# CA2 – Preparation Questions

**Question 1.**

The publishing company controlling the publishers database have decided to address the issue of invalid book types/categories being entered into the database.

You have been requested to write a function to do the following:

* + Write a function called CheckBookType().
  + This function take one parameter which is the book type to be checked.
  + The return type for this function is Boolean.
  + The purpose of the function is to check the book type sent in as a parameter against a list of acceptable book types. If the book is a valid type then true is returned otherwise false is returned.
  + Acceptable book types are business, psychology,mod\_cook, trad\_cook, popular\_comp.
  + You are also to write the block of code to test this function.
  + The block is to include an IF statement where if the book is valid you insert a new column into the titles table for the columns title\_id, title, type1, pubdate,contract.
  + You can enter the 1234 for the title\_id, test for the title, the type you tested for the type, sysdate for pubdate and 1234 for contract.
  + If the book type is invalid the block should display a message to this effect.

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION CheckBookType(

p\_Type Titles.type1%TYPE

)

RETURN BOOLEAN IS

v\_Type VARCHAR2;

v\_ReturnValue BOOLEAN;

BEGIN

SELECT type1

INTO v\_Type

FROM titles

WHERE type1 = p\_Type;

IF (v\_Type = 'psychology') THEN

INSERT INTO TITLES (title\_id, title, type1, pubdate,contract) VALUES (1234,'Test', p\_Type, sysdate, 1234);

v\_ReturnValue := TRUE;

ELSIF (v\_Type = 'mod\_cook') THEN

INSERT INTO TITLES (title\_id, title, type1, pubdate,contract) VALUES (1234,'Test', p\_Type, sysdate, 1234);

v\_ReturnValue := TRUE;

ELSIF (v\_Type = 'trad\_cook') THEN

INSERT INTO TITLES (title\_id, title, type1, pubdate,contract) VALUES (1234,'Test', p\_Type, sysdate, 1234);

v\_ReturnValue := TRUE;

ELSIF (v\_Type = 'popular\_comp') THEN

INSERT INTO TITLES (title\_id, title, type1, pubdate,contract) VALUES (1234,'Test', p\_Type, sysdate, 1234);

v\_ReturnValue := TRUE;

ELSE

DBMS\_OUTPUT.PUT\_LINE('THE BOOK TYPE ' || p\_Type|| ' DOES NOT EXIST');

v\_ReturnValue := FALSE;

END IF;

RETURN v\_ReturnValue;

END CheckBookType;

select \* from titles;

**Question 2.**

**Download CA2PrepTables.txt and execute the given code.**

The publishing company controlling the publishers database have decided that it is necessary to be able to easily update the advance for different book types.

You have been requested to write a function to do the following:

1. Write a Procedure called UpdateAdvance to allow the advance for a particular type of books to be updated.
   * 1. The procedure will take two parameters, the booktype and the new advance
     2. All books of the type given are to have their advance updated to the new advance
     3. All updates are to be recorded in the advance\_temptable. This table expects the following to be recorded for every update, a change Id (use sequence created), the title\_id of book being changed, the title of the book, the old advance and the new advance.
2. Write the block of code to test the procedure given above.
3. Write a Trigger which will fire if the advance for the titles table is updated to being less than 0 or greater than 100.
4. Show the block to test the trigger.

(a)

CREATE OR REPLACE PROCEDURE UpdateAdvance(

p\_BookType TITLES.type1%TYPE,

p\_newAdvance number) AS

v\_oldAdvance number;

CURSOR c\_TYPE1 IS

SELECT title

FROM titles

WHERE type1 = p\_BookType

FOR UPDATE OF advance;

BEGIN

FOR v\_TitleInfo IN c\_TYPE1 LOOP

SELECT advance

INTO v\_oldAdvance

FROM Titles

WHERE type1 = p\_BookType;

UPDATE titles

SET advance = p\_newAdvance

WHERE CURRENT OF c\_TYPE1;

INSERT INTO advance\_temptable (AdvanceChange\_id,title\_id,title,old\_advance,new\_advance) VALUES (advance\_sequence.NEXTVAL, v\_TitleInfo.title\_id, v\_TitleInfo.title, v\_oldAdvance, p\_newAdvance);

END LOOP;

COMMIT;

END;

(b)

Begin

UpdateAdvance(‘trad\_cook’,100);

END;

1. Write a Trigger which will fire if the advance for the titles table is updated to being less than 0 or greater than 100.

SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER ad\_check

BEFORE INSERT OR UPDATE ON titles

FOR EACH ROW

WHEN (NEW.advance < 0 OR NEW.advance > 100)

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘Advance update cannot be less then zero or more than 100!!’);

END;