**COMP 3005B**  
**Assignment #4**  
**Due: Nov. 13 @11:59PM**

**Instruction**

1. You should do the assignment independently. If copying is found, the case will be reported to the office of the Dean of Science immediately.
2. The assignment must be typed, completed on an individual basis, and submitted as a single Word/PDF file with your name as the filename to **brightspace**.
3. Last in the Supplier table is your last name. If your information is not shown correctly in the result, you will get a 0 mark for the assignment.
4. You should directly do your assignment on this document and name the document with your last name followed by your first name so that it is easy for TAs to mark.
5. For Part 2, you need to use [Oracle VM](https://git.scs.carleton.ca/downloads/CourseVirtualMachines/2022F-2023W/COMP3005-W23.ova) and SQLPLUS interface to Oracle DBMS, test each program carefully, and submit the final version of the program together with several representative screenshots of the execution of the program. If there is no screenshot, you will get 0 for the question.

**Part 1 Concepts (20 marks)**

Explain the following concepts based on the information in the lecture notes.

1. View

A derived table defined using a query. (not a stored table).

1. Impedance Mismatch

Two different ways to process data:   
A set of tuples at a time in RDB, a tuple at a time in PL/SQL

1. Execute immediate in PLSQL  
   A way to run SQL DDL and DML statements in PLSQL, cannot use for QL.
2. Cursor in PLSQL

A way to overcome impedance mismatch by generating a query result and using the cursor name as a pointer to go over the results one by one.

1. Update Cursor

Unlike normal cursors that are used to query data, update cursors are used to update tuples or attributes of tuples found by the query by putting a lock to previous concurrent access/updates.

1. Parameterized Cursor in PLSQL

A cursor is defined with a parameter that is replaced by a value when it is used.

1. Physical Model

Used to specify how data is stored in the database: heap, indexed, external file.

1. ER Model

A conceptual model used to represent real-world entity types and their various relationships.

1. EER Model

Extended ER model that can represent **subtypes of entities and their various relationships.**

1. Aggregation

A way to treat a relationship as an entity type to participate in a relationship.

**Part 2 PL/SQL (40 Marks)**

This part is based on the Suppliers-Parts database which has three tables shown below.

1. Delete all three tables if you created before and then use **execute immediate** statement to write a PL/SQL program to create and populate the three tables (10 marks)

begin

execute immediate 'create table Suppliers (

S# varchar2(2), name varchar2(10), status real, city varchar2(10))';

execute immediate 'insert into Suppliers values (''S1'', ''Smith'', 20, ''London'')';

execute immediate 'insert into Suppliers values (''S2'', ''Jones'', 30, ''Paris'')';

execute immediate 'insert into Suppliers values (''S3'', ''Blake'', 30, ''Paris'')';

execute immediate 'insert into Suppliers values (''S4'', ''Last'', 20, ''London'')';

execute immediate 'insert into Suppliers values (''S5'', ''Adams'', 30, ''Athens'')';

execute immediate 'create table Parts (

P# varchar2(2), name varchar2(10), color varchar2(10), weight real, city varchar2(10))';

execute immediate 'insert into Parts values (''P1'', ''Nut'', ''Red'', 12.0, ''London'')';

execute immediate 'insert into Parts values (''P2'', ''Bolt'', ''Green'', 17.0, ''Paris'')';

execute immediate 'insert into Parts values (''P3'', ''Screw'', ''Blue'', 17.0, ''Oslo'')';

execute immediate 'insert into Parts values (''P4'', ''Screw'', ''Red'', 14.0, ''London'')';

execute immediate 'insert into Parts values (''P5'', ''Cam'', ''Blue'', 12.0, ''Paris'')';

execute immediate 'insert into Parts values (''P6'', ''Cog'', ''Red'', 19.0, ''London'')';

execute immediate 'create table SP (S# varchar2(2), P# varchar2(2), qty real)';

execute immediate 'insert into SP values (''S1'', ''P1'', 300)';

execute immediate 'insert into SP values (''S1'', ''P2'', 200)';

execute immediate 'insert into SP values (''S1'', ''P3'', 400)';

execute immediate 'insert into SP values (''S1'', ''P4'', 200)';

execute immediate 'insert into SP values (''S1'', ''P5'', 100)';

execute immediate 'insert into SP values (''S1'', ''P6'', 100)';

execute immediate 'insert into SP values (''S2'', ''P1'', 300)';

execute immediate 'insert into SP values (''S2'', ''P2'', 400)';

execute immediate 'insert into SP values (''S3'', ''P2'', 200)';

execute immediate 'insert into SP values (''S4'', ''P2'', 200)';

execute immediate 'insert into SP values (''S4'', ''P3'', 300)';

execute immediate 'insert into SP values (''S4'', ''P4'', 400)';

execute immediate 'insert into SP values (''S4'', ''P5'', 500)';

execute immediate 'insert into SP values (''S4'', ''P6'', 600)';

end;

/

1. Write a PL/SQL program to list all supplier rows, in supplier number order so that each supplier row is immediately followed in the listing by all bank rows for parts that the supplier supplies in part number order. Supplies that do not supply parts should still be listed (15 marks)

begin

for S in (select \* from Suppliers order by S# ASC) loop

DBMS\_OUTPUT.PUT\_LINE(S.S#||' '||S.NAME||' '||S.STATUS||' '||S.CITY);

for P in (select P.P#, P.NAME, P.COLOR, P.WEIGHT, P.CITY from Parts P, SP T

where S.S# = T.S# AND T.P# = P.P#) loop

DBMS\_OUTPUT.PUT\_LINE(P.P#||' '||P.NAME||' '||P.COLOR||' '||P.WEIGHT||' '||P.CITY);

end loop;

end loop;

end;

/

1. Redo question 2 using a parameterized cursor that takes a supplier name. It should first prompt the user to enter a supplier name and then display the same information as in 2 just for the given customer. Use your own last name in the table to test this program. (15 marks)

declare

SNO Suppliers.S#%TYPE; NAME Suppliers.NAME%TYPE;

STATUS Suppliers.STATUS%TYPE;

CITY Suppliers.CITY%TYPE;

cursor C (input Suppliers.NAME%TYPE) is   
 select \* from Suppliers where input = Suppliers.NAME;

begin

open C('&NAME');

loop

fetch C into SNO, NAME, STATUS, CITY; EXIT when C%notfound;

DBMS\_OUTPUT.PUT\_LINE(SNO||' '||NAME||' '||STATUS||' '||CITY);

for P in (

select P.P#, P.NAME, P.COLOR, P.WEIGHT, P.CITY from Parts P, SP T

where T.S# = SNO AND T.P# = P.P# ORDER BY P.P#)

loop DBMS\_OUTPUT.PUT\_LINE(P.P#||' '||P.NAME||' '||P.COLOR||'

'||P.WEIGHT||' '||P.CITY); END LOOP;

end loop;

close C;

end;

/

**SP**

|  |  |  |
| --- | --- | --- |
| **S#** | **P#** | **QTY** |
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S1 | P3 | 400 |
| S1 | P4 | 200 |
| S1 | P5 | 100 |
| S1 | P6 | 100 |
| S2 | P1 | 300 |
| S2 | P2 | 400 |
| S3 | P2 | 200 |
| S4 | P2 | 200 |
| S4 | P3 | 300 |
| S4 | P4 | 400 |
| S4 | P5 | 500 |
| S4 | P6 | 600 |

**Suppliers**

|  |  |  |  |
| --- | --- | --- | --- |
| **S#** | **NAME** | **STATUS** | **CITY** |
| S1 | Smith | 20 | London |
| S2 | Jones | 30 | Paris |
| S3 | Blake | 30 | Paris |
| S4 | Last | 20 | London |
| S5 | Adams | 30 | Athens |

**Parts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P#** | **NAME** | **COLOR** | **WEIGHT** | **CITY** |
| P1 | Nut | Red | 12.0 | London |
| P2 | Bolt | Green | 17.0 | Paris |
| P3 | Screw | Blue | 17.0 | Oslo |
| P4 | Screw | Red | 14.0 | London |
| P5 | Cam | Blue | 12.0 | Paris |
| P6 | Cog | Red | 19.0 | London |

**Part 3 ER Model (40 marks)**

A university information system involves buildings, classrooms, offices, departments, courses, sections, timeslots, chairs, instructors, and students.

1. A building has a unique building number such as HP, a unique name, and a number of classrooms and offices.
2. A classroom has a room number such as 5125 that is unique in the building, the number of seats, and is either empty or used by a number of sections at different days and times.
3. An office has a room number that is unique in the building, the size in square feet, and is either empty or occupied by a chair or up to 4 instructors.
4. A department has a unique dept code such as COMP, a unique name, 1 chair, 1 to 40 instructors, 1 to 1000 students, 1 to 20 courses, 1 to 30 offices in the same or different buildings, and no offices are shared by different departments.
5. A course has a unique course number such as 3005 and a name such as Databases that are unique in the department that offers the course, credit hours, and a number of prerequisite courses. Courses are offered as sections and not all courses are offered.
6. A section has a unique section code A, B, C, D, or E *within* the course, semester, year, classroom, timeslot, and textbooks, and is related to one course, one instructor, and 5 to 200 students. Just consider current sections only.
7. A timeslot has a timeslot id, day, start time, and end time.
8. A chair or an instructor has a unique employee number, a name, an office, 0 to 2 phone numbers, and can only work in one department. Note that a chair is not an instructor, and vice versa. An instructor teaches 1-5 sections.
9. A student has a unique student number, a name, majors in one department, takes 1 to 5 sections, and has a grade for each section.
10. Draw the ER diagram for this information system that can represent the constraints specified above. You can use free **draw.io** to do this part.

