**PLSQL Assignments**

Consider the below mentioned table:

1. Department

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| DeptNo | Integer |  | Primary Key |
| DeptName | Varchar | 21 | Not Null |
| DeptLocation | Varchar | 13 | Not Null |

1. Employee

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| EmpNo | Integer |  | Primary Key |
| Ename | Varchar | 30 | Not Null |
| Job | Varchar | 12 |  |
| Mgr | Integer |  | References Employee(EmpNo) |
| Hiredate | Date |  |  |
| Salary | Number | 7,2 | Check (Salary > 0) |
| Comm | Number | 7,2 |  |
| Deptno | Integer |  | References Department(DeptNo) |
| Gender | Varchar | 12 | Check (Gender=’Male’ or Gender=’Female’ |

1. Student

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| StudentId | Varchar | 20 | Primary Key |
| FirstName | Varchar | 21 | Not Null |
| LastName | Varchar | 21 | Not Null |
| Date\_Of\_Birth | Date |  |  |
| Address | Varchar | 300 |  |
| City | Varchar | 20 | Not Null |
| State | Varchar | 12 | Not Null |
| ZipCode | Varchar | 9 | Not Null |
| Telephone | Varchar | 10 | Default ‘NA’ |
| Fax | Varchar | 10 | Default ‘NA’ |
| Email | Varchar | 100 | Not Null |

1. Instructor

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| InstructorId | Varchar | 20 | Primary Key |
| DeptNo | Integer |  | References Department(DeptNo) |
| FirstName | Varchar | 21 | Not Null |
| LastName | Varchar | 21 | Not Null |
| Telephone | Varchar | 10 | Default ‘NA’ |
| Fax | Varchar | 10 | Default ‘NA’ |
| Email | Varchar | 100 | Not Null |

1. Course

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| CourseId | Varchar | 5 |  |
| DeptNo | Integer |  | References Department(DeptNo) |
| Title | Varchar | 75 | Not Null |
| Description | Varchar | 200 | Not Null |
| Additional\_Fees | Number | 9,2 | Check(Additional\_Fees >0) |
|  |  |  | PrimaryKey(CourseId,DeptNo) |

1. Schedule\_Type

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| ScheduleId | Varchar | 20 | Primary Key |
| Description | Varchar | 200 | Not Null |
| Day | Number |  | Check (Day in (1,2,3,4,5,6,7) |
| StartingTime | Date |  | Default Sysdate |
| Duration | Integer |  | Not Null |

1. ClassLocation

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| ClassBuilding | Varchar | 25 | Not Null |
| ClassRoom | Varchar | 25 | Not Null |
| SeatingCapacity | Varchar | 2 | Default 100 |

1. Class

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| ClassId | Varchar | 20 | Primary Key |
| ScheduleId | Varchar | 20 | References Schedule\_Type(ScheduleId) |
| ClassBuilding | Varchar | 25 | Not Null |
| ClassRoom | Varchar | 25 | Not Null |
| DeptNo | Integer |  | References Department(DeptNo) |
| InstructorId | Varchar | 20 | References Instructor(InstructorId) |
| Semester | Varchar | 6 | Default ‘I’ |
| Year | Date |  | Default sysdate |

1. StudentSchedule

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Width | Constraints |
| StudentId | Varchar | 20 |  |
| ClassId | Varchar | 20 |  |
| Grade | Varchar | 2 | Check (Grade in (‘A’,’A+’,’B’,’B+’,’C’,’C+’) |
| GradeAssigned | Date |  | Default Sysdate |

Use the above-mentioned table structure to do the below mentioned assignments:

1. Write a PL/SQL block to insert the student details into the Student table until the user wishes to stop.
2. Write a PL/SQL block to display the numbers from 1to 50 in words.
3. Write a PL/SQL block to display the name of the employees whose salary is > 3000.
4. Write a PL/SQL block to accept the Department Number from the user, check for the existence in Department table, if exist display the department details for the specified department number else display appropriate error message.
5. Write a PL/SQL block to accept the CourseId from the user, check for the existence in Course table, if exist remove the records if the user wishes to delete else display appropriate error message.
6. Write a PL/SQL block to display ClassId, Grade and GradeAssigned details for the specified StudentId
7. Write a PL/SQL block to increase the salary of the employee by 15%, if their salary is > 15000
8. Write a PL/SQL block to insert the records recursively into the Instructor table until the user wishes to terminate. Provided if the recordcount is > 20.
9. Write a PL/SQL block to display the count of employees whose gender is “Male”.
10. Write a PL/SQL block to display the class location details whose seating capacity is > 200.
11. Write a PL/SQL block to insert only the odd numbers from 1 to 20 into Department table as Deptno and get the Department Name from the user.
12. Write a PL/SQL block to count the number of courses available in Course table.

**Cursors**

1. Write a PL/SQL block to display the total salary (ie., Salary + Comm) of each employee whose comm is not null.
2. Write a PL/SQL block to display the employee details, and the number of employees joined in each month.
3. Write a PL/SQL block to increase the additional fees by 10% and if the additional fees exceeds 100 then decrease the additional fees by 20%.
4. Write a PL/SQL block which displays the schedule type for a specified classroom entered by the user.
5. Write a PL/SQL block to display the employee details whose gender is Male.
6. Write a PL/SQL block to count the number of students available in each city.
7. Write a PL/SQL block to display the class location details whose seating capacity is >= 200.
8. Write a PL/SQL block to display all the students whose grade = ‘A’.
9. Write a PL/SQL block to display the details of the instructor who are handling for 1st semester.
10. Write a PL/SQL block to display all the records available from the table named StudentSchedule.

**Exceptions**

1. Write a PL/SQL block to handle the exception named “DUP\_VAL\_ON\_INDEX by inserting a duplicate row in the course table.
2. Write a PL/SQL block to handle the exception named “VALUE\_ERROR” by inserting the value for DeptName column of width greater than 21 into the department table.
3. Write a PL/SQL block to handle the exceptions which are not handled by using OTHERS and display a message.

**Sub-Programs and Packages**

1. Create a procedure that takes an argument(Description) and deletes the row from the table named “Course”.
2. Write a function that takes two arguments viz. studentid, classid from the user and checks whether any conflict occurs with any other class with the current class schedule. If occurs give an alert message that a conflict occurs with the corresponding class else display no conflict.
3. Write a function that will get the value from the user and insert rows into the class table without violating integrity constraints.
4. Write a procedure that will display the users to which the specified table is granted.
5. Create a package that contains overloaded functions for:
   1. Adding five integers.
   2. Subtracting two integers.
   3. Multiplying three integers.
6. Create a package that contains the following:
   1. Function to register for a class with arguments (student\_id,class\_id) and check if the student has already registered or not.
   2. Function to check for conflicts in classes with arguments (StudentId, ClassId).
   3. Procedure to assign an instructor for a class with arguments (ClassId, InstructorId) and check if the instructor is associated with the department that offers the class.
   4. Procedure to assign grade with arguments (StudentId, ClassId, Grade) and check if the students are registered for the class.
   5. Function to calculate the average grade point for the given student with arguments (StudentId).

**Triggers**

1. Write a database trigger before insert for each row on the course table not allowing transactions on Sundays and Saturdays.
2. Write a database trigger after update for each row giving the date and the day on which the update is performed on the class table.
3. Write a database trigger before delete for each row not allowing deletion and giving message on the department table.
4. Write a database trigger before insert/delete/update for each row not allowing any of these operations on the table student on Monday, Wednesday and Sunday.
5. Write a database trigger before insert / update for each row not allowing any of these operations on the table Instructor between 6 PM to 10 AM.