TRIGGERS:

1.Create a trigger to validate the age at the time before inserting a new record into Sailors table and if inv alid age(age<0 or age >100) found then it stores 0 instead.

sol:

```
CREATE OR REPLACE TRIGGER check_age_trigger
BEFORE INSERT ON Sailors
FOR EACH ROW
DECLARE
BEGIN
    -- Check if age is less than 0 or greater than 100
    IF :NEW.age < 0 OR :NEW.age > 100 THEN
        -- Set the age to 0
        :NEW.age := 0;
    END IF;
END;
/
```

2.Create a trigger to store the sid of a student into the table deleted_students before deleting any record fr om students table.

sol:

```
SQL> -- Create the students table
SQL> CREATE TABLE students (
  2      sid NUMBER PRIMARY KEY,
  3      name VARCHAR2(50),
  4      age NUMBER
  5  );

Table created.

SQL>
SQL> -- Create the deleted_students table
SQL> CREATE TABLE deleted_students (
  2      deleted_sid NUMBER
  3  );

Table created.

SQL> -- Create the trigger
SQL> CREATE OR REPLACE TRIGGER before_delete_students
  2  BEFORE DELETE ON students
  3  FOR EACH ROW
  4  BEGIN
  5      -- Insert the sid of the deleted student into the deleted_students table
  6      INSERT INTO deleted_students (deleted_sid) VALUES (:OLD.sid);
  7  END;
  8  /

Trigger created.

SQL> -- Insert sample data into the students table
SQL> INSERT INTO students (sid, name, age) VALUES (1, 'John Doe', 20);
```

1 row created.

SQL> INSERT INTO students (sid, name, age) VALUES (2, 'Jane Smith', 22);

1 row created.

SQL> INSERT INTO students (sid, name, age) VALUES (3, 'Michael Johnson', 25);

1 row created.

SQL> SELECT * FROM students;

```
       SID NAME                                               AGE
---------- -------------------------------------------------- ----------
         1 John Doe                                           20
         2 Jane Smith                                          22
         3 Michael Johnson                                        25
```

SQL> SELECT * FROM deleted_students;

no rows selected

```
SQL> -- Delete a record from the students table
SQL> DELETE FROM students WHERE sid = 2;
```

1 row deleted.

SQL> SELECT * FROM students;

```
       SID NAME                                               AGE
---------- -------------------------------------------------- ----------
         1 John Doe                                           20
         3 Michael Johnson                                        25
```

SQL> SELECT * FROM deleted_students;

```
DELETED_SID
-----------
          2
```

3.Write a trigger to do the following:
if the ticket is booked in advance of more than 60 days, reject it.
I.e, date of journey must not greater than 60 days from reservation date.

sol:

```
CREATE TABLE bookings (
    ticket_id NUMBER PRIMARY KEY,
    reservation_date DATE,
    journey_date DATE
);

-- Create the trigger
CREATE OR REPLACE TRIGGER reject_advanced_booking
BEFORE INSERT ON bookings
```

```
FOR EACH ROW
DECLARE
   max_advance_days CONSTANT NUMBER := 60;
BEGIN
   IF (:NEW.journey_date - SYSDATE) > max_advance_days THEN
      RAISE_APPLICATION_ERROR(-20001, 'Ticket booking is more than 60 days in advance. Booking r
ejected.');
   END IF;
END;
/

-- Insert a booking with a journey date more than 60 days in advance
INSERT INTO bookings (ticket_id, reservation_date, journey_date)
VALUES (4, TO_DATE('2023-08-02', 'YYYY-MM-DD'), TO_DATE('2023-10-03', 'YYYY-MM-DD'));
```

4.CREATE OR REPLACE TRIGGER trig1 before insert on Passenger for each row to avoid duplicate ins
ertion.

sol:

```
CREATE TABLE Passenger (
   passenger_id NUMBER PRIMARY KEY,
   name VARCHAR2(100),
   age NUMBER
);

CREATE OR REPLACE TRIGGER trig1
BEFORE INSERT ON Passenger
FOR EACH ROW
DECLARE
   duplicate_count NUMBER;
BEGIN
   -- Check if the new passenger_id already exists in the table
   SELECT COUNT(*) INTO duplicate_count
   FROM Passenger
   WHERE passenger_id = :NEW.passenger_id;

   IF duplicate_count > 0 THEN
      -- Raise an application error to prevent insertion of duplicates
      RAISE_APPLICATION_ERROR(-20001, 'Duplicate passenger_id. Insertion rejected.');
   END IF;
END;
/


-- Insert valid data
INSERT INTO Passenger (passenger_id, name, age) VALUES (1, 'John Doe', 25);
INSERT INTO Passenger (passenger_id, name, age) VALUES (2, 'Jane Smith', 30);

-- Insert a duplicate passenger_id to trigger the error
INSERT INTO Passenger (passenger_id, name, age) VALUES (1, 'Michael Johnson', 28);
```