# Midterm Project
# Part I
# Instructions for Students

Make sure you save all your work in files, so you can check it later and also re-use the work done here later in the next MidTerm Project.

## Project Setup: The Data

This project will use a case study called STUDENT ADMINISTRATION or SA. A set of database tables is used to manage a school's course offerings as delivered by instructors in many classes over time. Information is stored about classes that are offered, the students who take classes, and the grades the students receive on various assessments. The school administrators can use the SA database to manage the class offerings and to assign instructors. Teachers can also use the SA database to track student performance.

The database objects for this project are already in your accounts and they are as follows:

Tables:

INSTRUCTORS
SECTIONS
COURSES
CLASSES
ASSESSMENTS
STUDENTS
ENROLLMENTS
CLASS_ASSESSMENTS
ERROR_LOG
GRADE_CHANGES

Sequence:

ASSESSMENT_ID_SEQ

Synonyms:

sect FOR sections
instr FOR instructors
enroll FOR enrollments
stu FOR students
cl_assess FOR class_assessments
cl FOR classes
cour FOR courses
assess FOR assessment.

Copyright © 2009, Oracle. All rights reserved.

## Part 1: Student Information

1. Create an anonymous PL/SQL block to enroll a student in a particular class. Make sure the students save it in a file called *enroll_student_in_class.sql* . You will use the ENROLLMENTS table. Accept a STU_ID and CLASS_ID as input parameters. Use "today's date" for the ENROLLMENT_DATE and the string 'Enrolled' for the STATUS. Hint: To get your program to prompt for the values for STU_ID and CLASS_ID do the following:

   ```
   DECLARE
   v_stu_id         enrollments.stu_id%TYPE := :student_id;
   v_class_id       enrollments.class_id%TYPE) := :class_id;
   .....
   ```

2. Create an anonymous block to drop a student from a class. Save the block in a file called *drop_student_from_class.sql.* You will use the ENROLLMENTS table. Accept a STU_ID and CLASS_ID as input parameters.

3. Create an anonymous block that displays all of the classes a student has been enrolled in within the most recent 6 years. Save the block in a file *student_class_list* You will use the ENROLLMENTS table. For example: If you run your program on May 10, 2006, you should display all enrollments between May 10, 2000 and May 10, 2006. Accept the STU_ID as an input parameter. For each enrollment, display the ENROLLMENT_DATE, CLASSS_ID and STATUS.

4. Create an anonymous block to add "n" new classes for a particular course. Save the block in a file called *add_new_classes.sql* Accept the following IN parameter:
   - Number of new classes required. Set a default value of 1.
   - Course id; For each new class, use "today"as the START_DATE.
   - Period, to specify what days the class meets.
   - Frequency, to specify how often it meets.
   - Intsructor id, who is teaching the class(s).

   There are 2 ways you can generate a new CLASS_ID.
   First use a SELECT to find out what is currently the highest CLASS_ID. Increment this number by 1 for each new class you add. For example: When you run your program, the highest CLASS_ID is 12 and you want to create 3 new classes. Your new CLASS_IDs would be 13, 14, and 15.
   Create a sequence number that starts with a number higher than the maximum CLASS_ID. Use sequence_name.NEXTVAL in your INSERT statement.

   Test your program by adding 4 new classes for COURSE_ID #1001. Also test your default by calling the program to add only one class for COURSE_ID #1002.

Copyright © 2009, Oracle. All rights reserved.

## Part 2: Teacher Tools

1. Create an anonymous block that a teacher can run to see the students in a course across all classes of that course. Save the block in a file called *course_roster.sql.* Accept the INSTR_ID and COURSE_ID. For each ENROLLMENT, display: CLASS_ID, STATUS, Student FIRST_NAME and LAST_NAME.

2. Create an anonymous block which will convert a number grade to a letter grade. Save your work in a file called *convert_grade.sql.* Prompt for a number grade. RETURN a CHAR value. Use the following rules: A:90 or above, B: >=80 and<90 , C: >=70 and < 80, D: >=60 and < 70, F:<60.

3. Create an anonymous block that will RETURN the number of students in a particular class. Save your work in a file called *student_count.sql.* Accept a CLASS_ID as a parameter.

4. Create an anonymous block which a teacher can run to insert a new assignment (ASSESSMENT) including a DESCRIPTION. Save the program in a file called *create_assignment.sql.* Use the ASSESSMENT_ID_SEQ sequence to generate the class_assessment_id.

5. Create an anonymous block that a teacher can run to insert the student's grade on a particular assignment. Save the program in a file called *enter_student_grade.sql.* Accept a NUMERIC_GRADE, CLASS_ASSESSMENT_ID, CLASS_ID and STU_ID. Use "today's" date for the DATE_TURNED_IN.

Copyright © 2009, Oracle. All rights reserved.

## Part 3: School Administrator's Tools

1. Create an anonymous block to list any enrollments which have NEITHER a FINAL_NUMERIC_GRADE  or FINAL_LETTER_GRADE. Save the program in a file called *show_missing_grades.sql.* Accept a start_date and end_date to establish a date range. Display only enrollments between those 2 dates. Write your program so the start_date and end_date are optional. If both dates are not entered, display all applicable enrollments for the past year, and include a note about the date range.  For each enrollment, list the CLASS_ID, STU_ID, and STATUS. Order the output by ENROLLMENT_DATE with the most recent enrollments first.

2. Create an anonymous block to find the average grade for a class. Save the program in a file called *compute_average_grade* .sql. Assume the class has used numeric_grades. Accept a CLASS_ID.  Return the average grade.

3. Create an anonymous block to return the number of classes offered for a given course. Save the program in a file called *count_classes_per_course.sql.* Have the program prompt for a course ID.

4. Create an anonymous block to list all classes offered  between a range of dates. Save your program in a file called *show_class_offerings.sql.* Accept a start date and end date. For each class found, display the CLASS_ID, START_DATE, instructor FIRST_NAME and LAST_NAME, course TITLE and SECTION_CODE.

Copyright © 2009, Oracle.  All rights reserved.