: Because A1 is selected as avg(price) so it just Read it and second max(price) as A2.supose we have one item price is 40 and second item price is 30 so 40+30/number of items=70/2=35 so average is 35.

Q3. Transactions-Weaker isolation levels Consider two tables R(A,B) and S(C). Below are pairs of transactions. Compute what values will be reported by the read statements of weaker isolation levels. Assume individual statements are executed atomically.

(a) Transaction 1:
Set Transaction Isolation Level Read Uncommitted;
Select count(*) From R;
Select count(*) From S;
Commit; Transaction 2:
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Insert Into S Values (3);
Commit;

Answer:

T1	T2
S1 Read(R)	S4 Write(R)
S2 Read(S)	S5 Write(S)
S3 Commit	S6 Commit

S1, S2, S3, S4, S5, S6 = 0, 0

S4, S5, S6, S1, S2, S3 = 1, 1

S1, S4, S5, S6, S2, S3 = 0, 1

S4, S1, S2, S5, S3, S6 = 1, 0

(b) Transaction 1:
Set Transaction Isolation Level Read Committed;
Select count(*) From R;
Select count(*) From S;
Commit;

Transaction 2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (6,2);
Insert Into R Values (3,4);
Insert Into S Values (5);
Insert Into S Values (4);
Commit;
```

Answer:

T1	T2
S1 Read(R)	S4 Write(R)
S2 Read(S)	S5 Write(R)
S3 Commit	S6 Write(S)
	S7 Write(S)
	S8 Commit

S1, S2, S3, S4, S5, S6, S7, S8 = 0, 0

S4, S5, S6, S7, S8, S1, S2, S3 = 2, 2

S1, S4, S5, S6, S7, S8, S2, S3 = 0, 2

(c) Transaction 1:

```
Set Transaction Isolation Level Repeatable Read;
Select count(*) From R;
Select count(*) From S;
Select count(*) From R;
Commit;
```

Transaction 2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Select * From R;
Commit;
```

Answer:

T1	T2
S1 Read(R)	S5 Write(R)
S2 Read(S)	S6 Read(R)
S3 Read(R)	S7 Commit
S4 Commit	

S1, S2, S3, S4, S5, S6, S7 = 0, 0, 0

S5, S6, S7, S1, S2, S3, S4 = 1, 0, 1

S1, S5, S6, S7, S2, S3, S4 = 0, 0, 1