Task 4

tread.sql

Isolation Level:
READ ONLY (READ UNCOMMITTED)

Reasoning:
The only statement in this file is a SELECT statement. Since this statement will only retrieve data and not modify it in any way, this can be run at the lowest level of isolation, read only.

Justification:
N/A - this is the lowest level of isolation

twrite.sql

Isolation Level:
READ COMMITTED

Reasoning:
A read committed level will make sure that queries will execute on the latest copy of committed data in the database. A higher level is not necessary as there is only one modifying query being performed in this transaction.

Justification:
Running at READ ONLY (READ UNCOMMITTED):

twrite2.sql	**twrite.sql**

`UPDATE POSITION`
`SET SALARY = 1.1*SALARY`
`WHERE P# = &1;`	

	`UPDATE POSITION`
	`SET SALARY = SALARY + 10`
	`WHERE P# = &1;`
	
	The above query will now use the old uncommitted data from the 1st update in twrite2.sql which will corrupt the database. Having used the Read Committed isolation level would solve this.

`UPDATE POSITION`
`SET SALARY = (SELECT SALARY`
`FROM POSITION`
`WHERE P# = &1)`
`WHERE P# = &2;`	

twrite2.sql

Isolation Level:
SERIALISABLE

Reasoning:
In this transaction there are multiple UPDATE statements that will both modify the dataset and query existing data. Both the update statements and select statement will need to work on the same instance of the data. A serialisable isolation level will allow a snapshot of the data to be taken and used before being worked on by the transaction.

Justification:
Running at READ COMMITTED:

twrite2.sql	**twrite.sql**

`UPDATE POSITION`
`SET SALARY = 1.1*SALARY`
`WHERE P# = &1;`	

	`UPDATE POSITION`
	`SET SALARY = SALARY + 10`
	`WHERE P# = &1;`

`UPDATE POSITION`
`SET SALARY = (SELECT SALARY`
`FROM POSITION`
`WHERE P# = &1)`
`WHERE P# = &2;`
 	
The above query will select data that has been updated from twrite.sql. This would corrupt the data as the salary will be updated using the incorrect data for this transaction.	