**Please only use mySQL.**

**SQL procedures, functions, triggers, and prepared statements in MySQL.**

This assignment gives you an opportunity to create stored procedures, functions, triggers, and prepared statements from queries you created for the Shark schema.  **Complete this assignment given the *sharkDB* schema provided .**

**sharkDB is dump database please import the database and then start writing sql code for each question.**

 Please make one .sql file. The file should contain the SQL code for each question named. Each section starts with a comment that lists the question number as well as the question description and any other comment you believe helps to describe the solution. The solution is followed by test code that runs the solution, make sure you provide different executions of the solution.

1.      Write a function mostAttacks() that returns the township id of the township thay had the most number of attacks. The function accepts no arguments. If there are more than 1 town with the maximum value, return any town with the maximum value.

2. Write a procedure allReceivers(town, state) that accepts a town name and a state abbreviation and returns a result set of all receivers in that town. The result should contain all the fields in the receiver table as well as the provided town and state.

3.      Write a procedure named sharkLenGTE(length\_p)  that accepts a length for a shark and  returns a result set that contains the shark id, shark name, shark length, shark sex , and the number of detections for that shark for all sharks with a length greater than or equal to the passed length.

 4.      Write a function named numSharkWithLen(length\_p)  that accepts a shark length and returns the number of sharks with that length.

5.      Write a procedure  named sightingsByTown( ) that accepts no arguments  and returns a row for each township tuple in the township table. The result contains the number of sightings per town, the town name and the state abbreviation.

 6.   Write a function named moreDetections(shark1,shark2). It accepts 2 shark names and returns 1 if shark1 has had more detections than shark2, 0 if they have had the same number of detections , and -1 if shark2 has had more detections than shark1.

 7.      Create a procedure named createAttack( sname\_p , vname\_p , vage\_p , fatal\_p, attack\_date,  activity\_p,  description\_p ,town\_p, state\_p ) that inserts an attack into the database . Make sure you create the appropriate tuples in the victim, shark and township table as well. Insert another attack into the attack table. victim name = “Ace Ventura”, age = 35, town = “Wellfleet”, state = “MA”,  shark\_name = NULL, fatal = 0, date = ‘2021-08-11’, description = “right foot”, activity = “surfing”.

8. Modify the township table to track the number of shark attacks for that town. Call the new field numAttacks. Write a procedure named initialize\_num\_attack(townid) that initializes the field for a specific  township. Call the procedure for each town in the attack table.

 9. Write a trigger that updates township.numAttacks whenever an attack is added to the attack table. Name the trigger attack\_after\_insert(). Insert an attack into the attack table to verify your trigger is working;  victim name = “Jennifer Jones”, age = 25, town = “Truro”, state = “MA”,  sharkid = NULL, fatal = 0, date = ‘2021-11-11’, description = “left foot”, activity = “surfing”.

 10. Write a trigger that updates township.numAttacks whenever an attack is deleted from the attack table. Call the trigger attack\_after\_delete().  Delete the  attack from the attack table to verify your trigger is working;  victim name = “Jennifer Jones”, age = 25, town = “Truro”, state = “MA”,  sharkid = NULL, fatal = 0, date = ‘2021-11-11’, description = “left foot”, activity = “surfing”.

11.Create and execute a prepared statement from the SQL workbench that calls the function moreDetections(shark1, shark2). Use 2 user session variables to pass the two arguments to the function. Pass the values “Amy” and “Alex” as the shark variables .

12.  Create and execute a prepared statement from the SQL workbench that calls the function numSharkWithLen(length\_p). Use a user session variable to pass the length to the function. Pass the value 14 as the length .