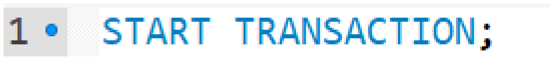
PART 5: DO ALL EXERCISE AT THE END OF EACH CHAPTER

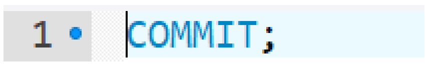
PART 1: Chapter 12 – Managing Transactions

1. You are setting up a transaction that first inserts data in one table and then updates data in a second table. The autocommit mode is set to on, so you must begin the transaction manually. Create an SQL statement that begins the transaction.



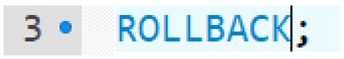
```
START TRANSACTION;
```

2. For the transaction that you created in Step 1, you want any changes committed to the database if the INSERT and UPDATE statements execute successfully. Create an SQL statement that commits the transaction.



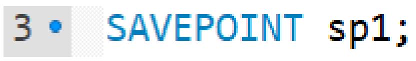
```
COMMIT;
```

3. For the transaction that you created in Step 1, you want all changes to the database rolled back if either the INSERT or UPDATE statement fails. Create an SQL statement that rolls back the transaction.



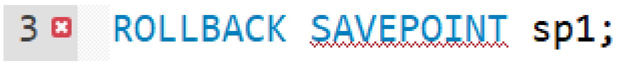
```
ROLLBACK;
```

4. For the transaction that you created in Step 1, you want to add a savepoint to the transaction after the INSERT statement. The name of the savepoint is save1. Create an SQL statement that creates the savepoint.



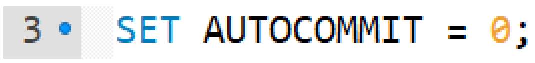
```
SAVEPOINT sp1;
```

5. For the transaction that you created in Step 1, you want to add a statement that rolls back the transaction to the savepoint. Create an SQL statement that rolls back the transaction to the savepoint.



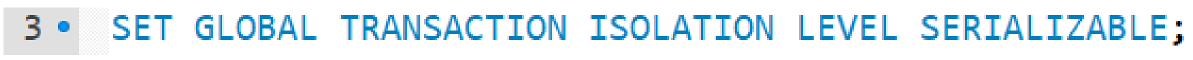
```
ROLLBACK SAVEPOINT sp1;
```

6. You want to turn the autocommit mode for the current session to off. Create an SQL statement that changes the autocommit mode to off.



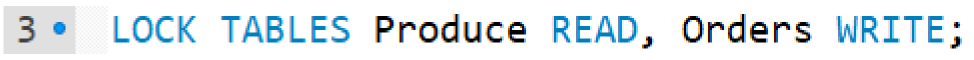
```
SET AUTOCOMMIT = 0;
```

7. You want to set the transaction isolation level to SERIALIZABLE. You want the new level to apply to all sessions. Create an SQL statement that changes the isolation level to SERIALIZABLE.



```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

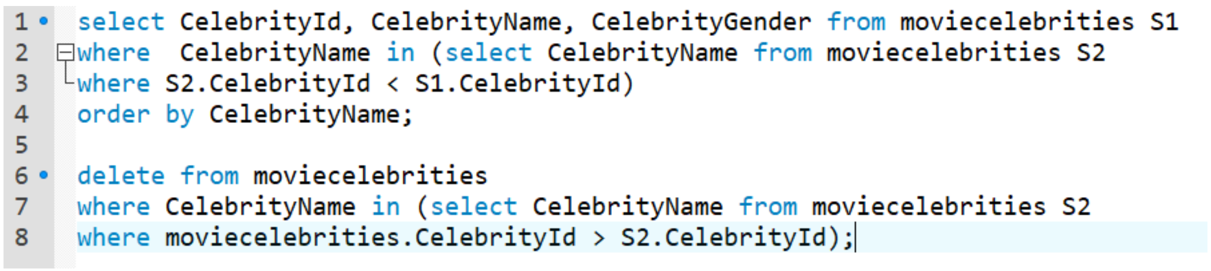
8. You want to lock two MyISAM tables: Produce and Orders. You want to set a READ lock on the Produce table and a WRITE lock on the Orders table. Create an SQL statement that locks the two tables.



```sql
LOCK TABLES Produce READ, Orders WRITE;
```

PART 2: Chapter 9 – Advanced Queries

1. Write the SQL to remove the unneeded duplicate rows, leaving just one row.

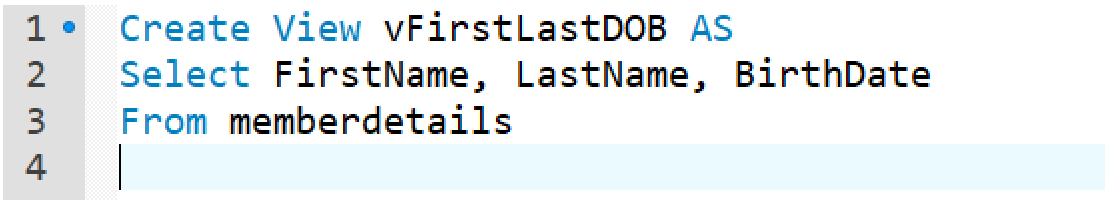


```sql
select CelebrityId, CelebrityName, CelebrityGender from moviecelebrities S1
where  CelebrityName in (select CelebrityName from moviecelebrities S2
where S2.CelebrityId < S1.CelebrityId)
order by CelebrityName;

delete from moviecelebrities
where CelebrityName in (select CelebrityName from moviecelebrities S2
where moviecelebrities.CelebrityId > S2.CelebrityId);
```

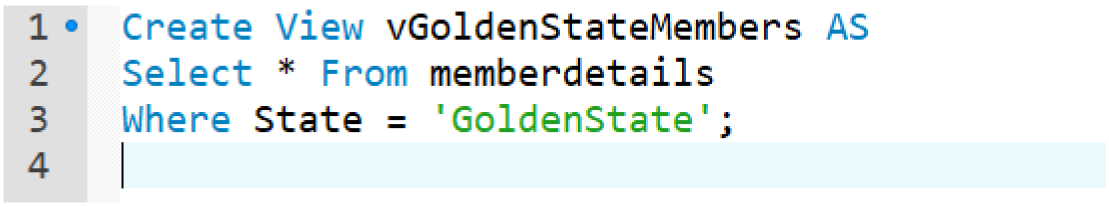
PART 3: Chapter 10 – Views

1. Build a saved field view named vFirstLastDOB to display the first name, last name, and birth date for all members in the MemberDetails table



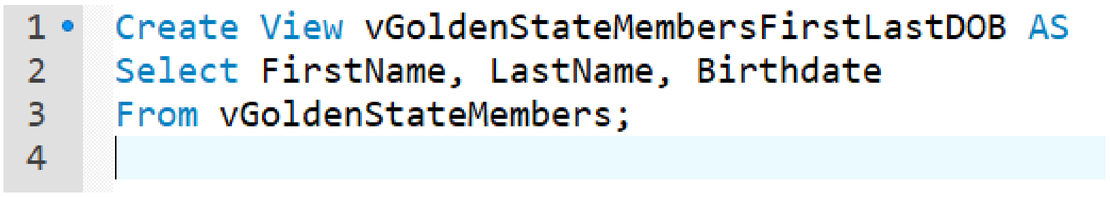
```sql
Create View vFirstLastDOB AS
Select FirstName, LastName, BirthDate
From memberdetails
```

2. Build a saved field view named vGoldenStateMembers displaying all the fields from the MemberDetails table from all members in Golden State



```sql
Create View vGoldenStateMembers AS
Select * From memberdetails
Where State = 'GoldenState';
```

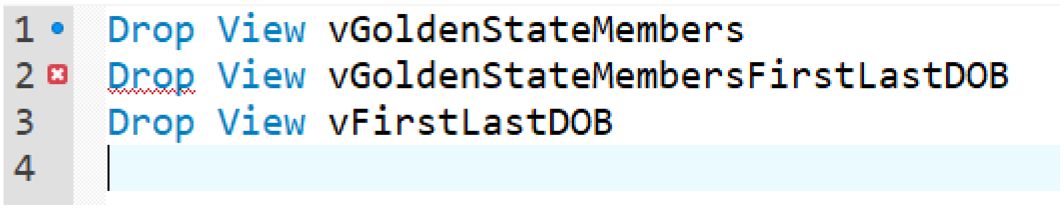
3. Build a windowed view named vGoldenStateMembersFirstLastDOB building in vGoldenStateMembers, pulling only the first name, last name, and birth date fields.



```
Create View vGoldenStateMembersFirstLastDOB AS
Select FirstName, LastName, Birthdate
From vGoldenStateMembers;

```

4. Drop all the views just created.

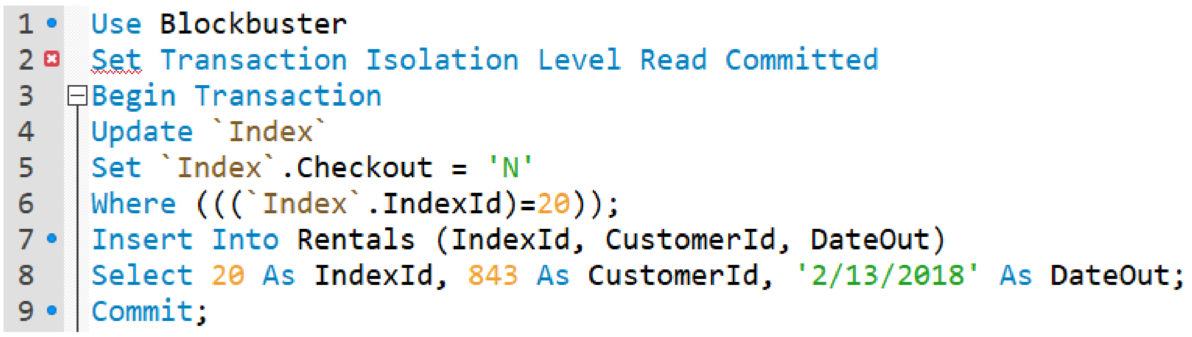


```
Drop View vGoldenStateMembers
Drop View vGoldenStateMembersFirstLastDOB
Drop View vFirstLastDOB

```

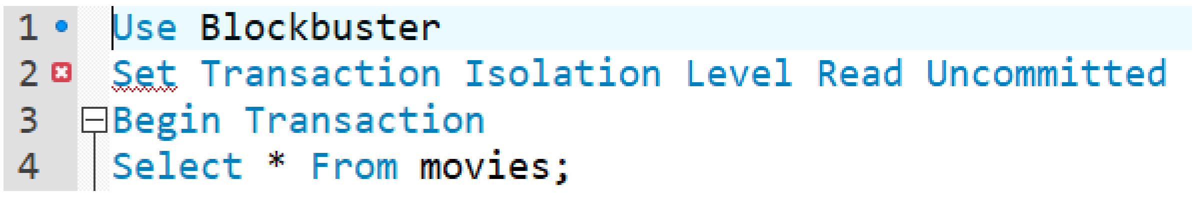
PART 4: Chapter 11 – Transactions

1. Rewrite the update to the Inventory and insert data into the Rentals table using a READ COMMITTED isolation level.



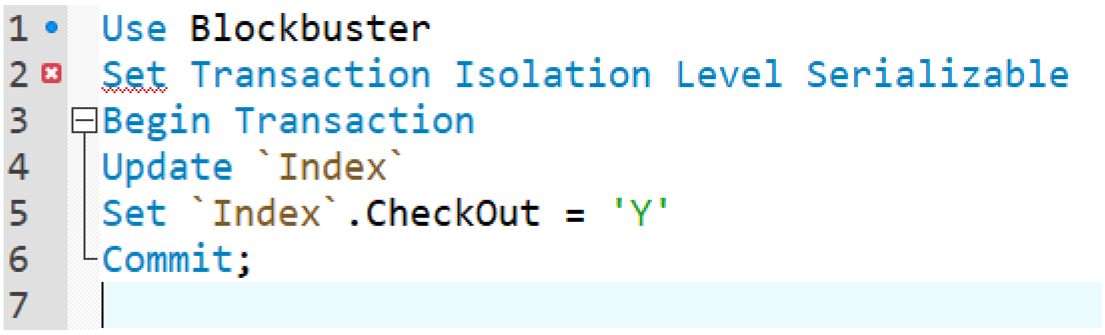
```
Use Blockbuster
Set Transaction Isolation Level Read Committed
Begin Transaction
Update `Index`
Set `Index`.Checkout = 'N'
Where (((`Index`.IndexId)=20));
Insert Into Rentals (IndexId, CustomerId, DateOut)
Select 20 As IndexId, 843 As CustomerId, '2/13/2018' As DateOut;
Commit;
```

2. Write a SELECT statement to view all titles in the Films table using the READ COMMITTED isolation level.



```
Use Blockbuster
Set Transaction Isolation Level Read Uncommitted
Begin Transaction
Select * From movies;
```

3. Write an UPDATE query to update all inventory to CheckedOut (= 'Y') using the SERIALIZABLE isolation level.



```
Use Blockbuster
Set Transaction Isolation Level Serializable
Begin Transaction
Update `Index`
Set `Index`.CheckOut = 'Y'
Commit;
```