Tshwane University of Technology

We empower people

YEAR: 2012
EXAMINATION: December Supplementary

SUBJECT NAME:	DEVELOPMENT SOFTWARE 2B /SOFTWARE SKILLS 2 B
SUBJECT CODE:	DSO23BT/ SFW20BT
QUALIFICATION(S):	NATIONAL DIPLOMA: INFORMATION TECHNOLOGY NATIONAL DIPLOMA: FINANCIAL INFORMATION SYSTEMS

PAPER DESCRIPTION: Closed Book **DURATION:** 4 Hrs **PAPER:** Only

SPECIAL REQUIREMENTS

☒ **NONE**
□ **NON-PROGRAMMABLE POCKET CALCULATOR**
□ **SCIENTIFIC CALCULATOR**
□ **COMPUTER ANSWER SHEET**
□ **GRAPH PAPER**
□ **DRAWING INSTRUMENTS**

OTHER:

INSTRUCTIONS TO CANDIDATES: Answer all questions

Ensure that your answers are saved in the specified location.
Save regularly.

TOTAL NUMBER OF PAGES INCLUDING COVER PAGE: 8
TOTAL NUMBER OF ANNEXURES: 0

EXAMINER:	GO Leroke SK Mogapi	**FULL MARKS:**	100
MODERATOR:	SM Marebane	**TOTAL MARKS:**	100

EXAMINATION RULES

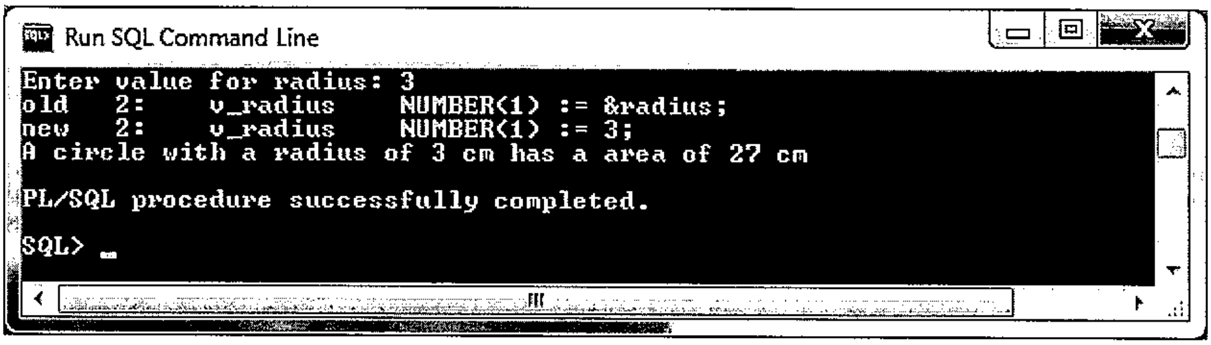
INSTRUCTIONS TO CANDIDATES

1. By writing your name and student number on this script you confirm that you are familiar with the examination rules and regulations of TUT.
2. Write your full first names and surname and student number, which appears on your student card, clearly and correct in the space provided on the test paper. Your student number must also be written in the right hand corner of every loose sheet of paper.
3. During practical tests the following must be written on the label of the disk: Student number, Surname, Initial, Program and Lecturer.
4. When answering Theory tests you must write neatly and clearly and answer in the spaces provided for it. If you answer in a separate book use both sides of the paper. Leave margins entirely free for use of examiner.
5. You need not commence with every new answer on a new page. After completing a question, draw a line across the page and start the new question. It must be clearly numbered. Group the answers to subsections of a question together.
6. During practical tests you are not allowed to have any application open other than the application on which the test is based.
7. While the test is in progress, you may not help or try to help another student, obtain assistance or try to obtain assistance, or communicate or try to communicate with anyone.
8. You may not have in your possession any book, memorandum, note(s), sketch, map, film or any other document (including unused paper) or any other aid with a bearing on the subject, with the exception of whatever is handed out to you in the test hall.
9. No cell phone may be in your possession or be used by you during the test.
10. You may not use any pocket calculator unless it is clearly authorized on test paper.
11. You will render yourself liable to disqualification if you make personal remarks to the examiner or invigilator. The writing or drawing of any offensive matter on the test material supplied to you will disqualify you.
12. No explanation of test questions may be asked or will be given
13. Your answers may be written in English or Afrikaans.
14. All work done must be submitted. If you wish any mark not to be marked, draw a line through it.
15. No pages may be removed or inserted to your script.
16. You must leave the test hall as soon as you submitted your work, but not before at least 30 minutes of the test time has expired. You are not allowed to leave the hall without the permission if the invigilator. After any test, no student is allowed to open the computer to access any other program than the test.
17. All test scripts; disks and aids handed out to you must be returned before you leave the test hall.
18. The invigilator may disqualify any candidate for unfair practices or unsatisfactory conduct.

A student who does not comply with these rules shall be subject to disciplinary steps in terms of the Examination rules and Regulations of TUT.

Question 1 [6]

Create a PL/SQL program that calculates the area of a circle. The program must prompt for the radius and initialize the constant value of pi which is 3,14. Area is the equivalent of radius squared multiplied by pi.

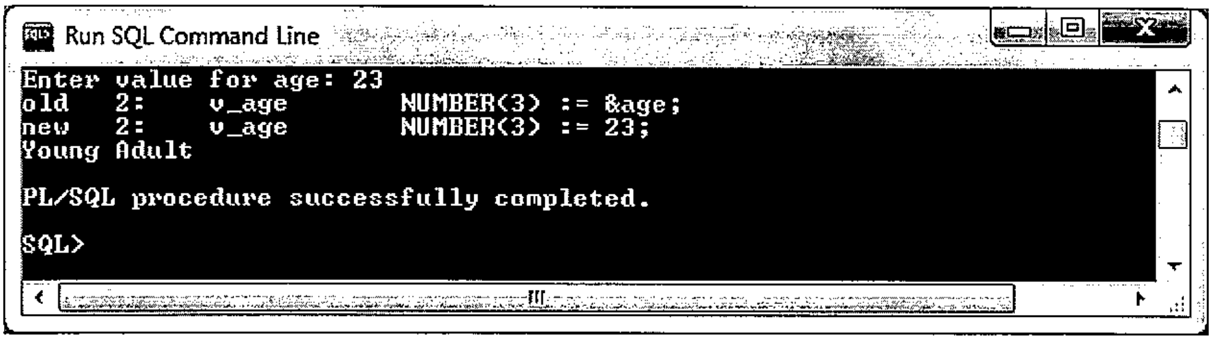


```
Enter value for radius: 3
old   2:     v_radius    NUMBER(1) := &radius;
new   2:     v_radius    NUMBER(1) := 3;
A circle with a radius of 3 cm has a area of 27 cm

PL/SQL procedure successfully completed.

SQL> _
```

Question 2 [9]

Create a PL/SQL block each will prompt a user for their age. Use a CASE Expression to determine the group to allocate according to the table below:

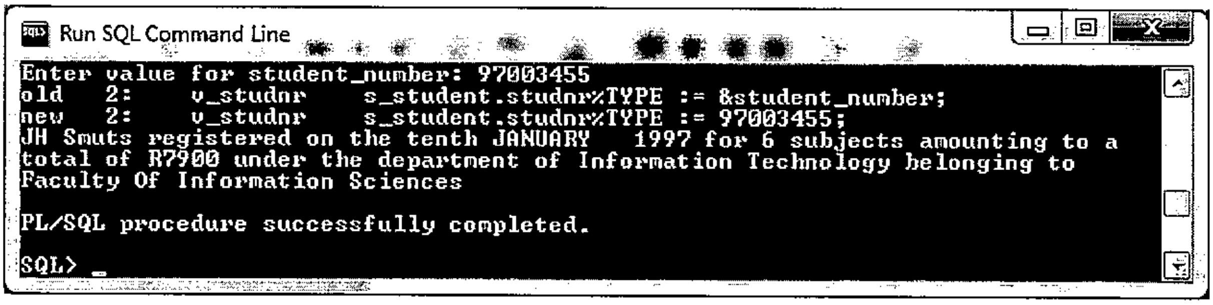
Age	Group
0 – 2	Infant
3 – 12	Child
13 – 19	Teen
20 – 30	Young Adult
30 – 60	Adult
60 +	Pensioner



```
Enter value for age: 23
old   2:     v_age       NUMBER(3) := &age;
new   2:     v_age       NUMBER(3) := 23;
Young Adult

PL/SQL procedure successfully completed.

SQL>
```

Question 3 [15]

Write a PL/SQL block that will prompt the user for the student number. Use % TYPE for all the variables that relate to a table. The output should be as the output below. Follow the output to determine variables needed.



```
Enter value for student_number: 97003455
old   2:     v_studnr     s_student.studnr%TYPE := &student_number;
new   2:     v_studnr     s_student.studnr%TYPE := 97003455;
JH Smuts registered on the tenth JANUARY   1997 for 6 subjects amounting to a
total of R7900 under the department of Information Technology belonging to
Faculty Of Information Sciences

PL/SQL procedure successfully completed.

SQL> _
```

Question 4 [15]

Create a lecture record that will stored lecture details. Lecture details for this Record contains lecturer surname, initials, office number, total number of subjects the lecture is assigned and the total salary of the lecture. The lecturer's total is the sum of salary and bonus, if the bonus of the lecturer is NULL then the bonus should be R500.00, this should be done using NVL function.

```
Enter the value for lecturer_number: 513500

Oliver G in office 213 lecturing 2 subjects earning R78,500.0
```

Question 5 [18]

The faculty of ICT advised TUT's management to add a levy fee on all subjects that has a word Systems in each subject. The levy fee is calculated 25% of that subject fee.

Create a PL/SQL block that declares a **cursor** named **SUBJECT_LEVY_CUR** and define a **record** based on that cursor with the %ROWTYPE attribute. The cursor must retrieve the Subject code, name, fee, and the 25% of levy fee, name 25% the levy fee as "**LEVY FEE**". Use an explicit cursor to FETCH values INTO a record from the **s_subject** table.

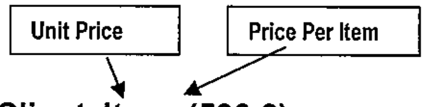
```
Subjects With Levies

*********************************************

INFORMATION SYSTEMS 1[IS1] fee is R1200 and its levy is R300

INFORMATION SYSTEMS 2[IS2] fee is R1300 and its levy is R325

INFORMATION SYSTEMS 3[IS3] fee is R1400 and its levy is R350

SYSTEMS PROGRAMMING 1[SP1] fee is R1400 and its levy is R350

MANAGEMENT INFORMATION SYSTEMS 3[MI3] fee is R1900 and its levy is R475

PRICIPLES OF INFORMATION SYSTEMS [POI] fee is R1200 and its levy is R300


PL/SQL procedure successfully completed.
```

Question 6 [14]

A client bought a number of items at a certain unit price per item. If the client buys more than 10 items, 5% discount is granted, but if more than 15 items are bought, 10% discount is granted.

Create a server side procedure **Item_Price** that accepts the unit price and the number of items bought supplied as input parameters. The procedure must calculate the discount, VAT, and amount payable.

Use the following statements and test data to test your procedure:



SQL> Execute Client_Items(500,2)

SQL> Execute **Client_Items(500,2);** You bought 2 items at R500 Discount: R0 Vat: R140 Final Amount Due: R1140	**SQL> Execute** **Client_Items(500,10)**; You bought 10 items at R500 Discount : R0 Vat: R700 Final Amount Due: R5700
SQL> Execute **Client_Items(500,14);** You bought 14 items at R500 Discount: R350 Vat: R931 Final Amount Due: R7581	**SQL> Execute** **Client_Items(500,20);** You bought 20 items at R500 Discount : R1000 Vat: R1260 Final Amount Due: R10260

Question 7 **[23]**

Students at TUT may not register more than twice for the same subject.
You have to handle this business rule by using a combination of a stored procedure (for inserting a registration record into the REGISTRATION table) and a cursor (check_cursor) to check and display the number of previous registrations for the student in the REGSTRATION table.

Create a server side procedure **REGISTER** that accepts two parameters: (Student number and Subject code). You should also declare a parameterized cursor **CHECK_CURSOR** to determine how many times a given student has registered for a specific subject and to display these registration details (Refer to the output). The parameters for the cursor will be the **student number** and **subject code**. Make use of a **cursor FOR loop** to process the result-set returned by the cursor.
Declare a user-defined exception **too_many_regs** which will be raised when there are 2 or more previous registrations for a given subject and student number.

The following business rules need to be applied:

a) If the student has not registered for the applicable subject before or the student has registered for it only once before, allow him/her to register for that subject. Use today's date for the REG_DATE field and the value 'P' for the CAMPUS field. Display a suitable message to notify the user that the registration was successful.

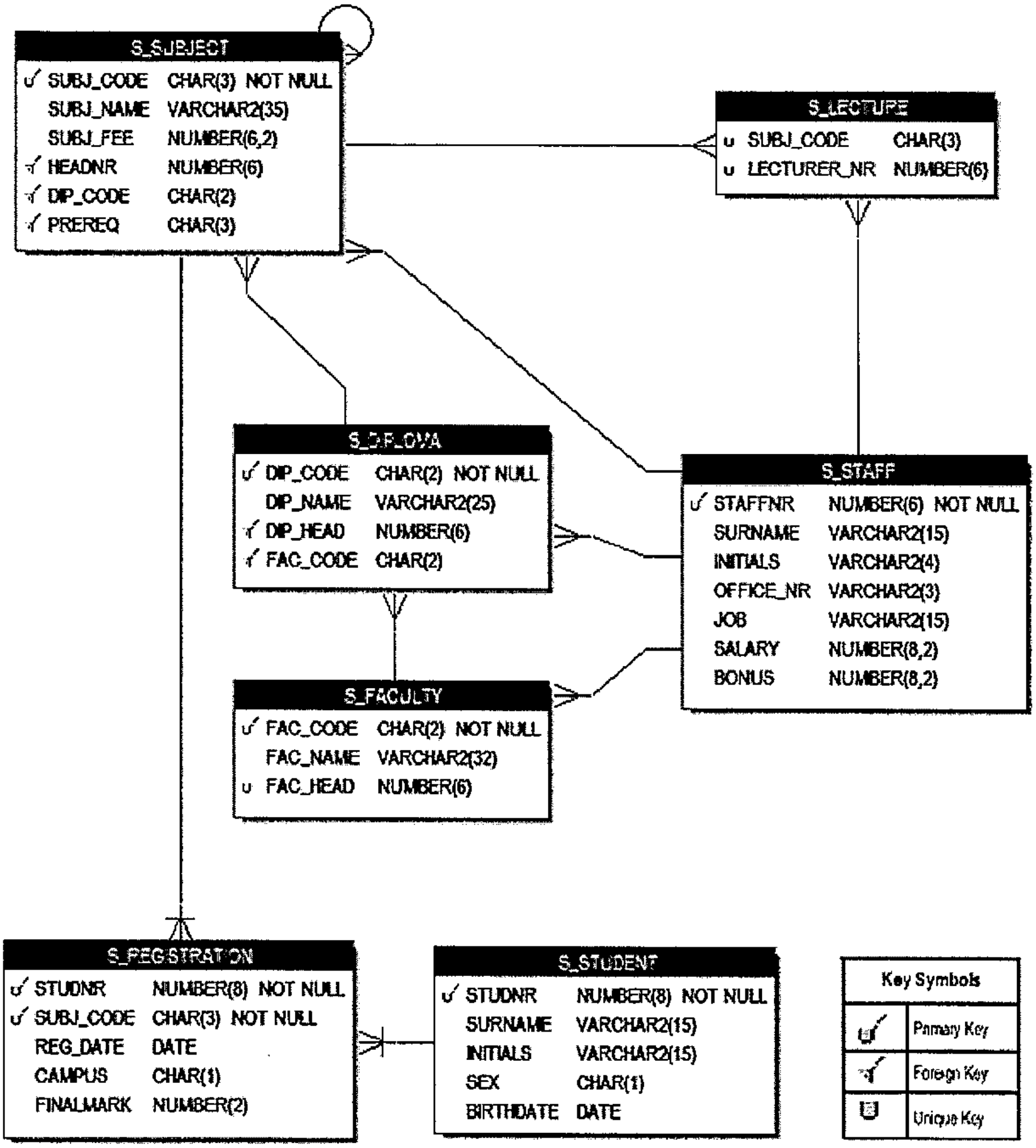
b) If the student has registered for the applicable subject twice already, and he/she attempts to register for it again, use the exception to display a suitable message informing the user that the student cannot register again for that subject. Also display the previous registration dates for the subject.

c) You may assume that the procedure will always be tested using valid student numbers and subject codes only. Hence, you don't have to do any further validations.

d) Include the statements used to test a registration for the students and subjects as indicated in the test data in your script file.

a).	b).
STUDENT :94000001 successfully registered for subject: CM2 PL/SQL procedure successfully completed.	Registration date: 17/JAN/97 Registration date: 17/JAN/98 STUDENT: 94000001 registered for subject: CM1 twice already - registration not allowed!

Student Database



	S_SUBJECT		
u✓	SUBJ_CODE	CHAR(3)	NOT NULL
	SUBJ_NAME	VARCHAR2(35)	
	SUBJ_FEE	NUMBER(6,2)	
✓	HEADNR	NUMBER(6)	
✓	DIP_CODE	CHAR(2)	
✓	PREREQ	CHAR(3)	

	S_LECTURE	
u	SUBJ_CODE	CHAR(3)
u	LECTURER_NR	NUMBER(6)

	S_DIPLOMA		
u✓	DIP_CODE	CHAR(2)	NOT NULL
	DIP_NAME	VARCHAR2(25)	
✓	DIP_HEAD	NUMBER(6)	
✓	FAC_CODE	CHAR(2)	

	S_STAFF		
u✓	STAFFNR	NUMBER(6)	NOT NULL
	SURNAME	VARCHAR2(15)	
	INITIALS	VARCHAR2(4)	
	OFFICE_NR	VARCHAR2(3)	
	JOB	VARCHAR2(15)	
	SALARY	NUMBER(8,2)	
	BONUS	NUMBER(8,2)	

	S_FACULTY		
u✓	FAC_CODE	CHAR(2)	NOT NULL
	FAC_NAME	VARCHAR2(32)	
u	FAC_HEAD	NUMBER(6)	

	S_REGISTRATION		
u✓	STUDNR	NUMBER(8)	NOT NULL
u✓	SUBJ_CODE	CHAR(3)	NOT NULL
	REG_DATE	DATE	
	CAMPUS	CHAR(1)	
	FINALMARK	NUMBER(2)	

	S_STUDENT		
u✓	STUDNR	NUMBER(8)	NOT NULL
	SURNAME	VARCHAR2(15)	
	INITIALS	VARCHAR2(15)	
	SEX	CHAR(1)	
	BIRTHDATE	DATE	

Key Symbols	
u✓	Primary Key
✓	Foreign Key
u	Unique Key