PL/SQL

Database Project

Hogwarts School of Witchcraft and Wizardry

MIS409

Hogwarts School Database

By: Amer Rahman

April 5th, 2024

Table of Contents

SQL Code for Data Tables

```
--Drop Table

Drop Table Spells;

Drop Table Classroom;

Drop Table Enrollment;

Drop Table Clubs;

Drop Table HousePoints;

Drop Table Wands;

Drop Table Course;

Drop Table Professor;

Drop Table Team;

Drop Table Student;

Drop Table House;

Drop Table Books;

--Create Table & Insert Commands


CREATE TABLE Books (ISBN VARCHAR(13),Title VARCHAR(255),Author
VARCHAR(255),Publisher VARCHAR(255),PublishYear INT,Genre
VARCHAR(50),Pages INT,PRIMARY KEY(ISBN));

CREATE TABLE House (HouseID INT,HouseHead
VARCHAR(50),CommonRoomLocation VARCHAR(100),HouseGhost
VARCHAR(50),Founder VARCHAR(50),HouseName VARCHAR(50),PRIMARY
KEY(HouseID));
```

```
CREATE TABLE Student (StudentID INT NOT NULL,FirstName
VARCHAR(50),LastName VARCHAR(50),Gender VARCHAR(10),Age INT,HouseID
INT,BloodStatus VARCHAR(50),PRIMARY KEY(StudentID),FOREIGN KEY (HouseID)
REFERENCES House(HouseID));

CREATE TABLE Team (TeamID INT,TeamName VARCHAR(50),StudentID
INT,Position VARCHAR(50),TeamMascot VARCHAR(50),Wins INT,PRIMARY
KEY(TeamID,StudentID),FOREIGN KEY (StudentID) REFERENCES
Student(StudentID));

CREATE TABLE Professor (ProfessorID INT,LastName VARCHAR(50),FirstName
VARCHAR(50),OfficeLocation VARCHAR(100),SubjectTaught VARCHAR(50),Salary
INT,HouseID INT, PRIMARY KEY(ProfessorID),FOREIGN KEY(HOUSEID)REFERENCES
House(HouseID));

CREATE TABLE Course (CourseID VARCHAR(10),CourseName
VARCHAR(50),ProfessorID INT,Description VARCHAR(100),ISBN
VARCHAR(13),Credits INT,PRIMARY KEY(CourseID),FOREIGN KEY (ProfessorID)
REFERENCES Professor(ProfessorID),FOREIGN KEY(ISBN) REFERENCES
Books(ISBN));

CREATE TABLE Wands (WandID INT PRIMARY KEY,WoodType
VARCHAR(50),CoreType VARCHAR(50),Length DECIMAL(4, 2),Flexibility
VARCHAR(20),StudentID INT,YearMade INT,FOREIGN KEY (StudentID) REFERENCES
Student(StudentID));

CREATE TABLE HousePoints (HouseName VARCHAR(50),StudentID
INT,PointsEarned INT,HouseID INT,Reason VARCHAR(200),PRIMARY KEY
(StudentID, HouseID),FOREIGN KEY (StudentID) REFERENCES
Student(StudentID),FOREIGN KEY (HouseID) REFERENCES House(HouseID));

CREATE TABLE Clubs (ClubID VARCHAR(10) PRIMARY KEY,ClubName
VARCHAR(50),ClubLeader VARCHAR(50),Participants INT,Advisor
VARCHAR(50),ProfessorID INT,FOREIGN KEY (ProfessorID) REFERENCES
Professor(ProfessorID));

CREATE TABLE Enrollment (EnrollmentID INT,StudentID INT,CourseID
VARCHAR(10),Grade VARCHAR(20),EnrollmentPeriod VARCHAR(20),PRIMARY
```

```
KEY(EnrollmentID),FOREIGN KEY(StudentID) REFERENCES  Student(StudentID),
FOREIGN KEY(CourseID) REFERENCES Course(CourseID));

CREATE TABLE Classroom (RoomID INT PRIMARY KEY,RoomName
VARCHAR(255),Capacity INT,HouseID INT,ProfessorID INT,FOREIGN KEY (HouseID)
REFERENCES House(HouseID),FOREIGN KEY (ProfessorID) REFERENCES
Professor(ProfessorID));

CREATE TABLE Spells (SpellID INT PRIMARY KEY,SpellName
VARCHAR(50),Description VARCHAR(200),Type VARCHAR(50),ProfessorID
INT,FOREIGN KEY (ProfessorID) REFERENCES Professor(ProfessorID));



Insert into Books values('9780545582989','Advanced Potion-Making','Severus
Snape','Hogwarts Scholastic Inc.',1997,'Non-Fiction',320);

Insert into Books values('9780439139601','Fantastic Beasts & Where To Find
Them','Newt Scamander','Magical Publications',1998,'Non-Fiction',352);

Insert into Books values('9780439554930','Rune Dictionary','Edwardus
Lima','Magical Publications',1999,'Non-Fiction',448);

Insert into Books values('9780439785969','One Thousand Magical Herbs &
Fungi','Phyllida Spore','Green Goblin Books LLC',2000,'Non-Fiction',752);

Insert into Books values('9780439358071','A History Of Magic','Bathilda
Bagshot','Little Red Books',2003,'History',870);

Insert into Books values('9780439023528','Beginners Guide to
Transfiguration','Emeric Switch','Hogwarts Scholastic Inc.',2005,'Non-Fiction',652);

Insert into Books values('9780545010221','Dark Forces: A Guide to Self-
Protection','Quentin Trimble','Little Red Books',2007,'Narrative',784);

Insert into Books values('9780765348270','Quidditch Champions Guide',' Wilbert
Slinkhard','Green Goblin Books LLC',2003,'Article',544);

Insert into Books values('9780812513714','Extreme Incantations','Violeta
Stitch','Magical Publications',1990,'Non-Fiction',814);
```

```
Insert into Books values('9780140449259','The Dream Oracle','Inigo Imago','Little Red Books',1844,'Spirituality',704);

commit;


Insert into House values(100,'Professor McGonagall','GryffindorCommon Room','Nearly Headless Nick','Godric Gryffindor','Gryffindor');

Insert into House values(200,'Professor Sprout','HufflepuffCommon Room','Fat Friar','Helga Hufflepuff','Hufflepuff');

Insert into House values(300,'Professor Flitwick','RavenclawCommon Room','The Grey Lady','Rowena Ravenclaw','Ravenclaw');

Insert into House values(400,'Professor Snape','SlytherinCommon Room','The Bloody Baron','Salazar Slytherin','Slytherin');

Insert into House values(500,'Professor Dumbledore','HeadMasters Tower','Professor Binns','4 Founders','Hogwarts Staff');

commit;


Insert into Student values(111111,'Harry','Potter','Male',17,100,'Half-Blood');

Insert into Student values(222222,'Hermione','Granger','Female',19,100,'Muggle-Born');

Insert into Student values(333333,'Ron','Weasley','Male',18,100,'Pure-Blood');

Insert into Student values(444444,'Oliver','Wood','Male',22,100,'Pure-Blood');

Insert into Student values(555555,'Neville','Longbottom','Male',18,100,'Pure-Blood');

Insert into Student values(666666,'Fred','Weasley','Male',20,100,'Pure-Blood');

Insert into Student values(777777,'George','Weasley','Male',20,100,'Pure-Blood');

Insert into Student values(888888,'Cedric','Diggory','Male',9,200,'Half-Blood');
```

```
Insert into Student values(999999,'Constance','Pickering','Female',17,200,'Half-Blood');

Insert into Student values(900000,'Luna','Lovegood','Female',15,300,'Pure-Blood');

Insert into Student values(800000,'Cho','Chang','Female',17,300,'Half-Blood');

Insert into Student values(700000,'Helena','Ravenclaw','Female',1016,300,'Pure-Blood');

Insert into Student values(600000,'Draco','Malfoy','Male',18,400,'Pure-Blood');

Insert into Student values(500000,'Vincent','Crabbe','Male',19,400,'Pure-Blood');

Insert into Student values(400000,'Tom','Riddle','Male',71,400,'Pure-Blood');

Insert into Student values(300000,'Marcus','Flint','Male',23,400,'Half-Blood');

commit;


Insert into Team values(100,'Gryffindor',111111,'Seeker','Lion',20);

Insert into Team values(100,'Gryffindor',333333,'Beater','Lion',20);

Insert into Team values(100,'Gryffindor',444444,'Chaser','Lion',20);

Insert into Team values(100,'Gryffindor',666666,'Chaser','Lion',20);

Insert into Team values(100,'Gryffindor',777777,'Chaser','Lion',20);

Insert into Team values(200,'Hufflepuff',888888,'Chaser','Badger',11);

Insert into Team values(200,'Hufflepuff',999999,'Keeper','Badger',11);

Insert into Team values(300,'Ravenclaw',900000,'Chaser','Eagle',17);

Insert into Team values(300,'Ravenclaw',700000,'Seeker','Eagle',17);

Insert into Team values(300,'Ravenclaw',800000,'Chaser','Eagle',17);

Insert into Team values(400,'Slytherin',400000,'Beater','Serpent',23);

Insert into Team values(400,'Slytherin',600000,'Seeker','Serpent',23);

Insert into Team values(400,'Slytherin',500000,'Beater','Serpent',23);
```

```
Insert into Team values(400,'Slytherin',300000,'Chaser','Serpent',23);

commit;


Insert into Professor values(246,'McGonagall','Minerva','Office 1, Tower
A','Transfiguration',95000,100);

Insert into Professor values(420,'Sprout','Pomona','Greenhouse 3,
East','Herbology',78000,200);

Insert into Professor values(324,'Flitwick','Filius','Office 2, Tower
B','Charms',82000,300);

Insert into Professor values(974,'Snape','Severus','Dungeon,
Corridor','Potions',180000,400);

Insert into Professor values(804,'Moody','Alastor','Office 3, Tower C','Defense
Against the Dark Arts',93000,500);

Insert into Professor values(113,'Trelawney','Sybill','Office 4, Tower
D','Divination',87000,300);

Insert into Professor values(625,'Vector','Septima','Office 5, Tower
E','Arithmancy',88000,100);

Insert into Professor values(882,'Hooch','Rolanda','Quidditch Pitch','Quidditch
Training',75000,200);

Insert into Professor values(654,'Hagrid','Rubeus','Hut near Forest','Care of
Magical Creatures',88500,100);

Insert into Professor values(591,'Binns','Cuthbert','Office 6, Tower F','History of
Magic',80000,200);

Insert into Professor values(773,'Sinistra','Aurora','Observatory
Tower','Astronomy',79000,500);

Insert into Professor values(999,'Dumbledore ','Albus ','Headmasters Office ,
Tower Z','Headmaster',200000,500);

commit;
```

```
Insert into course values('Pot104','Potions',974,'Study of magical
potions','9780545582989',4);

Insert into course values('Tran115','Transfiguration',246,'Transforming
objects','9780439023528',3);

Insert into course values('DEF101','Defense Against the Dark Arts',804,'Protection
against Dark Arts','9780545010221',4);

Insert into course values('H808','Herbology',420,'Study of magical
plants','9780439785969',3);

Insert into course values('AST324','Astronomy',773,'Study of celestial
objects','9780140449259',2);

Insert into course values('CHA258','Charms',324,'Learning magical
spells','9780812513714',3);

Insert into course values('HIH204','History of Magic',591,'Study of wizarding
history','9780439358071',2);

Insert into course values('QT3002','Quidditch Training',882,'Training for
Quidditch','9780765348270',2);

Insert into course values('MCI497','Magical Creatures',625,'Study of magical
Creatures','9780439139601',3);

Insert into course values('DIV100','Divination',113,'Foretelling the
future','9780439554930',2);

commit;


Insert into Wands values(116,'Oak','Phoenix Feather',10.25,'Soft',111111,1800);

Insert into Wands values(215,'Maple','Dragon
Heartstring',11.5,'Flexible',222222,1805);

Insert into Wands values(314,'Redwood','Unicorn Hair',9.75,'Rigid',333333,1903);

Insert into Wands values(413,'Ebony','Thestral Tail Hair',12,'Soft',444444,1850);
```

```
Insert into Wands values(512,'Willow','Veela Hair',10.75,'Flexible',555555,1899);

Insert into Wands values(612,'Mahogany','Phoenix
Feather',10,'Rigid',666666,1930);

Insert into Wands values(711,'Rosewood','Hippogriff
Feather',11.25,'Flexible',777777,1823);

Insert into Wands values(810,'Ash','Thunderbird Tail
Feather',10.5,'Soft',888888,1870);

Insert into Wands values(991,'Cherry','Basilisk Horn',9.5,'Rigid',999999,1950);

Insert into Wands values(102,'Vine','Kelpie Hair',10.25,'Flexible',900000,1980);

Insert into Wands values(113,'Yew','Dragon Heartstring',11,'Rigid',800000,1805);

Insert into Wands values(124,'Holly','Phoenix Feather',10.75,'Soft',700000,1867);

Insert into Wands values(135,'Blackthorn','Unicorn
Hair',9.75,'Flexible',600000,1877);

Insert into Wands values(146,'Pear','Thestral Tail Hair',12.25,'Rigid',500000,1917);

Insert into Wands values(157,'Walnut','Veela Hair',10.5,'Soft',400000,1800);

Insert into Wands values(168,'Hawthorn','Phoenix
Feather',10.25,'Flexible',300000,1880);

commit;


Insert into HousePoints values('Gryffindor',111111,150,100,'Brave act in the
Forbidden Forest');

Insert into HousePoints values('Gryffindor',222222,120,100,'Saving a fellow
student');

Insert into HousePoints values('Gryffindor',555555,135,100,'Exceptional
performance in Defense Against the Dark Arts');

Insert into HousePoints values('Gryffindor',333333,110,100,'Outstanding
Quidditch performance');
```

```
Insert into HousePoints values('Hufflepuff',888888,110,200,'Charitable work');

Insert into HousePoints values('Hufflepuff',999999,100,200,'Helping a student in need');

Insert into HousePoints values('Ravenclaw',900000,145,300,'Outstanding research project');

Insert into HousePoints values('Ravenclaw',700000,155,300,'Top marks in the O.W.L exams');

Insert into HousePoints values('Ravenclaw',800000,160,300,'Solving a complex riddle');

Insert into HousePoints values('Slytherin',600000,140,400,'Winning the Quidditch Cup');

Insert into HousePoints values('Slytherin',500000,125,400,'Winning the House Cup');

Insert into HousePoints values('Slytherin',300000,115,400,'Duelling competition victory');

Insert into HousePoints values('Slytherin',400000,210,400,'Largest Donation to Hogwarts');

commit;


Insert into clubs values('DC12','Dueling Club','Harry Potter',12,'Professor Snape',974);

Insert into clubs values('FC02','Frog Choir','Cho Chang',15,'Professor Flitwick',324);

Insert into clubs values('CC88','Chess Club','Ron Weasley',10,'Professor McGonagall',246);

Insert into clubs values('AC68','Astronomy Club','Luna Lovegood',18,'Professor Sinistra',773);
```

```
Insert into clubs values('MCM71','Magical Creature Club','Neville
Longbottom',8,'Professor Hagrid',654);

commit;


Insert into Enrollment values(1,111111,'Pot104','Outstanding','Spring');

Insert into Enrollment values(2,111111,'Tran115','Exceeds Expectations','Spring');

Insert into Enrollment values(3,111111,'DEF101','Outstanding','Spring');

Insert into Enrollment values(4,222222,'Pot104','Outstanding','Spring');

Insert into Enrollment values(5,222222,'DEF101','Acceptable','Spring');

Insert into Enrollment values(6,222222,'H808','Exceeds Expectations','Spring');

Insert into Enrollment values(7,333333,'CHA258','Acceptable','Spring');

Insert into Enrollment values(8,333333,'AST324','Dreadful','Spring');

Insert into Enrollment values(9,333333,'QT3002','Outstanding','Spring');

Insert into Enrollment values(10,444444,'HIH204','Troll','Spring');

Insert into Enrollment values(11,444444,'MCI497','Exceeds Expectations','Spring');

Insert into Enrollment values(12,444444,'DIV100','Outstanding','Spring');

Insert into Enrollment values(13,555555,'Pot104','Outstanding','Summer');

Insert into Enrollment values(14,555555,'DEF101','Exceeds
Expectations','Summer');

Insert into Enrollment values(15,555555,'H808','Acceptable','Summer');

Insert into Enrollment values(16,666666,'QT3002','Outstanding','Summer');

Insert into Enrollment values(17,666666,'AST324','Outstanding','Summer');

Insert into Enrollment values(18,666666,'CHA258','Dreadful','Summer');

Insert into Enrollment values(19,777777,'Pot104','Exceeds
Expectations','Summer');
```

```
Insert into Enrollment values(20,777777,'DEF101','Outstanding','Summer');
Insert into Enrollment values(21,777777,'H808','Troll','Summer');
Insert into Enrollment values(22,888888,'HIH204','Exceeds Expectations','Fall');
Insert into Enrollment values(23,888888,'MCI497','Acceptable','Fall');
Insert into Enrollment values(24,888888,'DIV100','Outstanding','Fall');
Insert into Enrollment values(25,999999,'Pot104','Troll','Fall');
Insert into Enrollment values(26,999999,'Tran115','Exceeds Expectations','Fall');
Insert into Enrollment values(27,999999,'DEF101','Exceeds Expectations','Fall');
Insert into Enrollment values(28,900000,'HIH204','Outstanding','Fall');
Insert into Enrollment values(29,900000,'MCI497','Exceeds Expectations','Fall');
Insert into Enrollment values(30,900000,'DIV100','Dreadful','Fall');
Insert into Enrollment values(31,800000,'HIH204','Outstanding','Fall');
Insert into Enrollment values(32,800000,'MCI497','Exceeds Expectations','Fall');
Insert into Enrollment values(33,800000,'DIV100','Troll','Fall');
Insert into Enrollment values(34,700000,'AST324','Outstanding','Spring');
Insert into Enrollment values(35,700000,'CHA258','Exceeds Expectations','Spring');
Insert into Enrollment values(36,700000,'HIH204','Exceeds Expectations','Spring');
Insert into Enrollment values(37,600000,'Pot104','Acceptable','Spring');
Insert into Enrollment values(38,600000,'DEF101','Outstanding','Spring');
Insert into Enrollment values(39,600000,'H808','Outstanding','Spring');
Insert into Enrollment values(40,500000,'Pot104','Exceeds Expectations','Spring');
Insert into Enrollment values(41,500000,'DEF101','Dreadful','Spring');
Insert into Enrollment values(42,500000,'H808','Outstanding','Spring');
Insert into Enrollment values(43,400000,'Tran115','Exceeds Expectations','Fall');
```

```
Insert into Enrollment values(44,400000,'AST324','Exceeds Expectations','Fall');
Insert into Enrollment values(45,400000,'CHA258','Acceptable','Fall');
Insert into Enrollment values(46,300000,'Tran115','Dreadful','Fall');
Insert into Enrollment values(47,300000,'AST324','Outstanding','Fall');
Insert into Enrollment values(48,300000,'CHA258','Troll','Fall');
commit;

Insert into Classroom values(101,'Common Rooms',100,100,246);
Insert into Classroom values(102,'The Great Hall',500,500,804);
Insert into Classroom values(103,'Dumbledores Office',10,500,246);
Insert into Classroom values(104,'Defense Against The Dark Arts
Classroom',30,300,804);
Insert into Classroom values(105,'Room Of Requirement',50,300,324);
Insert into Classroom values(106,'Potions Dungeon',20,400,974);
Insert into Classroom values(107,'The Kitchens',100,500,654);
Insert into Classroom values(108,'The Library',200,500,591);
Insert into Classroom values(109,'The Chamber of Secrets ',5,200,999);
Insert into Classroom values(110,'The Hospital Wing',15,500,654);
Insert into Classroom values(201,'Divination Classroom',20,300,113);
Insert into Classroom values(202,'Herbology Classroom',25,200,420);
Insert into Classroom values(203,'Charms Classroom',35,100,324);
Insert into Classroom values(204,'Astronomy Tower',30,300,773);
Insert into Classroom values(205,'Transfiguration Classroom',40,400,246);
commit;
```

Insert into Spells values(1,'Lumos','Illuminates the tip of the casters wand','Charm',246);

Insert into Spells values(2,'Expelliarmus','Disarms your opponent','Combat',974);

Insert into Spells values(3,'Accio','Summons an object to the caster','Charm',113);

Insert into Spells values(4,'Alohomora','Unlocks doors and windows','Charm',324);

Insert into Spells values(5,'Protego','Shields the caster from spells and physical attacks','Defensive',974);

Insert into Spells values(6,'Expecto Patronum','Conjures a protective Patronus against Dementors','Defensive',804);

Insert into Spells values(7,'Stupefy','Stuns and temporarily incapacitates the target','Attack',974);

Insert into Spells values(8,'Petrificus Totalus','Renders the target completely immobile','Attack',246);

Insert into Spells values(9,'Incendio','Produces fire at the tip of the wand','Charm',324);

Insert into Spells values(10,'Aguamenti','Produces a jet of water from the wand','Charm',324);

Insert into Spells values(11,'Expulso','Causes the target to explode','Attack',804);

Insert into Spells values(12,'Sectumsempra','Causes deep slashing wounds','Attack',804);

Insert into Spells values(13,'Cruciatus Curse','Inflicts intense, excruciating pain ','Attack',804);

Insert into Spells values(14,'Patronus','Conjures a Patronus a guardian of positive energy','Defensive',246);

==========

Question #1: 78 Lines, iterates through Professor departments showing what an increase of 15% in all the professors salaries would look like. It does not make the change; it only shows what the salaries would look like so the Director of the school can see the changes before approving the change.

```
DECLARE
   v_total_salary_increase NUMBER := 0;
   v_old_salary NUMBER;
   v_new_salary NUMBER;
   v_individual_increase NUMBER;

   -- Declare record variable to hold professor data
   professor_rec Professor%ROWTYPE;

   -- Declare cursor for selecting professors from HouseID 100 through 500
   CURSOR c_professors IS
      SELECT * FROM Professor WHERE HouseID IN (100, 200, 300, 400, 500);

   -- Type for professor ID collection
   TYPE professor_id_list IS TABLE OF Professor.ProfessorID%TYPE INDEX BY
PLS_INTEGER;

   l_professor_ids professor_id_list;

BEGIN
   -- Fetch all ProfessorIDs into the collection
   FOR professor_rec IN (SELECT ProfessorID FROM Professor) LOOP
```

```
        -- If the ProfessorID already exists in the collection, raise an exception

        IF l_professor_ids.EXISTS(professor_rec.ProfessorID) THEN

            RAISE_APPLICATION_ERROR(-20001, 'Duplicate ProfessorID found: ' ||
professor_rec.ProfessorID);
        ELSE
            -- Otherwise, add the ProfessorID to the collection
            l_professor_ids(professor_rec.ProfessorID) := 1;
        END IF;
    END LOOP;
DBMS_OUTPUT.PUT_LINE('Professor salary increased by 15% *****'); -- Display
original and new salary

    OPEN c_professors;  -- Open cursor

    LOOP  -- Loop through each professor
        FETCH c_professors INTO professor_rec; -- Fetch professor record
        EXIT WHEN c_professors%NOTFOUND; -- Exit loop if no more records

        v_old_salary := professor_rec.Salary; -- Calculate the new salary with a 15%
increase

        v_new_salary := v_old_salary * 1.15;  -- 15% increase
```

```
        v_individual_increase := v_new_salary - v_old_salary; -- Calculate the individual
increase



DBMS_OUTPUT.PUT_LINE(

            professor_rec.FirstName

            || ' '

            || professor_rec.LastName ||  '  '

            ||' $'

            || TO_CHAR(v_old_salary, '999,999.99')

            || ' ==> $'
            || TO_CHAR(v_new_salary, '999,999.99'));

        DBMS_OUTPUT.PUT_LINE('Total increase of: $'

            || TO_CHAR(v_individual_increase, '999,999.99'));

                                DBMS_OUTPUT.PUT_LINE('                                                ');
```

```
        DBMS_OUTPUT.PUT_LINE(''); -- Add a break line between professors

        v_total_salary_increase := v_total_salary_increase + v_individual_increase;    -- Accumulate total salary increase
    END LOOP;

    CLOSE c_professors;    -- Close cursor

    DBMS_OUTPUT.PUT_LINE('Total Salary Increase for all Professors ===> $'
        || TO_CHAR(v_total_salary_increase, '999,999.99')); -- Display total salary increase for all professors
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any exceptions
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

Question #1 OUTPUT:

```
Statement processed.
Professor salary increased by 15% *****
Minerva McGonagall   $  95,000.00 ==> $ 109,250.00
Total increase of: $  14,250.00

Pomona Sprout   $  78,000.00 ==> $  89,700.00
Total increase of: $  11,700.00

Filius Flitwick   $  82,000.00 ==> $  94,300.00
Total increase of: $  12,300.00

Severus Snape   $ 180,000.00 ==> $ 207,000.00
Total increase of: $  27,000.00

Alastor Moody   $  93,000.00 ==> $ 106,950.00
Total increase of: $  13,950.00

Sybill Trelawney   $  87,000.00 ==> $ 100,050.00
Total increase of: $  13,050.00

Septima Vector   $  88,000.00 ==> $ 101,200.00
Total increase of: $  13,200.00

Rolanda Hooch   $  75,000.00 ==> $  86,250.00
Total increase of: $  11,250.00

Rubeus Hagrid   $  88,500.00 ==> $ 101,775.00
Total increase of: $  13,275.00

Cuthbert Binns   $  80,000.00 ==> $  92,000.00
Total increase of: $  12,000.00

Aurora Sinistra   $  79,000.00 ==> $  90,850.00
Total increase of: $  11,850.00

Albus  Dumbledore    $ 200,000.00 ==> $ 230,000.00
Total increase of: $  30,000.00

Total Salary Increase for all Professors ===> $ 183,825.00
```

Question #2: – 83 Lines -- When provided with a HouseID, the program will provide statistics on all Students and related information to all students that are in that house. It will gather the total number of students in that house, total number of each gender and the average age.

```
-- Create functions to calculate statistics
CREATE OR REPLACE FUNCTION find_total_students(
    house_id_given IN Student.HouseID%TYPE
) RETURN NUMBER IS
    total_students NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO total_students
    FROM Student
    WHERE HouseID = house_id_given;

    RETURN total_students;
END;
/

CREATE OR REPLACE FUNCTION find_total_males(
    house_id_given IN Student.HouseID%TYPE
) RETURN NUMBER IS
    total_males NUMBER;
BEGIN
```

```
    SELECT COUNT(*)
    INTO total_males
    FROM Student
    WHERE HouseID = house_id_given
    AND Gender = 'Male';

    RETURN total_males;
END;
/

CREATE OR REPLACE FUNCTION find_total_females(
    house_id_given IN Student.HouseID%TYPE
) RETURN NUMBER IS
    total_females NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO total_females
    FROM Student
    WHERE HouseID = house_id_given
    AND Gender = 'Female';

    RETURN total_females;
END;
/
```

```
CREATE OR REPLACE FUNCTION find_average_age(
    house_id_given IN Student.HouseID%TYPE
) RETURN NUMBER IS
    avg_age NUMBER;
BEGIN
    SELECT AVG(Age)
    INTO avg_age
    FROM Student
    WHERE HouseID = house_id_given;

    RETURN avg_age;
END;
/

-- PL/SQL block to display statistics for a given HouseID
DECLARE
    v_house_id_given Student.HouseID%TYPE := 100;
    v_total_students NUMBER;
    v_total_males NUMBER;
    v_total_females NUMBER;
    v_avg_age NUMBER;
BEGIN
    v_total_students := find_total_students(v_house_id_given);
```

```
    v_total_males := find_total_males(v_house_id_given);
    v_total_females := find_total_females(v_house_id_given);
    v_avg_age := find_average_age(v_house_id_given);

    DBMS_OUTPUT.PUT_LINE('Statistics for HouseID ' || v_house_id_given);
    DBMS_OUTPUT.PUT_LINE('----------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Total Students: ' || v_total_students);
    DBMS_OUTPUT.PUT_LINE('Total Males: ' || v_total_males);
    DBMS_OUTPUT.PUT_LINE('Total Females: ' || v_total_females);
    DBMS_OUTPUT.PUT_LINE('Average Age: ' || v_avg_age);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

Question #2 Output:

```
Function created.

Function created.

Function created.

Function created.

Statement processed.
Statistics for HouseID 100
----------------------------------------
Total Students: 7
Total Males: 6
Total Females: 1
Average Age: 19.142857142857142857142857142857142857142857
```

Question #3: –74 lines, This program is going to iterate through the entire student table in preparation for the 10-year Hogwarts reunion, the program is going to print out all the information in the student table and it will display the current age of the students and how old the will be in 10 years.

```
DECLARE
   CURSOR
   student_cursor
   IS
      SELECT
   StudentID,
   FirstName,
   LastName,
   HouseID,
   Age,
   Gender
   FROM
   Student;


   student_id Student.StudentID%TYPE;
   first_name Student.FirstName%TYPE;
   last_name Student.LastName%TYPE;
   house_id Student.HouseID%TYPE;
   current_age Student.Age%TYPE;
   new_age Student.Age%TYPE;
   student_gender Student.Gender%TYPE; -- Corrected syntax
```

```
BEGIN
   OPEN
   student_cursor;
   LOOP
      BEGIN
         FETCH
      student_cursor
      INTO
      student_id,
      first_name,
      last_name,
      house_id,
      current_age,
      student_gender; -- Added missing INTO clause for student_gender
         EXIT WHEN
            student_cursor%NOTFOUND;

         -- Calculate new age after 10 years
         new_age := current_age + 10;

         -- Print student information
         DBMS_OUTPUT.PUT_LINE('Welcome to the Hogwarts 10 year anniversary');
```

```
        -- Added blank line for better readability
        DBMS_OUTPUT.PUT_LINE('   ');

        -- Corrected the concatenation and included student gender in the output
        DBMS_OUTPUT.PUT_LINE(first_name
                             || ' '
                             || last_name
                             || ', current age: '
                             || current_age
                             || ', house: '
                             || house_id
                             || ', gender: '
                             || student_gender  -- Included gender in the output
                             || ', age after 10 years: '
                             || new_age);

        -- Added blank line for better readability
        DBMS_OUTPUT.PUT_LINE('');

        -- Perform other operations with student data here

    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
```

```
    END;
  END LOOP;
  CLOSE student_cursor;
END;
/
```

Question #3 Output:

```
Statement processed.
Welcome to the Hogwarts 10 year anniversary

Harry Potter, current age: 17, house: 100, gender: Male, age after 10 years: 27
Welcome to the Hogwarts 10 year anniversary

Hermione Granger, current age: 19, house: 100, gender: Female, age after 10 years: 29
Welcome to the Hogwarts 10 year anniversary

Ron Weasley, current age: 18, house: 100, gender: Male, age after 10 years: 28
Welcome to the Hogwarts 10 year anniversary

Oliver Wood, current age: 22, house: 100, gender: Male, age after 10 years: 32
Welcome to the Hogwarts 10 year anniversary

Neville Longbottom, current age: 18, house: 100, gender: Male, age after 10 years: 28
Welcome to the Hogwarts 10 year anniversary

Fred Weasley, current age: 20, house: 100, gender: Male, age after 10 years: 30
Welcome to the Hogwarts 10 year anniversary

George Weasley, current age: 20, house: 100, gender: Male, age after 10 years: 30
Welcome to the Hogwarts 10 year anniversary

Cedric Diggory, current age: 9, house: 200, gender: Male, age after 10 years: 19
Welcome to the Hogwarts 10 year anniversary

Constance Pickering, current age: 17, house: 200, gender: Female, age after 10 years: 27
Welcome to the Hogwarts 10 year anniversary

Luna Lovegood, current age: 15, house: 300, gender: Female, age after 10 years: 25
Welcome to the Hogwarts 10 year anniversary

Cho Chang, current age: 17, house: 300, gender: Female, age after 10 years: 27
Welcome to the Hogwarts 10 year anniversary

Helena Ravenclaw, current age: 1016, house: 300, gender: Female, age after 10 years: 1026
Welcome to the Hogwarts 10 year anniversary

Draco Malfoy, current age: 18, house: 400, gender: Male, age after 10 years: 28
Welcome to the Hogwarts 10 year anniversary

Vincent Crabbe, current age: 19, house: 400, gender: Male, age after 10 years: 29
Welcome to the Hogwarts 10 year anniversary

Tom Riddle, current age: 71, house: 400, gender: Male, age after 10 years: 81
Welcome to the Hogwarts 10 year anniversary

Marcus Flint, current age: 23, house: 400, gender: Male, age after 10 years: 33
```

Question #4: --101 lines – This program is going to introduce new additions to previous textbooks that are now available. The school curriculum added a part 2 edition for every small textbook. Any textbook with less than 500 pages has introduced a part 2 edition. The program adds these new text to the database and displays all the values in the table.

```
DECLARE
 v_count NUMBER;
BEGIN
 -- Count the number of books that have less than 500 pages
 SELECT COUNT(*)
 INTO v_count
 FROM Books
 WHERE Pages < 500;

 -- Check if there are books with less than 500 pages
 IF v_count > 0 THEN
   -- Display introduction
   DBMS_OUTPUT.PUT_LINE('We would like to announce that a number of our textbooks have a Part 2.');

   -- Cursor to fetch all books
   FOR book IN (SELECT * FROM Books WHERE Pages < 500) LOOP
     -- Display book information before adding Part 2
```

```
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Book Title: ' || book.Title);
    DBMS_OUTPUT.PUT_LINE('Author: ' || book.Author);
    DBMS_OUTPUT.PUT_LINE('Year Published: ' || book.PublishYear);
    DBMS_OUTPUT.PUT_LINE('Number of Pages: ' || book.Pages);

    -- Truncate the ISBN value to fit within the defined length
    DECLARE
      v_isbn_part2 VARCHAR(13);
    BEGIN
      v_isbn_part2 := SUBSTR(book.ISBN, 1, 10) || '_2';

      -- Insert Part 2 for the book
      INSERT INTO Books (ISBN, Title, Author, Publisher, PublishYear, Genre, Pages)
      VALUES (v_isbn_part2, book.Title || ' Part 2', book.Author, book.Publisher,
book.PublishYear, book.Genre, book.Pages + 35); -- Add 35 pages for Part 2

      -- Display book information after adding Part 2
      DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
      DBMS_OUTPUT.PUT_LINE('Newly Added Part 2:');
      DBMS_OUTPUT.PUT_LINE('Book Title: ' || book.Title || ' Part 2');
      DBMS_OUTPUT.PUT_LINE('Author: ' || book.Author);
      DBMS_OUTPUT.PUT_LINE('Year Published: ' || book.PublishYear);
      DBMS_OUTPUT.PUT_LINE('Number of Pages: ' || (book.Pages + 35)); --
Display updated number of pages
```

```
    END;
  END LOOP;

  -- Display all data in the Books table after changes
  DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
  DBMS_OUTPUT.PUT_LINE('All Data in the Books Table After Changes:');

  FOR book_data IN (SELECT * FROM Books) LOOP
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_data.Title);
    DBMS_OUTPUT.PUT_LINE('Author: ' || book_data.Author);
    DBMS_OUTPUT.PUT_LINE('Year Published: ' || book_data.PublishYear);
    DBMS_OUTPUT.PUT_LINE('Number of Pages: ' || book_data.Pages);
  END LOOP;

 ELSE
  -- If no books have less than 500 pages
  DBMS_OUTPUT.PUT_LINE('No books found with less than 500 pages.');
 END IF;
END;
/
```

Question #4 Output:

```
Table truncated.
We would like to announce that a number of our textbooks have a Part 2.
---------------------------------------------
Book Title: Advanced Potion-Making
Author: Severus Snape
Year Published: 1997
Number of Pages: 320
---------------------------------------------
Newly Added Part 2:
Book Title: Advanced Potion-Making Part 2
Author: Severus Snape
Year Published: 1997
Number of Pages: 355
---------------------------------------------
Book Title: Fantastic Beasts & Where To Find Them
Author: Newt Scamander
Year Published: 1998
Number of Pages: 352
---------------------------------------------
Newly Added Part 2:
Book Title: Fantastic Beasts & Where To Find Them Part 2
Author: Newt Scamander
Year Published: 1998
Number of Pages: 387
---------------------------------------------
Book Title: Rune Dictionary
Author: Edwardus Lima
Year Published: 1999
Number of Pages: 448
---------------------------------------------
Newly Added Part 2:
Book Title: Rune Dictionary Part 2
Author: Edwardus Lima
Year Published: 1999
Number of Pages: 483
---------------------------------------------
All Data in the Books Table After Changes:
---------------------------------------------
```

ISBN	TITLE	AUTHOR	PUBLISHER	PUBLISHYEAR	GENRE	PAGES
9780545582989	Advanced Potion-Making	Severus Snape	Hogwarts Scholastic Inc.	1997	Non-Fiction	320
9780439139601	Fantastic Beasts & Where To Find Them	Newt Scamander	Magical Publications	1998	Non-Fiction	352
9780439554930	Rune Dictionary	Edwardus Lima	Magical Publications	1999	Non-Fiction	448
9780439785969	One Thousand Magical Herbs & Fungi	Phyllida Spore	Green Goblin Books LLC	2000	Non-Fiction	752
9780439358071	A History Of Magic	Bathilda Bagshot	Little Red Books	2003	History	870
9780439023528	Beginners Guide to Transfiguration	Emeric Switch	Hogwarts Scholastic Inc.	2005	Non-Fiction	652
9780545010221	Dark Forces: A Guide to Self-Protection	Quentin Trimble	Little Red Books	2007	Narrative	784
9780765348270	Quidditch Champions Guide	Wilbert Slinkhard	Green Goblin Books LLC	2003	Article	544
9780812513714	Extreme Incantations	Violeta Stitch	Magical Publications	1990	Non-Fiction	814
9780140449259	The Dream Oracle	Inigo Imago	Little Red Books	1844	Spirituality	704
9780545582_2	Advanced Potion-Making Part 2	Severus Snape	Hogwarts Scholastic Inc.	1997	Non-Fiction	355
9780439139_2	Fantastic Beasts & Where To Find Them Part 2	Newt Scamander	Magical Publications	1998	Non-Fiction	387
9780439554_2	Rune Dictionary Part 2	Edwardus Lima	Magical Publications	1999	Non-Fiction	483

Question #5: - 82 Lines – This program will iterate all the information in the Wands table, student table and join the House table to be able to gather information on the wands and who the wands belong while giving information about wand and owner.

```
DECLARE
    WandID INT;
    WoodType VARCHAR(50);
    CoreType VARCHAR(50);
    Length DECIMAL(4, 2);
    Flexibility VARCHAR(20);
    WandStudentID INT;
    WandYearMade INT;

    StudentID INT;
    FirstName VARCHAR(50);
    LastName VARCHAR(50);
    Gender VARCHAR(10);
    Age INT;
    HouseID INT;
    BloodStatus VARCHAR(50);

    HouseName VARCHAR(50);

    WandBelongsToStudent VARCHAR(255);
```

```
CURSOR wand_cursor IS
    SELECT
    WandID,
    WoodType,
    CoreType,
    Length,
    Flexibility,
    StudentID,
    YearMade
    FROM Wands;

CURSOR student_cursor
    IS
    SELECT
    s.StudentID,
    s.FirstName,
    s.LastName,
    s.Gender,
    s.Age,
    s.HouseID,
    s.BloodStatus,
    h.HouseName
    FROM Student s
```

```
    INNER JOIN House h ON s.HouseID = h.HouseID;


BEGIN
  OPEN wand_cursor;
  LOOP
    FETCH wand_cursor
    INTO WandID,
    WoodType,
    CoreType,
    Length,
    Flexibility,
    WandStudentID,
    WandYearMade;
    EXIT WHEN wand_cursor%NOTFOUND;

    -- Find the corresponding student and house for the wand
    FOR student_rec IN student_cursor
    LOOP
      IF student_rec.StudentID = WandStudentID THEN
        WandBelongsToStudent := student_rec.FirstName || ' ' ||
student_rec.LastName;
        HouseName := student_rec.HouseName;
        Age := student_rec.Age; -- Assigning the age of the student
        EXIT;
      END IF;
```

```
        END LOOP;


        -- Output wand information, student, and house
        DBMS_OUTPUT.PUT_LINE('This wand belongs to: ' || WandBelongsToStudent
|| ', Age: ' || Age || ', House: ' || HouseName);
                DBMS_OUTPUT.PUT_LINE('          ');
        DBMS_OUTPUT.PUT_LINE('Wand Details: WandID: ' || WandID || ', WoodType:
' || WoodType || ', CoreType: ' || CoreType || ', Length: ' || Length || ', Flexibility: ' ||
Flexibility || ', YearMade: ' || WandYearMade);
                DBMS_OUTPUT.PUT_LINE('          ');

DBMS_OUTPUT.PUT_LINE('====================================')
;


    END LOOP;


    CLOSE wand_cursor;
END;
/
```

Question #5 Output:

```
Statement processed.
This wand belongs to: Harry Potter, Age: 17, House: Gryffindor
Wand Details: WandID: 116, WoodType: Oak, CoreType: Phoenix Feather, Length: 10.25, Flexibility: Soft, YearMade: 1800
====================================
This wand belongs to: Hermione Granger, Age: 19, House: Gryffindor
Wand Details: WandID: 215, WoodType: Maple, CoreType: Dragon Heartstring, Length: 11.5, Flexibility: Flexible, YearMade: 1805
====================================
This wand belongs to: Ron Weasley, Age: 18, House: Gryffindor
Wand Details: WandID: 314, WoodType: Redwood, CoreType: Unicorn Hair, Length: 9.75, Flexibility: Rigid, YearMade: 1903
====================================
This wand belongs to: Oliver Wood, Age: 22, House: Gryffindor
Wand Details: WandID: 413, WoodType: Ebony, CoreType: Thestral Tail Hair, Length: 12, Flexibility: Soft, YearMade: 1850
====================================
This wand belongs to: Neville Longbottom, Age: 18, House: Gryffindor
Wand Details: WandID: 512, WoodType: Willow, CoreType: Veela Hair, Length: 10.75, Flexibility: Flexible, YearMade: 1899
====================================
This wand belongs to: Fred Weasley, Age: 20, House: Gryffindor
Wand Details: WandID: 612, WoodType: Mahogany, CoreType: Phoenix Feather, Length: 10, Flexibility: Rigid, YearMade: 1930
====================================
This wand belongs to: George Weasley, Age: 20, House: Gryffindor
Wand Details: WandID: 711, WoodType: Rosewood, CoreType: Hippogriff Feather, Length: 11.25, Flexibility: Flexible, YearMade: 1823
====================================
This wand belongs to: Cedric Diggory, Age: 9, House: Hufflepuff
Wand Details: WandID: 810, WoodType: Ash, CoreType: Thunderbird Tail Feather, Length: 10.5, Flexibility: Soft, YearMade: 1870
====================================
This wand belongs to: Constance Pickering, Age: 17, House: Hufflepuff
Wand Details: WandID: 991, WoodType: Cherry, CoreType: Basilisk Horn, Length: 9.5, Flexibility: Rigid, YearMade: 1950
====================================
This wand belongs to: Luna Lovegood, Age: 15, House: Ravenclaw
Wand Details: WandID: 102, WoodType: Vine, CoreType: Kelpie Hair, Length: 10.25, Flexibility: Flexible, YearMade: 1980
====================================
This wand belongs to: Cho Chang, Age: 17, House: Ravenclaw
Wand Details: WandID: 113, WoodType: Yew, CoreType: Dragon Heartstring, Length: 11, Flexibility: Rigid, YearMade: 1805
====================================
This wand belongs to: Helena Ravenclaw, Age: 1016, House: Ravenclaw
Wand Details: WandID: 124, WoodType: Holly, CoreType: Phoenix Feather, Length: 10.75, Flexibility: Soft, YearMade: 1867
====================================
This wand belongs to: Draco Malfoy, Age: 18, House: Slytherin
Wand Details: WandID: 135, WoodType: Blackthorn, CoreType: Unicorn Hair, Length: 9.75, Flexibility: Flexible, YearMade: 1877
====================================
This wand belongs to: Vincent Crabbe, Age: 19, House: Slytherin
Wand Details: WandID: 146, WoodType: Pear, CoreType: Thestral Tail Hair, Length: 12.25, Flexibility: Rigid, YearMade: 1917
====================================
This wand belongs to: Tom Riddle, Age: 71, House: Slytherin
Wand Details: WandID: 157, WoodType: Walnut, CoreType: Veela Hair, Length: 10.5, Flexibility: Soft, YearMade: 1800
====================================
This wand belongs to: Marcus Flint, Age: 23, House: Slytherin
Wand Details: WandID: 168, WoodType: Hawthorn, CoreType: Phoenix Feather, Length: 10.25, Flexibility: Flexible, YearMade: 1880
====================================
```

Question #6: -84 Lines This block of code will display the 4 types of spells that are taught in the class, and it will further display all the spells of a certain category. It will count the amount and display the names. It will than add an additional 3 spells to one of the types.

```
DECLARE
   DefensiveSpellCount INT := 0;
   AttackSpellCount INT := 0;
   CharmSpellCount INT := 0;
   CombatSpellCount INT := 0;

   DefensiveSpellList VARCHAR2(4000) := '';
   AttackSpellList VARCHAR2(4000) := '';
   CharmSpellList VARCHAR2(4000) := '';
   CombatSpellList VARCHAR2(4000) := '';

   CURSOR
      spells_cursor
      IS
      SELECT
      SpellName, Type
      FROM
      Spells;
BEGIN
   FOR
   spell_rec IN spells_cursor
```

```
  LOOP
    IF
            spell_rec.Type = 'Defensive'
   THEN
      DefensiveSpellCount := DefensiveSpellCount + 1;
      DefensiveSpellList := DefensiveSpellList || spell_rec.SpellName || ', ';
    ELSIF
      spell_rec.Type = 'Attack'
      THEN
      AttackSpellCount := AttackSpellCount + 1;
      AttackSpellList := AttackSpellList || spell_rec.SpellName || ', ';
    ELSIF
      spell_rec.Type = 'Charm'
      THEN
      CharmSpellCount := CharmSpellCount + 1;
      CharmSpellList := CharmSpellList || spell_rec.SpellName || ', ';
    ELSIF
      spell_rec.Type = 'Combat'
      THEN
      CombatSpellCount := CombatSpellCount + 1;
      CombatSpellList := CombatSpellList || spell_rec.SpellName || ', ';
    END IF;
  END LOOP;
```

```
   -- Add the new Combat Spells
   CombatSpellCount := CombatSpellCount + 3; -- Increment the count for the new spells
   CombatSpellList := CombatSpellList || 'Firewave, Water Blast, Thunder, '; -- Add the new spells to the list


   -- Print the counts and lists
   DBMS_OUTPUT.PUT_LINE('Defensive spells have a total of '
      || DefensiveSpellCount
      || ' spells');
   DBMS_OUTPUT.PUT_LINE('The '
      || DefensiveSpellCount
         || ' spells are: '
      || DefensiveSpellList);
   DBMS_OUTPUT.PUT_LINE('======================');
   DBMS_OUTPUT.PUT_LINE('Attack spells have a total of '
      || AttackSpellCount
      || ' spells');
   DBMS_OUTPUT.PUT_LINE('The '
      || AttackSpellCount
      || ' spells are: '
      || AttackSpellList);
   DBMS_OUTPUT.PUT_LINE('======================');
   DBMS_OUTPUT.PUT_LINE('Charm spells have a total of '
      || CharmSpellCount
```

```
    || ' spells');
  DBMS_OUTPUT.PUT_LINE('The '
    || CharmSpellCount
    || ' spells are: '
    || CharmSpellList);
  DBMS_OUTPUT.PUT_LINE('======================');
  DBMS_OUTPUT.PUT_LINE('Combat spells have a total of '
    || CombatSpellCount
    || ' spells');
  DBMS_OUTPUT.PUT_LINE('The '
    || CombatSpellCount
    || ' spells are: '
    || CombatSpellList);


END;
/
```

Question #6 Output:

```
Statement processed.
Defensive spells have a total of 3 spells
The 3 spells are: Protego, Expecto Patronum, Patronus,
======================
Attack spells have a total of 5 spells
The 5 spells are: Stupefy, Petrificus Totalus, Expulso, Sectumsempra, Cruciatus Curse,
======================
Charm spells have a total of 5 spells
The 5 spells are: Lumos, Accio, Alohomora, Incendio, Aguamenti,
======================
Combat spells have a total of 4 spells
The 4 spells are: Expelliarmus, Firewave, Water Blast, Thunder,
```

Question #7: -80 Lines – This Program is going to introduce a new club to the Clubs table. It will iterate all the Classrooms to find the largest one to hold the event. It will iterate all the clubs to display the club name and number of participants to formally invite and it will create a new club that will be added to the database.

```
DECLARE
   GreatRoomName VARCHAR2(50);
   GreatHallCapacity INT;
   MaxCapacity INT := 0;
   MaxCapacityRoomName VARCHAR2(255);

   -- Cursor to fetch the Great Hall information
   CURSOR
      room_cursor
      IS
      SELECT
      RoomName,
      Capacity
      FROM
      Classroom
      WHERE
      RoomName = 'The Great Hall';

   -- Cursor to fetch information about clubs
   CURSOR
```

```
    clubs_cursor
    IS
    SELECT
    ClubName,
    Participants
    FROM
    Clubs;

BEGIN
  -- Retrieve information about the Great Hall
  OPEN
  room_cursor;
  FETCH
    room_cursor
    INTO
    GreatRoomName,
    GreatHallCapacity;
  CLOSE
    room_cursor;

  -- Iterate through all classrooms to find the one with the maximum capacity
  FOR room_rec IN
    (SELECT RoomName, Capacity FROM Classroom)
    LOOP
```

```
    IF
    room_rec.Capacity > MaxCapacity
    THEN
        MaxCapacity := room_rec.Capacity;
        MaxCapacityRoomName := room_rec.RoomName;
    END IF;
END LOOP;

-- Insert information about the Sports Club into the Clubs table
INSERT INTO
    Clubs (ClubID, ClubName, Participants, Advisor, ProfessorID)
VALUES
    ('SC01', 'Sports Club', 14, 'Professor Dumbledore', 999);

-- Print the invitation and information about all clubs
DBMS_OUTPUT.PUT_LINE('We will have our annual club meeting held at "'
    || GreatRoomName
    || '" that can sit '
    || GreatHallCapacity
    || ' people.');

-- Cursor through all clubs and print their names and participants
FOR club_rec IN clubs_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('We formally invite "'
```

```
        || club_rec.ClubName
        || '" and all '
        || club_rec.Participants
        || ' participants.');
    END LOOP;

    -- Print information about the newest club
    DBMS_OUTPUT.PUT_LINE('And we would love to announce our newest club "Sports Club" with a total of 14 participants.');

END;
/
Select * from Clubs;
```

Question #7 Output:

```
Statement processed.
We will have our annual club meeting held at "The Great Hall" that can sit 500 people.
We formally invite "Dueling Club" and all 12 participants.
We formally invite "Frog Choir" and all 15 participants.
We formally invite "Chess Club" and all 10 participants.
We formally invite "Astronomy Club" and all 18 participants.
We formally invite "Magical Creature Club" and all 8 participants.
We formally invite "Sports Club" and all 14 participants.
And we would love to announce our newest club "Sports Club" with a total of 14 participants.
```

CLUBID	CLUBNAME	CLUBLEADER	PARTICIPANTS	ADVISOR	PROFESSORID
DC12	Dueling Club	Harry Potter	12	Professor Snape	974
FC02	Frog Choir	Cho Chang	15	Professor Flitwick	324
CC88	Chess Club	Ron Weasley	10	Professor McGonagall	246
AC68	Astronomy Club	Luna Lovegood	18	Professor Sinistra	773
MCM71	Magical Creature Club	Neville Longbottom	8	Professor Hagrid	654
SC01	Sports Club	-	14	Professor Dumbledore	999

Question #8: -74 Lines – It is time to award the winning house with the most points the House Cup. This program will iterate through the Housepoints table to mathematically figure out which House accumulated the most points and which students contributed and finally awarding the house with the most points the award.

```
DECLARE
   WinningHouse VARCHAR2(50);
   MaxPoints INT := 0;


   -- Cursor to fetch house points information
   CURSOR
      house_points_cursor
      IS
      SELECT
      HouseName,
      SUM(PointsEarned)
      AS
      TotalPoints
      FROM
      HousePoints
      GROUP BY
      HouseName
      ORDER BY
      TotalPoints DESC;
```

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('        ');
   DBMS_OUTPUT.PUT_LINE('    Welcome to the Award ceremony for House Cup!
');


   -- Iterate through house points to find the winning house
   FOR
      house_points_rec
      IN
      house_points_cursor LOOP
      IF
      house_points_rec.TotalPoints > MaxPoints
      THEN
         MaxPoints := house_points_rec.TotalPoints;
         WinningHouse := house_points_rec.HouseName;
      END IF;


      -- Print information about each house
                DBMS_OUTPUT.PUT_LINE('        ');
      DBMS_OUTPUT.PUT_LINE('House '
         || house_points_rec.HouseName
         || ' has a total of '
         || house_points_rec.TotalPoints
         || ' Points');
                DBMS_OUTPUT.PUT_LINE('        ');
```

```
        -- List the students who earned points for this house
        FOR
           student_rec
           IN
           (SELECT s.FirstName
           || ' '
           || s.LastName AS StudentName, hp.PointsEarned
                       FROM Student s
                       JOIN HousePoints hp ON s.StudentID = hp.StudentID
                       WHERE hp.HouseName = house_points_rec.HouseName)
           LOOP
           DBMS_OUTPUT.PUT_LINE('"'
           || student_rec.StudentName
           || '" earned '
           || student_rec.PointsEarned
           || ' points');
        END LOOP;

        -- Add an extra line for readability
                DBMS_OUTPUT.PUT_LINE('        ');

DBMS_OUTPUT.PUT_LINE('=====================================
======');
                DBMS_OUTPUT.PUT_LINE('    ');
    END LOOP;
```

```
    -- Print the winning house
    DBMS_OUTPUT.PUT_LINE('The House cup award goes to "'
        || WinningHouse
        || '"');
END;
/
```

Question #8 Output:

```
Statement processed.

    Welcome to the Award ceremony for House Cup!

House Slytherin has a total of 590 Points

"Draco Malfoy" earned 140 points
"Vincent Crabbe" earned 125 points
"Tom Riddle" earned 210 points
"Marcus Flint" earned 115 points

========================================


House Gryffindor has a total of 515 Points

"Harry Potter" earned 150 points
"Hermione Granger" earned 120 points
"Ron Weasley" earned 110 points
"Neville Longbottom" earned 135 points

========================================


House Ravenclaw has a total of 460 Points

"Luna Lovegood" earned 145 points
"Cho Chang" earned 160 points
"Helena Ravenclaw" earned 155 points

========================================


House Hufflepuff has a total of 210 Points

"Cedric Diggory" earned 110 points
"Constance Pickering" earned 100 points

========================================

The House cup award goes to "Slytherin"
```

Question# 9: -99 Lines – This iterates all of the information from the Teams table. It will calculate the number of wins and compare to see which team has the most wins throughout the season. It will go through the players and display their information such as first name and position.

```
DECLARE
  TeamNameVar VARCHAR2(50);
  TeamMascotVar VARCHAR2(50);
  PlayerCountVar INT;
  TotalWinsVar INT;
  MaxWinsVar INT := 0; -- Variable to store the maximum number of wins
  WinningTeamVar VARCHAR2(50); -- Variable to store the name of the team with the most wins

  -- Cursor to fetch team information
  CURSOR
    team_cursor
    IS
    SELECT
    TeamName,
    TeamMascot,
    COUNT(*) AS
    PlayerCount,
    SUM(Wins)
    AS
    TotalWins
```

```
    FROM
    Team
    GROUP BY
    TeamName,
    TeamMascot
    ORDER BY
    TotalWins DESC; -- Ordering by total wins descending

  -- Cursor to fetch player information for each team
  CURSOR
    player_cursor(p_team_name VARCHAR2)
    IS
    SELECT
    s.FirstName
    || ' '
    || s.LastName
    AS PlayerName,
    t.Position
    FROM
    Team t
    JOIN
    Student s
    ON
    t.StudentID = s.StudentID
```

```
    WHERE
    t.TeamName = p_team_name;

BEGIN
  -- Iterate through teams
  FOR
  team_rec
  IN
  team_cursor LOOP
    -- Store team information
    TeamNameVar := team_rec.TeamName;
    TeamMascotVar := team_rec.TeamMascot;
    PlayerCountVar := team_rec.PlayerCount;
    TotalWinsVar := team_rec.TotalWins;

    -- Print team information
    DBMS_OUTPUT.PUT_LINE('=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=');
    DBMS_OUTPUT.PUT_LINE('Team Name: '
      || TeamNameVar);
    DBMS_OUTPUT.PUT_LINE('Mascot: '
      || TeamMascotVar);
    DBMS_OUTPUT.PUT_LINE('Number of Players: '
      || PlayerCountVar);
```

```
        -- Print player information for the team
        DBMS_OUTPUT.PUT_LINE('----Players------');
        FOR player_rec IN player_cursor(team_rec.TeamName) LOOP
            DBMS_OUTPUT.PUT_LINE(player_rec.PlayerName
            || ' - '
            || player_rec.Position);
        END LOOP;

        -- Print total wins for the team
        DBMS_OUTPUT.PUT_LINE('-----Total Wins: ' || TotalWinsVar);

        -- Compare total wins with the maximum wins
        IF
            TotalWinsVar > MaxWinsVar
            THEN
            MaxWinsVar := TotalWinsVar; -- Update maximum wins
            WinningTeamVar := TeamNameVar; -- Update winning team
        END IF;

        DBMS_OUTPUT.PUT_LINE('');
    END LOOP;

    -- Print the winning team with the most wins
    DBMS_OUTPUT.PUT_LINE('-------------------------------------');
```

```
    DBMS_OUTPUT.PUT_LINE('The team with the most wins is: '
        || WinningTeamVar
        || ' with '
        || MaxWinsVar
        || ' wins.');
END;
/
```

Question #9 Output:

```
Statement processed.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Team Name: Gryffindor
Mascot: Lion
Number of Players: 5
----Players------
Harry Potter - Seeker
Ron Weasley - Beater
Oliver Wood - Chaser
Fred Weasley - Chaser
George Weasley - Chaser
-----Total Wins: 100
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Team Name: Slytherin
Mascot: Serpent
Number of Players: 4
----Players------
Draco Malfoy - Seeker
Vincent Crabbe - Beater
Tom Riddle - Beater
Marcus Flint - Chaser
-----Total Wins: 92
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Team Name: Ravenclaw
Mascot: Eagle
Number of Players: 3
----Players------
Luna Lovegood - Chaser
Cho Chang - Chaser
Helena Ravenclaw - Seeker
-----Total Wins: 51
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Team Name: Hufflepuff
Mascot: Badger
Number of Players: 2
----Players------
Cedric Diggory - Chaser
Constance Pickering - Keeper
-----Total Wins: 22
---------------------------------------
The team with the most wins is: Gryffindor with 100 wins.
```

Question #10: -146 Lines, This program is intended to look-up all information regarding a student. Once a Student ID is entered via user input, the program will itereate all information in the database and present the information regarding the student such as name, age, House, enrolled classes and teams, wand type etc

```
DECLARE
    v_student_id INT := 111111; -- Replace with the actual StudentID you want to query

    v_student_first_name Student.FirstName%TYPE;
    v_student_last_name Student.LastName%TYPE;
    v_student_age Student.Age%TYPE;
    v_student_house House.HouseName%TYPE;
    v_student_blood_status Student.BloodStatus%TYPE;
    v_enrollment_cursor SYS_REFCURSOR;
    v_clubs_cursor SYS_REFCURSOR;
    v_teams_cursor SYS_REFCURSOR;
    v_wand_cursor SYS_REFCURSOR;
    v_course_name Course.CourseName%TYPE; -- Declare v_course_name
    v_club_name Clubs.ClubName%TYPE;
    v_team_name Team.TeamName%TYPE;
    v_wood_type Wands.WoodType%TYPE;
    v_core_type Wands.CoreType%TYPE;
```

```
    v_length Wands.Length%TYPE;
    v_flexibility Wands.Flexibility%TYPE;
    v_points_earned NUMBER;
    v_professor_name VARCHAR2(100); -- Change data type to VARCHAR2
    v_professors_cursor SYS_REFCURSOR;
BEGIN
    -- Retrieve student information
    SELECT FirstName, LastName, Age, HouseName, BloodStatus
    INTO v_student_first_name, v_student_last_name, v_student_age,
v_student_house, v_student_blood_status
    FROM Student s
    JOIN House h ON s.HouseID = h.HouseID
    WHERE StudentID = v_student_id;

    -- Open cursors to retrieve additional information
    OPEN v_enrollment_cursor FOR
        SELECT c.CourseName
        FROM Enrollment e
        JOIN Course c ON e.CourseID = c.CourseID
        WHERE e.StudentID = v_student_id;

    OPEN v_clubs_cursor FOR
        SELECT ClubName
        FROM Clubs
        WHERE ClubID IN (
```

```
        SELECT ClubID
        FROM Enrollment
        WHERE StudentID = v_student_id
    );

  OPEN v_teams_cursor FOR
    SELECT TeamName
    FROM Team
    WHERE StudentID = v_student_id;

  OPEN v_wand_cursor FOR
    SELECT WoodType, CoreType, Length, Flexibility
    FROM Wands
    WHERE StudentID = v_student_id;

  SELECT SUM(PointsEarned)
  INTO v_points_earned
  FROM HousePoints
  WHERE StudentID = v_student_id;

  OPEN v_professors_cursor FOR
    SELECT DISTINCT p.FirstName || ' ' || p.LastName AS ProfessorName
    FROM Professor p
    JOIN Course c ON p.ProfessorID = c.ProfessorID
```

```
    JOIN Enrollment e ON c.CourseID = e.CourseID
    WHERE e.StudentID = v_student_id;

  -- Print student information
        DBMS_OUTPUT.PUT_LINE('    ');
  DBMS_OUTPUT.PUT_LINE('***** Student Information:');
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_student_first_name || ' ' ||
v_student_last_name);
  DBMS_OUTPUT.PUT_LINE('Age: ' || v_student_age);
  DBMS_OUTPUT.PUT_LINE('House: ' || v_student_house);
  DBMS_OUTPUT.PUT_LINE('Blood Status: ' || v_student_blood_status);
  DBMS_OUTPUT.PUT_LINE('');

  -- Print enrolled classes
        DBMS_OUTPUT.PUT_LINE('    ');
  DBMS_OUTPUT.PUT_LINE('***** Classes Enrolled:');
  LOOP
    FETCH v_enrollment_cursor INTO v_course_name;
    EXIT WHEN v_enrollment_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('- ' || v_course_name);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('');

  -- Print enrolled clubs
        DBMS_OUTPUT.PUT_LINE('    ');
```

```
    DBMS_OUTPUT.PUT_LINE('***** Clubs Enrolled:');
    LOOP
        FETCH v_clubs_cursor INTO v_club_name;
        EXIT WHEN v_clubs_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('- ' || v_club_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');

    -- Print enrolled teams
            DBMS_OUTPUT.PUT_LINE('    ');
    DBMS_OUTPUT.PUT_LINE('***** Teams Enrolled:');
    LOOP
        FETCH v_teams_cursor INTO v_team_name;
        EXIT WHEN v_teams_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('- ' || v_team_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');

    -- Print wand details
            DBMS_OUTPUT.PUT_LINE('    ');
    DBMS_OUTPUT.PUT_LINE('***** Wand Details:');
    FETCH v_wand_cursor INTO v_wood_type, v_core_type, v_length, v_flexibility;
    IF v_wand_cursor%FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Wood Type: ' || v_wood_type);
```

```
        DBMS_OUTPUT.PUT_LINE('Core Type: ' || v_core_type);
        DBMS_OUTPUT.PUT_LINE('Length: ' || v_length);
        DBMS_OUTPUT.PUT_LINE('Flexibility: ' || v_flexibility);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No wand details found.');
    END IF;
    DBMS_OUTPUT.PUT_LINE('');

    -- Print total points earned
    DBMS_OUTPUT.PUT_LINE('Total Points Earned: ' || v_points_earned);
    DBMS_OUTPUT.PUT_LINE('');

    -- Print professors
            DBMS_OUTPUT.PUT_LINE('    ');
    DBMS_OUTPUT.PUT_LINE('***** Professors:');
    LOOP
        FETCH v_professors_cursor INTO v_professor_name;
        EXIT WHEN v_professors_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('- ' || v_professor_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');

    -- Close cursors
    CLOSE v_enrollment_cursor;
```

```
    CLOSE v_clubs_cursor;
    CLOSE v_teams_cursor;
    CLOSE v_wand_cursor;
    CLOSE v_professors_cursor;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

Question #10 Output:

```
Statement processed.

***** Student Information:
Name: Harry Potter
Age: 17
House: Gryffindor
Blood Status: Half-Blood

***** Classes Enrolled:
- Potions
- Transfiguration
- Defense Against the Dark Arts

***** Clubs Enrolled:
- Dueling Club
- Frog Choir
- Chess Club
- Astronomy Club
- Magical Creature Club

***** Teams Enrolled:
- Gryffindor

***** Wand Details:
Wood Type: Oak
Core Type: Phoenix Feather
Length: 10.25
Flexibility: Soft
Total Points Earned: 150

***** Professors:
- Severus Snape
- Minerva McGonagall
- Alastor Moody
```

Question #11 – 81 Lines; In this program the Professors are getting another 5% raise, the code block will increase all of the professor salaries in a newly created Professors table, it will update all the salaries by 5% and a trigger code block will be triggered every time the salary is updated, it will display all the updates in a specific log table.

```
Drop table Professor2;
Drop table SalaryIncreaseLog;


Create table
   Professor2
   as select * from Professor;


CREATE TABLE
   SalaryIncreaseLog (
   ProfessorID INT,
   Date_time TIMESTAMP(6),
   Salary_old INT,
   Salary_new INT
);



CREATE OR REPLACE TRIGGER Salary_Log
AFTER UPDATE ON Professor2
FOR EACH ROW
BEGIN
```

```
    IF :OLD.Salary != :NEW.Salary
    THEN
       INSERT INTO
    SalaryIncreaseLog
    (PROFESSORID, Date_time, Salary_old, Salary_new)
       VALUES
                (:NEW.ProfessorID, SYSDATE, :OLD.Salary, :NEW.Salary);
    END IF;
END;
/

UPDATE
    Professor2
    SET
    Salary = Salary * 1.05
    WHERE
    Salary  >= 0;

DECLARE
    v_old_salary Professor2.Salary%TYPE;
    v_new_salary Professor2.Salary%TYPE;
    v_difference NUMBER;
BEGIN
    -- Update salaries by 5%
```

```
    UPDATE Professor2
    SET Salary = Salary * 1.05;

    -- Display message indicating everyone is getting a raise
    DBMS_OUTPUT.PUT_LINE('Everyone is getting a raise!!');
    DBMS_OUTPUT.PUT_LINE('ProfessorID | First Name | Old Salary | New Salary |
Difference');

    -- Retrieve and display information for each professor
    FOR prof IN
        (SELECT
        p.ProfessorID,
        p.FirstName,
        p.LastName,
        s.Salary_old,
        s.Salary_new
                FROM Professor2 p
                JOIN SalaryIncreaseLog s
                                ON p.ProfessorID = s.ProfessorID)
    LOOP
        v_old_salary := prof.Salary_old;
        v_new_salary := prof.Salary_new;
        v_difference := v_new_salary - v_old_salary;

        -- Display professor information
```

```
        DBMS_OUTPUT.PUT_LINE(prof.ProfessorID
            || '         |    '
            || prof.FirstName
            || '      $'
            || v_old_salary
            || '   =>    $'
            || v_new_salary
            || '      $'
            || v_difference);
    END LOOP;
END;
/
Select * from SalaryIncreaseLog;
```

Question #11 Output:

```
Table dropped.

Table dropped.

Table created.

Table created.

Trigger created.

12 row(s) updated.

Statement processed.
Everyone is getting a raise!!
ProfessorID | First Name | Old Salary | New Salary | Difference
246         |    Minerva      $95000   =>    $99750      $4750
420         |    Pomona      $78000   =>    $81900      $3900
324         |    Filius      $82000   =>    $86100      $4100
974         |    Severus      $180000   =>    $189000      $9000
804         |    Alastor      $93000   =>    $97650      $4650
113         |    Sybill      $87000   =>    $91350      $4350
625         |    Septima      $88000   =>    $92400      $4400
882         |    Rolanda      $75000   =>    $78750      $3750
654         |    Rubeus      $88500   =>    $92925      $4425
591         |    Cuthbert      $80000   =>    $84000      $4000
773         |    Aurora      $79000   =>    $82950      $3950
999         |    Albus      $200000   =>    $210000      $10000
246         |    Minerva      $99750   =>    $104738      $4988
420         |    Pomona      $81900   =>    $85995      $4095
324         |    Filius      $86100   =>    $90405      $4305
974         |    Severus      $189000   =>    $198450      $9450
804         |    Alastor      $97650   =>    $102533      $4883
113         |    Sybill      $91350   =>    $95918      $4568
625         |    Septima      $92400   =>    $97020      $4620
882         |    Rolanda      $78750   =>    $82688      $3938
654         |    Rubeus      $92925   =>    $97571      $4646
591         |    Cuthbert      $84000   =>    $88200      $4200
773         |    Aurora      $82950   =>    $87098      $4148
999         |    Albus      $210000   =>    $220500      $10500
```

PROFESSORID	DATE_TIME	SALARY_OLD	SALARY_NEW
246	28-APR-24 03.19.13.000000 AM	95000	99750
420	28-APR-24 03.19.13.000000 AM	78000	81900
324	28-APR-24 03.19.13.000000 AM	82000	86100
974	28-APR-24 03.19.13.000000 AM	180000	189000
804	28-APR-24 03.19.13.000000 AM	93000	97650
113	28-APR-24 03.19.13.000000 AM	87000	91350
625	28-APR-24 03.19.13.000000 AM	88000	92400
882	28-APR-24 03.19.13.000000 AM	75000	78750
654	28-APR-24 03.19.13.000000 AM	88500	92925
591	28-APR-24 03.19.13.000000 AM	80000	84000
773	28-APR-24 03.19.13.000000 AM	79000	82950
999	28-APR-24 03.19.13.000000 AM	200000	210000
246	28-APR-24 03.19.13.000000 AM	99750	104738
420	28-APR-24 03.19.13.000000 AM	81900	85995
324	28-APR-24 03.19.13.000000 AM	86100	90405
974	28-APR-24 03.19.13.000000 AM	189000	198450
804	28-APR-24 03.19.13.000000 AM	97650	102533
113	28-APR-24 03.19.13.000000 AM	91350	95918
625	28-APR-24 03.19.13.000000 AM	92400	97020
882	28-APR-24 03.19.13.000000 AM	78750	82688
654	28-APR-24 03.19.13.000000 AM	92925	97571
591	28-APR-24 03.19.13.000000 AM	84000	88200
773	28-APR-24 03.19.13.000000 AM	82950	87098
999	28-APR-24 03.19.13.000000 AM	210000	220500

Question #12 – 82 LINES; The school is preparing for Graduation and it wants to clear up the School Database, This line of code, deletes students who are preparing to graduate. It creates a 2nd Student table called Student2 and it removes all the students who are over the age of 18, it logs all the deletions using a Trigger method and logs them into the GraduationLog table. It then cursors and iterates the GraduationLog to create a Congratulations message for the grades and iterates and cursors the remaining students wishing them luck on next years graduation.

```
DROP TABLE GraduationLog;

DROP TABLE Student2;


-- Create Students2 table

CREATE TABLE

   Student2

   AS SELECT *

   FROM Student;


-- Create GraduationLog table

CREATE TABLE

   GraduationLog (

   StudentID INT,

   Name VARCHAR2(100),

   GraduationDate TIMESTAMP(6)

);


-- Create trigger to log deletions in GraduationLog

CREATE OR REPLACE TRIGGER Graduation_Trigger
```

```
AFTER DELETE ON
   Student2
FOR EACH ROW
BEGIN
   -- Insert into GraduationLog
   INSERT INTO
   GraduationLog
   (StudentID,
   Name,
   GraduationDate)
   VALUES
   (:OLD.StudentID,
   :OLD.FirstName
   || ' '
   ||
   :OLD.LastName, SYSTIMESTAMP);
END;
/

-- Delete students older than 19
DELETE FROM Student2
WHERE Age >= 19;

DECLARE
```

```
    -- Declare variables to hold values from GraduationLog
    v_StudentID Student2.StudentID%TYPE;
    v_Name Student2.FirstName%TYPE;
    v_LName Student2.LastName%TYPE;
BEGIN
    -- Display congratulations message
    DBMS_OUTPUT.PUT_LINE('******CONGRATULATIONS CLASS OF 2024!!!******');
    DBMS_OUTPUT.PUT_LINE('We would Like to Congratulate the Following
Students who are Graduating this year in 2024!!!');


    -- Declare cursor to fetch data from GraduationLog
    FOR grad_rec IN (SELECT StudentID, Name FROM GraduationLog)
    LOOP
        -- Assign values from cursor to variables
        v_StudentID := grad_rec.StudentID;
        v_Name := grad_rec.Name;


        -- Output congratulations message for each student
        DBMS_OUTPUT.PUT_LINE('    ');
        DBMS_OUTPUT.PUT_LINE('| '
            || v_Name
            || ' ');
    END LOOP;


    -- Display message for next year's graduating class
```

```
        DBMS_OUTPUT.PUT_LINE('    ');
    DBMS_OUTPUT.PUT_LINE('We look forward to next year''s graduation for the following students:');
        DBMS_OUTPUT.PUT_LINE('Next year''s graduating class: ');


    -- Declare cursor to fetch data from Student2
    FOR student_rec IN (SELECT FirstName, LastName FROM Student2)
    LOOP
        -- Output message for each student


        DBMS_OUTPUT.PUT_LINE('    ');
        DBMS_OUTPUT.PUT_LINE('| ' || student_rec.FirstName || ' ' || student_rec.LastName);
    END LOOP;
END;
/


-- Display tables
        SELECT * FROM GraduationLog;
    SELECT * FROM Student2;
```

Question #12 Output:

```
Statement processed.
******CONGRATULATIONS CLASS OF 2024!!!******
We would Like to Congratulate the Following Students who are Graduating this year in 2024!!!

| Hermione Granger

| Oliver Wood

| Fred Weasley

| George Weasley

| Helena Ravenclaw

| Vincent Crabbe

| Tom Riddle

| Marcus Flint

We look forward to next year's graduation for the following students:
Next year's graduating class:

| Harry Potter

| Ron Weasley

| Neville Longbottom

| Cedric Diggory

| Constance Pickering

| Luna Lovegood

| Cho Chang

| Draco Malfoy
```

STUDENTID	NAME	GRADUATIONDATE
222222	Hermione Granger	28-APR-24 05.03.04.655137 AM
444444	Oliver Wood	28-APR-24 05.03.04.657265 AM
666666	Fred Weasley	28-APR-24 05.03.04.657334 AM
777777	George Weasley	28-APR-24 05.03.04.657368 AM
700000	Helena Ravenclaw	28-APR-24 05.03.04.657396 AM
500000	Vincent Crabbe	28-APR-24 05.03.04.657422 AM
400000	Tom Riddle	28-APR-24 05.03.04.657445 AM
300000	Marcus Flint	28-APR-24 05.03.04.657476 AM

Download CSV

8 rows selected.

STUDENTID	FIRSTNAME	LASTNAME	GENDER	AGE	HOUSEID	BLOODSTATUS
111111	Harry	Potter	Male	17	100	Half-Blood
333333	Ron	Weasley	Male	18	100	Pure-Blood
555555	Neville	Longbottom	Male	18	100	Pure-Blood
888888	Cedric	Diggory	Male	9	200	Half-Blood
999999	Constance	Pickering	Female	17	200	Half-Blood
900000	Luna	Lovegood	Female	15	300	Pure-Blood
800000	Cho	Chang	Female	17	300	Half-Blood
600000	Draco	Malfoy	Male	18	400	Pure-Blood

Download CSV

8 rows selected.

Question #13 – 77 lines: This code calculates the average grade for each student in a Hogwarts database. It goes through each student's grades, converting them to numerical values and adding them up. Then, it divides the total by the number of grades to get the average, rounding to two decimal places. Finally, it prints out each student's ID, name, and average grade.

```
DECLARE
   v_total_grade_points
   NUMBER := 0;
   v_grade_count
      NUMBER := 0;
   v_average_grade
      NUMBER;
BEGIN
   -- Cursor to fetch student IDs and names
   FOR
   student_rec
   IN
   (SELECT
   StudentID,
   FirstName,
   LastName
   FROM Student)
   LOOP
      -- Reset total grade points and grade count for each student
      v_total_grade_points := 0;
      v_grade_count := 0;
```

```
-- Cursor to fetch grades for each student
FOR
   enrollment_rec
   IN
   (SELECT
   Grade
   FROM
   Enrollment
   WHERE
   StudentID = student_rec.StudentID)
LOOP
   -- Map grade statements to numerical values
   IF enrollment_rec.Grade = 'Outstanding'
   THEN
      v_total_grade_points := v_total_grade_points + 4;
   ELSIF enrollment_rec.Grade = 'Exceeds Expectations'
      THEN
      v_total_grade_points := v_total_grade_points + 3;
   ELSIF enrollment_rec.Grade = 'Acceptable'
      THEN
      v_total_grade_points := v_total_grade_points + 2;
   ELSIF enrollment_rec.Grade = 'Troll'
      THEN
```

```
            v_total_grade_points := v_total_grade_points + 1;
        ELSIF enrollment_rec.Grade = 'Dreadful'
            THEN
            v_total_grade_points := v_total_grade_points + 0;
        END IF;

        -- Increment the count of grades fetched
        v_grade_count := v_grade_count + 1;
    END LOOP;

    -- Calculate the average grade for the current student
    IF v_grade_count > 0
        THEN
        v_average_grade := v_total_grade_points / v_grade_count;
        -- Round the average grade to two decimal places
        v_average_grade := ROUND(v_average_grade, 2);
    ELSE
        v_average_grade := 0; -- Handle division by zero
    END IF;

    -- Display the Student ID, First Name, Last Name, and average grade for the
current student
    DBMS_OUTPUT.PUT_LINE('Student ID: '
        || student_rec.StudentID
        || ', Name: '
```

```
        || student_rec.FirstName
        || ' '
        || student_rec.LastName
        || ', Average Grade: '
        || v_average_grade);
    END LOOP;
END;
/
```

Question #13 Output:

```
Statement processed.
Student ID: 111111, Name: Harry Potter, Average Grade: 3.67
Student ID: 222222, Name: Hermione Granger, Average Grade: 3
Student ID: 333333, Name: Ron Weasley, Average Grade: 2
Student ID: 444444, Name: Oliver Wood, Average Grade: 2.67
Student ID: 555555, Name: Neville Longbottom, Average Grade: 3
Student ID: 666666, Name: Fred Weasley, Average Grade: 2.67
Student ID: 777777, Name: George Weasley, Average Grade: 2.67
Student ID: 888888, Name: Cedric Diggory, Average Grade: 3
Student ID: 999999, Name: Constance Pickering, Average Grade: 2
Student ID: 900000, Name: Luna Lovegood, Average Grade: 2.33
Student ID: 800000, Name: Cho Chang, Average Grade: 2.67
Student ID: 700000, Name: Helena Ravenclaw, Average Grade: 3.33
Student ID: 600000, Name: Draco Malfoy, Average Grade: 3.33
Student ID: 500000, Name: Vincent Crabbe, Average Grade: 2.33
Student ID: 400000, Name: Tom Riddle, Average Grade: 2.67
Student ID: 300000, Name: Marcus Flint, Average Grade: 1.67
```

Question #14 – 89 Lines; This script finds the student(s) with the highest grade among various courses. It starts by setting up variables to hold student details and the highest grade found. Then, it goes through each student's grades, converting them to numbers and updating the highest grade if needed. Finally, it prints out the student(s) with the highest grade along with their details. It's a straightforward process of checking and comparing grades to identify the top performer(s) across courses.

```
DECLARE
   v_student_id Student.StudentID%TYPE;
   v_student_fname Student.FirstName%TYPE;
   v_student_lname Student.LastName%TYPE;
   v_highest_grade NUMBER := 0;
   v_highest_grade_book Books.Title%TYPE;
   v_highest_grade_professor Professor.FirstName%TYPE;
BEGIN
   FOR
   student_rec
   IN
   (SELECT
   s.StudentID,
   s.FirstName,
   s.LastName,
   e.Grade,
   b.Title,
   cr.ProfessorID
                FROM
```

```
Student s
                JOIN
Enrollment e
ON
s.StudentID = e.StudentID
                JOIN
Course cr
ON
e.CourseID = cr.CourseID
                JOIN
Books b
ON
cr.ISBN = b.ISBN)
LOOP
   DECLARE
      v_grade_value NUMBER;
   BEGIN
      CASE
      student_rec.Grade
         WHEN 'Outstanding'
      THEN v_grade_value := 4;
         WHEN 'Exceeds Expectations'
            THEN v_grade_value := 3;
         WHEN 'Acceptable'
```

```
        THEN v_grade_value := 2;
    WHEN 'Troll'
        THEN v_grade_value := 1;
    WHEN 'Dreadful'
        THEN v_grade_value := 0;
    ELSE v_grade_value := -1; -- Handle unexpected grades
END CASE;

IF
    v_grade_value >= 0
    AND v_grade_value > v_highest_grade
    THEN
    v_highest_grade := v_grade_value;
    v_student_id := student_rec.StudentID;
    v_student_fname := student_rec.FirstName;
    v_student_lname := student_rec.LastName;
    v_highest_grade_book := student_rec.Title;
    -- Fetch professor's first name based on ProfessorID
    SELECT
        FirstName
        INTO
        v_highest_grade_professor
    FROM
        Professor
```

```
            WHERE
                ProfessorID = student_rec.ProfessorID;
        END IF;
    END;
END LOOP;

-- Display the information of the student with the highest grade
DBMS_OUTPUT.PUT_LINE('The student(s) with the highest grade is/are:');
DBMS_OUTPUT.PUT_LINE('Student ID: '
    || v_student_id);
DBMS_OUTPUT.PUT_LINE('Name: '
    || v_student_fname
    || ' '
    || v_student_lname);
DBMS_OUTPUT.PUT_LINE('Average Grade: '
    || v_highest_grade);
DBMS_OUTPUT.PUT_LINE('Book: '
    || v_highest_grade_book);
DBMS_OUTPUT.PUT_LINE('Professor: '
    || v_highest_grade_professor);
END;
/
```

Question #14 Output:

```
Statement processed.
The student(s) with the highest grade is/are:
Student ID: 111111
Name: Harry Potter
Average Grade: 4
Book: Advanced Potion-Making
Professor: Severus
```

Question #15: 127 Lines; This code creates a new table Enrollment2 by copying the structure and data from the Enrollment table. It converts the grades in Enrollment2 to numerical values using a CASE statement. Next, it creates a Gradeupdate table to log grade updates. A trigger named Log_Grade_Update is then created to automatically log any updates to grades in Enrollment2. After updating all grades in Enrollment2 by adding 1, the code retrieves each student's name and their highest grade from Gradeupdate, displaying them. Finally, it outputs a message about issuing a curve credit and shows the contents of Enrollment2 and Gradeupdate.

```
Drop Table
   Enrollment2;
Drop Table
   Gradeupdate;

CREATE TABLE
   Enrollment2
   AS SELECT * FROM
   Enrollment;

UPDATE
   Enrollment2
SET
   Grade =
   CASE
   Grade
      WHEN 'Outstanding' THEN 4
```

```
        WHEN 'Exceeds Expectations' THEN 3
        WHEN 'Acceptable' THEN 2
        WHEN 'Troll' THEN 1
        WHEN 'Dreadful' THEN 0
        ELSE NULL -- Handle unexpected values
    END;

CREATE TABLE
    Gradeupdate (
    UpdateID INT GENERATED ALWAYS AS IDENTITY,
    StudentID INT,
    OldGrade VARCHAR(50),
    NewGrade NUMERIC,
    UpdateTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE TRIGGER
    Log_Grade_Update
AFTER UPDATE OF
    Grade
    ON Enrollment2
FOR EACH ROW
BEGIN
    IF
```

```
    :OLD.Grade != :NEW.Grade
    THEN
        INSERT INTO
    Gradeupdate
    (StudentID,
    OldGrade,
    NewGrade)
        VALUES
    (:OLD.StudentID,
    :OLD.Grade,
    :NEW.Grade);
    END IF;
END;
/
UPDATE
    Enrollment2
SET
    Grade = Grade + 1;
DECLARE
    v_student_name VARCHAR2(100);
    v_new_grade NUMBER;
    v_student_max_grade NUMBER;
    CURSOR
        c_student_grades
```

```
        IS
        SELECT
        s.FirstName
        || ' '
        || s.LastName
        AS
        StudentName,
        gu.NewGrade
        FROM
        Gradeupdate gu
        JOIN
        Student s
        ON
        gu.StudentID = s.StudentID
        ORDER BY
        s.FirstName,
        s.LastName,
        gu.NewGrade DESC;
BEGIN
    -- Initialize variables
    v_student_name := NULL;
    v_student_max_grade := NULL;

            -- Display the message about issuing curve credit
```

```
    DBMS_OUTPUT.PUT_LINE('We have issued a Curve credit to all the students to ensure
everyone has a better chance of passing');
    -- Fetch and display each student's highest grade
    FOR
        grade_rec
        IN
        c_student_grades
        LOOP
        IF
        v_student_name
        IS NULL OR
        v_student_name != grade_rec.StudentName
        THEN
            -- Display student's name and highest grade
            IF
        v_student_name
        IS NOT NULL THEN
                DBMS_OUTPUT.PUT_LINE('Student: '
        || v_student_name
        || ', Highest Grade: '
        || v_student_max_grade);
            END IF;
            -- Update student name and highest grade
            v_student_name := grade_rec.StudentName;
            v_student_max_grade := grade_rec.NewGrade;
```

```
        END IF;
    END LOOP;
    -- Display the last student's name and highest grade
    IF v_student_name
        IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Student: '
        || v_student_name
        || ', Highest Grade: '
        || v_student_max_grade);
    END IF;
END;
/
Select * from Enrollment2;
Select * from Gradeupdate;
```

Question 15 Output:

```
Table dropped.

Table dropped.

Table created.

48 row(s) updated.

Table created.

Trigger created.

48 row(s) updated.

Statement processed.
We have issued a Curve credit to all the students to ensure everyone has a better chance of passing
Student: Cedric Diggory, Highest Grade: 5
Student: Cho Chang, Highest Grade: 5
Student: Constance Pickering, Highest Grade: 4
Student: Draco Malfoy, Highest Grade: 5
Student: Fred Weasley, Highest Grade: 5
Student: George Weasley, Highest Grade: 5
Student: Harry Potter, Highest Grade: 5
Student: Helena Ravenclaw, Highest Grade: 5
Student: Hermione Granger, Highest Grade: 5
Student: Luna Lovegood, Highest Grade: 5
Student: Marcus Flint, Highest Grade: 5
Student: Neville Longbottom, Highest Grade: 5
Student: Oliver Wood, Highest Grade: 5
Student: Ron Weasley, Highest Grade: 5
Student: Tom Riddle, Highest Grade: 4
Student: Vincent Crabbe, Highest Grade: 5
```

ENROLLMENTID	STUDENTID	COURSEID	GRADE	ENROLLMENTPERIOD
1	111111	Pot104	5	Spring
2	111111	Tran115	4	Spring
3	111111	DEF101	5	Spring
4	222222	Pot104	5	Spring
5	222222	DEF101	3	Spring
6	222222	H808	4	Spring
7	333333	CHA258	3	Spring