```
/*LAP 3*/

/*1*/
START TRANSACTION;

INSERT INTO Students (FirstName, LastName, Gender, BirthDate) VALUES ('Michael', 'Johnson', 'male',
'1993-09-25');
SET @newStudentID = LAST_INSERT_ID();

INSERT INTO Exams (StudentID, SubjectID, ExamDate, Score) VALUES
(@newStudentID, 1, '2024-06-10', 85),
(@newStudentID, 2, '2024-06-12', 90),
(@newStudentID, 3, '2024-06-14', 80),
(@newStudentID, 4, '2024-06-16', 75);

COMMIT;

/*2*/
SELECT CONCAT(DAY(ExamDate), ' ', MONTHNAME(ExamDate), ' ', YEAR(ExamDate)) AS
ExamDateFormatted FROM Exams;

/*3*/
SELECT FirstName, LastName, YEAR(CURDATE()) - YEAR(BirthDate) - (DATE_FORMAT(CURDATE(), '%m%d')
< DATE_FORMAT(BirthDate, '%m%d')) AS Age FROM Students;
```

```
/*4*/

SELECT s.FirstName, s.LastName, ROUND(e.Score) AS RoundedScore FROM Students s

JOIN Exams e ON s.StudentID = e.StudentID;


/*5*/

SELECT CONCAT(FirstName, ' ', LastName) AS StudentName, YEAR(BirthDate) AS BirthYear FROM
Students;


/*6*/

INSERT INTO Exams (StudentID, SubjectID, ExamDate, Score) VALUES (1, 1, NOW(), 95);


/*7*/



DELIMITER //


CREATE FUNCTION HelloWorld(username VARCHAR(100))

RETURNS VARCHAR(255)

BEGIN

   RETURN CONCAT('Welcome, ', username, '!');

END//


DELIMITER ;
```

```
/*8*/

DELIMITER //


CREATE FUNCTION Multiply(a INT, b INT)

RETURNS INT

BEGIN

   RETURN a * b;

END//


DELIMITER ;


/*9*/

DELIMITER //


CREATE FUNCTION GetScore(studentID INT, examID INT)

RETURNS INT

BEGIN

   DECLARE examScore INT;

   SELECT Score INTO examScore FROM Exams WHERE StudentID = studentID AND ExamID = examID;

   RETURN examScore;

END//


DELIMITER ;
```

```
/*10*/

DELIMITER //


CREATE FUNCTION CountFailedStudents(examID INT)

RETURNS INT

BEGIN

   DECLARE failedCount INT;

   SELECT COUNT(*) INTO failedCount FROM Exams WHERE ExamID = examID AND Score < 50;

   RETURN failedCount;

END//


DELIMITER ;


/*11*/

DELIMITER //


CREATE FUNCTION AvgMaxScore(subjectName VARCHAR(100))

RETURNS DECIMAL(5, 2)

BEGIN

   DECLARE avgMax DECIMAL(5, 2);

   SELECT AVG(MaxScore) INTO avgMax FROM Subjects WHERE Name = subjectName;

   RETURN avgMax;

END//
```

```
DELIMITER ;


/*12*/

CREATE TABLE Deleted_Students LIKE Students;


/*13*/

DELIMITER //


CREATE TRIGGER After_Delete_Student

AFTER DELETE ON Students

FOR EACH ROW

BEGIN

   INSERT INTO Deleted_Students SELECT * FROM Students WHERE StudentID = OLD.StudentID;

END//


DELIMITER ;


/*14*/

DELIMITER //


CREATE TRIGGER After_Insert_Student

AFTER INSERT ON Students

FOR EACH ROW

BEGIN

   INSERT INTO Backup_Students SELECT * FROM Students WHERE StudentID = NEW.StudentID;
```

```
END//


DELIMITER ;


/*15*/

-- Assuming you have a contact info table called Contact_Info with columns: ActionTime,
ActionDescription


DELIMITER //


CREATE TRIGGER Contact_Info_Change

AFTER INSERT ON Contact_Info

FOR EACH ROW

BEGIN

   INSERT INTO Contact_Info_Change_Log (ActionTime, ActionDescription) VALUES (NOW(), 'New row
added to Contact_Info table');

END//


DELIMITER ;

/*16*/

CREATE TABLE Contact_Info_Change_Logs (

   LogID INT AUTO_INCREMENT PRIMARY KEY,

   ActionTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   ActionDescription VARCHAR(255)

);
```

```
CREATE TRIGGER Contact_Info_Change_Trigger

AFTER INSERT OR UPDATE ON Contact_Info

FOR EACH ROW

BEGIN

  DECLARE actionDesc VARCHAR(255);


  IF NEW IS NOT NULL AND OLD IS NULL THEN

    SET actionDesc = 'New row added to Contact_Info table';

  ELSEIF NEW IS NOT NULL AND OLD IS NOT NULL THEN

    SET actionDesc = 'Row updated in Contact_Info table';

  END IF;


  INSERT INTO Contact_Info_Change_Logs (ActionDescription) VALUES (actionDesc);

END;
```