**ОТЧЕТ**

**По лабораторной работе №6**

**«Программные объекты БД»**

**Вариант 8**

Студентка группы 13  
Врублевская Е. А.

**Задание 1**

**1. Добавьте в таблицу SALARY столбец TAX (налог) для вычисления ежемесячного**

**подоходного налога на зарплату по прогрессивной шкале. Налог вычисляется по**

**следующему правилу:**

налог равен 9% от начисленной в месяце зарплаты, если суммарная зарплата с

начала года до конца рассматриваемого месяца не превышает 20 000;

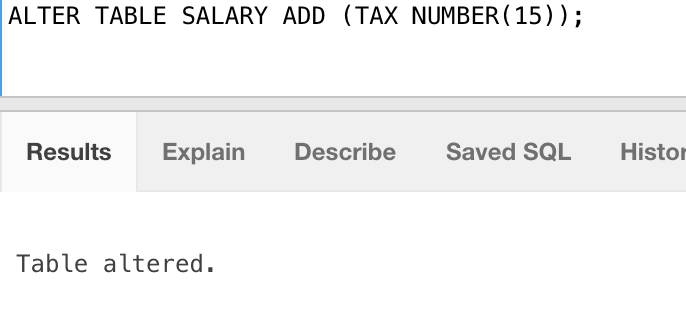
налог равен 12% от начисленной в месяце зарплаты, если суммарная зарплата с

начала года до конца рассматриваемого месяца больше 20 000, но не превышает 30

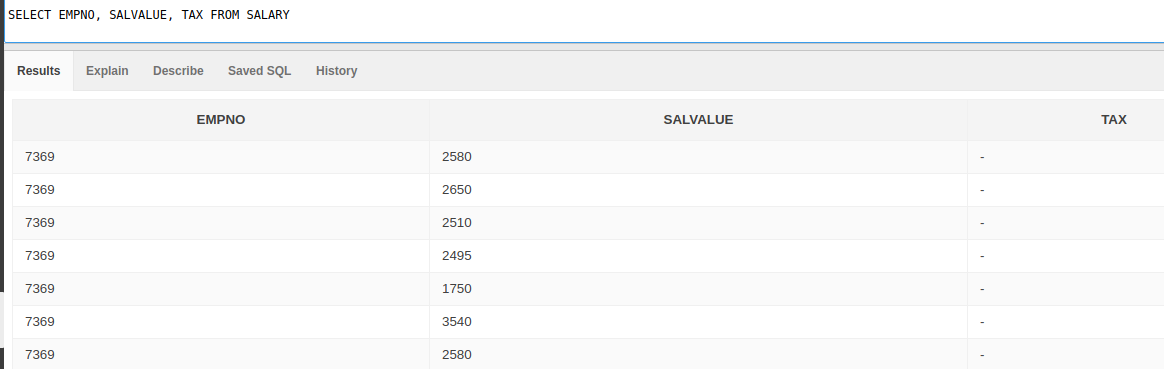
000; налог равен 15% от начисленной в месяце зарплаты, если суммарная зарплата с

начала года до конца рассматриваемого месяца больше 30 000.

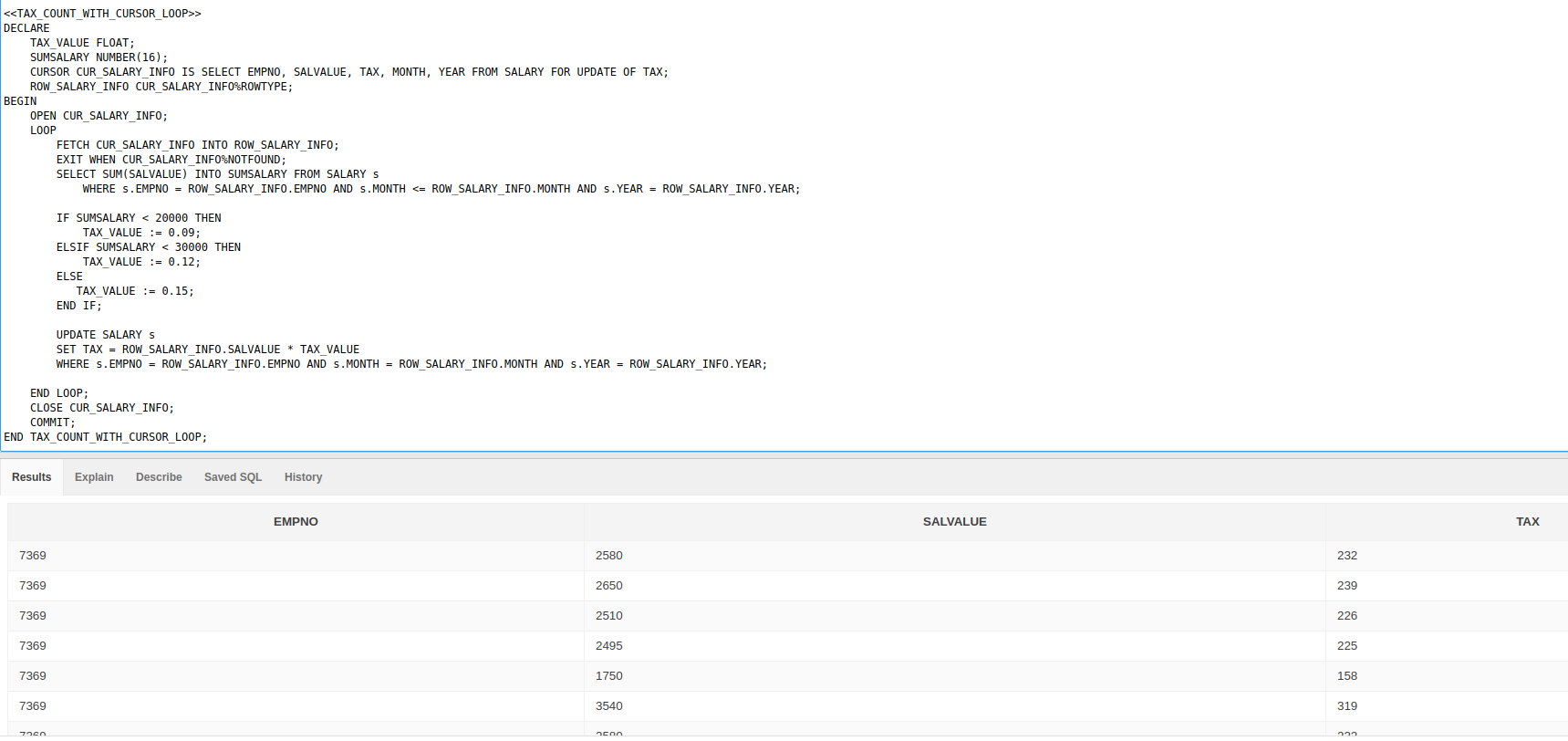
ALTER TABLE SALARY ADD (TAX NUMBER(15));



**2. Составьте программу вычисления налога и вставки её в таблицу SALARY:**



**a) с помощью простого цикла (loop) с курсором и оператора if;**



<<TAX\_COUNT\_WITH\_CURSOR\_LOOP\_IF>>

DECLARE

TAX\_VALUE FLOAT;

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

ROW\_SALARY\_INFO CUR\_SALARY\_INFO%ROWTYPE;

BEGIN

OPEN CUR\_SALARY\_INFO;

LOOP

FETCH CUR\_SALARY\_INFO INTO ROW\_SALARY\_INFO;

EXIT WHEN CUR\_SALARY\_INFO%NOTFOUND;

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

IF SUMSALARY < 20000 THEN

TAX\_VALUE := 0.09;

ELSIF SUMSALARY < 30000 THEN

TAX\_VALUE := 0.12;

ELSE

TAX\_VALUE := 0.15;

END IF;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* TAX\_VALUE

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

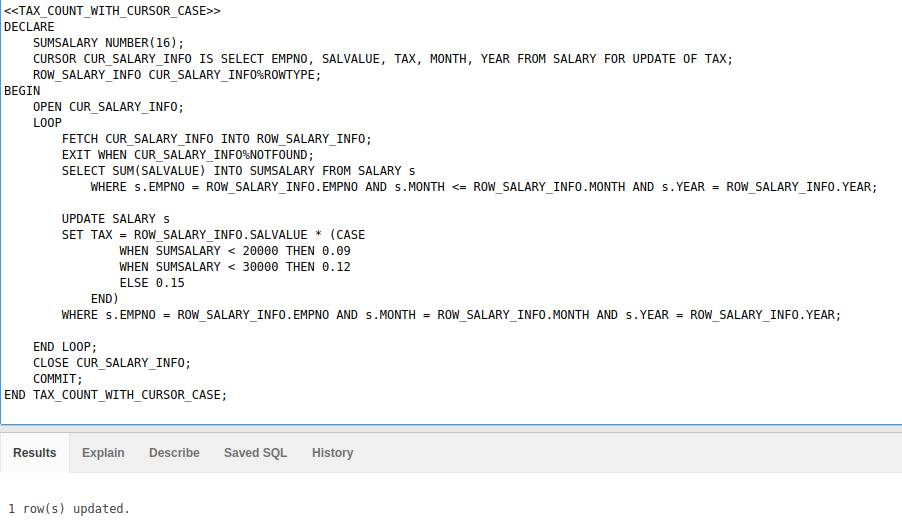
END LOOP;

CLOSE CUR\_SALARY\_INFO;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_LOOP\_IF;

**b) с помощью простого цикла (loop) с курсором и оператора case;**



<<TAX\_COUNT\_WITH\_CURSOR\_CASE>>

DECLARE

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

ROW\_SALARY\_INFO CUR\_SALARY\_INFO%ROWTYPE;

BEGIN

OPEN CUR\_SALARY\_INFO;

LOOP

FETCH CUR\_SALARY\_INFO INTO ROW\_SALARY\_INFO;

EXIT WHEN CUR\_SALARY\_INFO%NOTFOUND;

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

CLOSE CUR\_SALARY\_INFO;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_CASE;

**c) с помощью курсорного цикла FOR;**

<<TAX\_COUNT\_WITH\_CURSOR\_IN\_LOOP>>

DECLARE

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_IN\_LOOP;

**d) с помощью курсора с параметром, передавая номер сотрудника, для которого**

**необходимо посчитать налог.**

<<TAX\_COUNT\_WITH\_CURSOR\_PARAM>>

DECLARE

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO(EMPID NUMBER) IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY

WHERE EMPNO = EMPID

FOR UPDATE OF TAX;

CURSOR CUR\_EMP IS SELECT EMPNO FROM EMP;

BEGIN

FOR EMP\_R IN CUR\_EMP LOOP

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO(EMP\_R.EMPNO) LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

END LOOP;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_PARAM;

**3. Оформите составленные программы в виде процедур.**

CREATE OR REPLACE PROCEDURE TAX\_COUNT\_WITH\_CURSOR\_LOOP\_IF IS

TAX\_VALUE FLOAT;

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

ROW\_SALARY\_INFO CUR\_SALARY\_INFO%ROWTYPE;

BEGIN

OPEN CUR\_SALARY\_INFO;

LOOP

FETCH CUR\_SALARY\_INFO INTO ROW\_SALARY\_INFO;

EXIT WHEN CUR\_SALARY\_INFO%NOTFOUND;

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

IF SUMSALARY < 20000 THEN

TAX\_VALUE := 0.09;

ELSIF SUMSALARY < 30000 THEN

TAX\_VALUE := 0.12;

ELSE

TAX\_VALUE := 0.15;

END IF;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* TAX\_VALUE

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

CLOSE CUR\_SALARY\_INFO;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_LOOP\_IF;

CREATE OR REPLACE PROCEDURE TAX\_COUNT\_WITH\_CURSOR\_CASE IS

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

ROW\_SALARY\_INFO CUR\_SALARY\_INFO%ROWTYPE;

BEGIN

OPEN CUR\_SALARY\_INFO;

LOOP

FETCH CUR\_SALARY\_INFO INTO ROW\_SALARY\_INFO;

EXIT WHEN CUR\_SALARY\_INFO%NOTFOUND;

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

CLOSE CUR\_SALARY\_INFO;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_CASE;

CREATE OR REPLACE PROCEDURE TAX\_COUNT\_WITH\_CURSOR\_IN\_LOOP IS

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_IN\_LOOP

CREATE OR REPLACE PROCEDURE TAX\_COUNT\_WITH\_CURSOR\_PARAM (EMPID NUMBER) IS

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO(EMPID NUMBER) IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY

WHERE EMPNO = EMPID

FOR UPDATE OF TAX;

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO(EMPID) LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

COMMIT;

END TAX\_COUNT\_WITH\_CURSOR\_PARAM;

**4. Создайте функцию, вычисляющую налог на зарплату за всё время начислений для конкретного сотрудника. В качестве параметров передать процент налога (до 20000, до 30000, выше 30000, номер сотрудника).**

CREATE OR REPLACE PROCEDURE COUNT\_TAX\_FOR\_EMP (EMPID NUMBER, UNDER\_20 NUMBER, OVER\_20\_AND\_UNDER\_30 NUMBER, OVER\_30 NUMBER) IS

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, YEAR, MONTH FROM SALARY

WHERE EMPNO = EMPID

FOR UPDATE OF TAX;

SUMSALARY NUMBER(16);

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN UNDER\_20

WHEN SUMSALARY < 30000 THEN OVER\_20\_AND\_UNDER\_30

ELSE OVER\_30

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

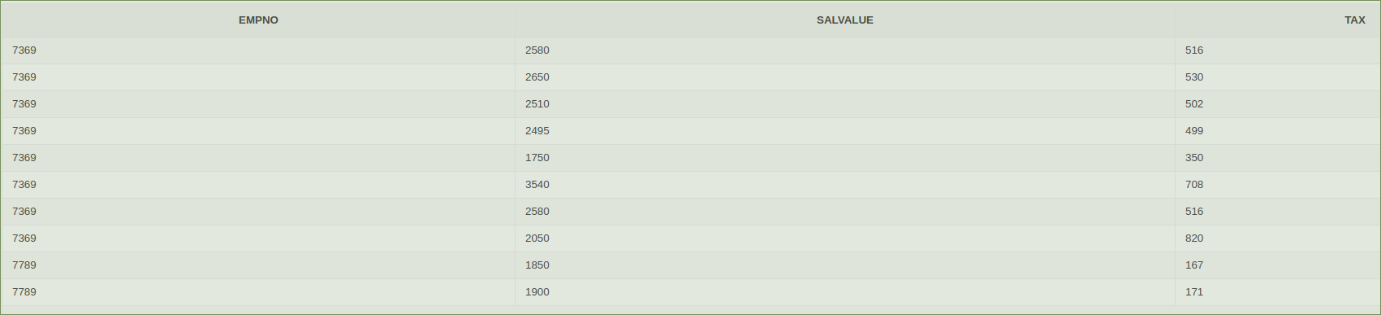
COMMIT;

END COUNT\_TAX\_FOR\_EMP;

BEGIN

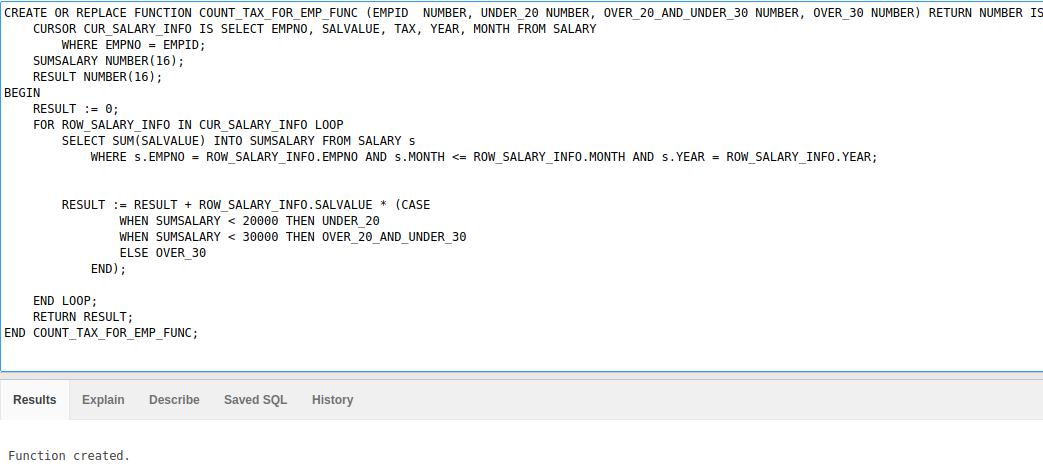
COUNT\_TAX\_FOR\_EMP(7369, 0.20, 0.40, 0.75);

END;



**5. Создайте подпрограмму, вычисляющую суммарный налог на зарплату сотрудника**

**за всё время начислений. В качестве параметров передать процент налога (до 20000, до 30000, выше 30000, номер сотрудника). Возвращаемое значение – суммарный налог.**



CREATE OR REPLACE FUNCTION COUNT\_TAX\_FOR\_EMP\_FUNC (EMPID NUMBER, UNDER\_20 NUMBER, OVER\_20\_AND\_UNDER\_30 NUMBER, OVER\_30 NUMBER) RETURN NUMBER IS

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, YEAR, MONTH FROM SALARY

WHERE EMPNO = EMPID;

SUMSALARY NUMBER(16);

RESULT NUMBER(16);

BEGIN

RESULT := 0;

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

RESULT := RESULT + ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN UNDER\_20

WHEN SUMSALARY < 30000 THEN OVER\_20\_AND\_UNDER\_30

ELSE OVER\_30

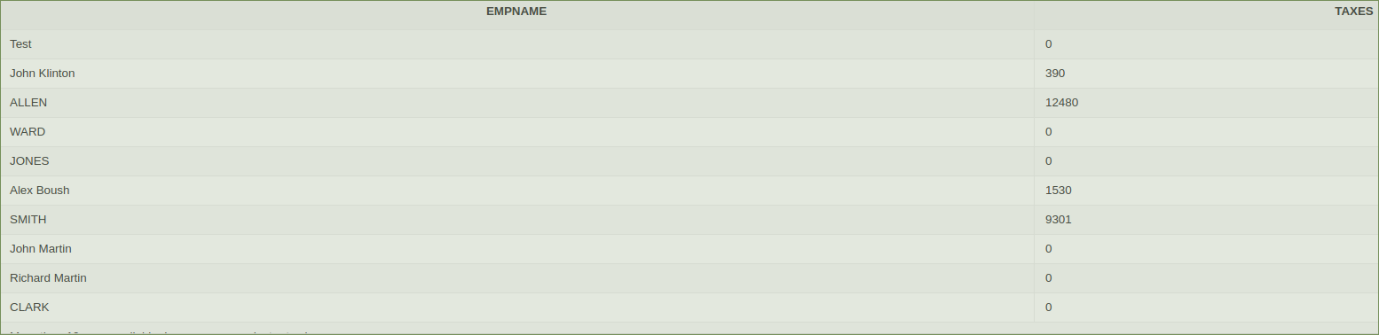
END);

END LOOP;

RETURN RESULT;

END COUNT\_TAX\_FOR\_EMP\_FUNC;

SELECT EMPNAME, COUNT\_TAX\_FOR\_EMP\_FUNC(EMPNO, 0.20, 0.40, 0.75) AS TAXES FROM EMP;

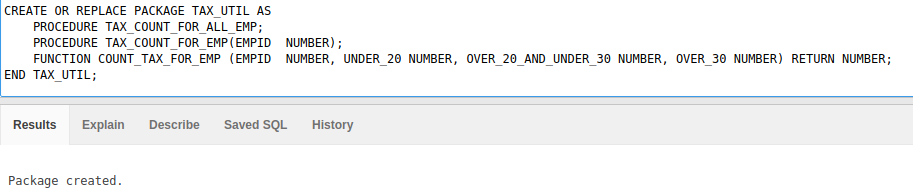


**6. Создайте пакет, включающий в свой состав процедуру вычисления налога для всех**

**сотрудников, процедуру вычисления налогов для отдельного сотрудника,**

**идентифицируемого своим номером, функцию вычисления суммарного налога на**

**зарплату сотрудника за всё время начислений.**



CREATE OR REPLACE PACKAGE TAX\_UTIL AS

PROCEDURE TAX\_COUNT\_FOR\_ALL\_EMP;

PROCEDURE TAX\_COUNT\_FOR\_EMP(EMPID NUMBER);

FUNCTION COUNT\_TAX\_FOR\_EMP (EMPID NUMBER, UNDER\_20 NUMBER, OVER\_20\_AND\_UNDER\_30 NUMBER, OVER\_30 NUMBER) RETURN NUMBER;

END TAX\_UTIL;

CREATE OR REPLACE PACKAGE BODY TAX\_UTIL AS

FUNCTION COUNT\_TAX\_FOR\_EMP (EMPID NUMBER, UNDER\_20 NUMBER, OVER\_20\_AND\_UNDER\_30 NUMBER, OVER\_30 NUMBER) RETURN NUMBER IS

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, YEAR, MONTH FROM SALARY

WHERE EMPNO = EMPID;

SUMSALARY NUMBER(16);

RESULT NUMBER(16);

BEGIN

RESULT := 0;

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

RESULT := RESULT + ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN UNDER\_20

WHEN SUMSALARY < 30000 THEN OVER\_20\_AND\_UNDER\_30

ELSE OVER\_30

END);

END LOOP;

RETURN RESULT;

END COUNT\_TAX\_FOR\_EMP;

PROCEDURE TAX\_COUNT\_FOR\_EMP (EMPID NUMBER) IS

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO(EMPID NUMBER) IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY

WHERE EMPNO = EMPID

FOR UPDATE OF TAX;

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO(EMPID) LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

COMMIT;

END TAX\_COUNT\_FOR\_EMP;

PROCEDURE TAX\_COUNT\_FOR\_ALL\_EMP IS

SUMSALARY NUMBER(16);

CURSOR CUR\_SALARY\_INFO IS SELECT EMPNO, SALVALUE, TAX, MONTH, YEAR FROM SALARY FOR UPDATE OF TAX;

BEGIN

FOR ROW\_SALARY\_INFO IN CUR\_SALARY\_INFO LOOP

SELECT SUM(SALVALUE) INTO SUMSALARY FROM SALARY s

WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH <= ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

UPDATE SALARY s

SET TAX = ROW\_SALARY\_INFO.SALVALUE \* (CASE

WHEN SUMSALARY < 20000 THEN 0.09

WHEN SUMSALARY < 30000 THEN 0.12

ELSE 0.15

END)

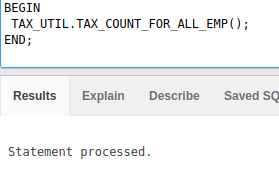
WHERE s.EMPNO = ROW\_SALARY\_INFO.EMPNO AND s.MONTH = ROW\_SALARY\_INFO.MONTH AND s.YEAR = ROW\_SALARY\_INFO.YEAR;

END LOOP;

COMMIT;

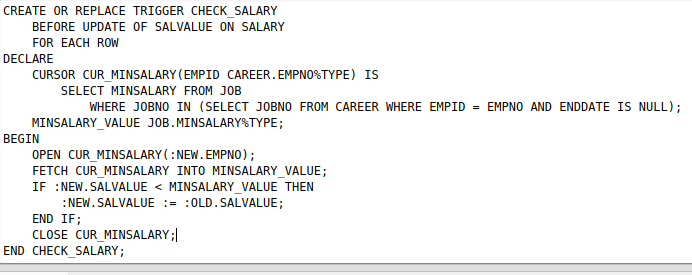
END TAX\_COUNT\_FOR\_ALL\_EMP;

END TAX\_UTIL;



**7. Создайте триггер, действующий при обновлении данных в таблице SALARY. А именно, если происходит обновление поля SALVALUE, то при назначении новой зарплаты, меньшей чем должностной оклад (таблица JOB, поле MINSALARY), изменение не вносится и сохраняется старое значение, если новое значение зарплаты больше должностного оклада, то изменение вносится.**

**WHEN**



CREATE OR REPLACE TRIGGER CHECK\_SALARY

BEFORE UPDATE OF SALVALUE ON SALARY

FOR EACH ROW

DECLARE

CURSOR CUR\_MINSALARY(EMPID CAREER.EMPNO%TYPE) IS

SELECT MINSALARY FROM JOB

WHERE JOBNO IN (SELECT JOBNO FROM CAREER WHERE EMPID = EMPNO AND ENDDATE IS NULL);

MINSALARY\_VALUE JOB.MINSALARY%TYPE;

BEGIN

OPEN CUR\_MINSALARY(:NEW.EMPNO);

FETCH CUR\_MINSALARY INTO MINSALARY\_VALUE;

IF :NEW.SALVALUE < MINSALARY\_VALUE THEN

:NEW.SALVALUE := :OLD.SALVALUE;

END IF;

CLOSE CUR\_MINSALARY;

END CHECK\_SALARY;

SELECT \* FROM SALARY

WHERE EMPNO = 7369 AND MONTH = 5 AND YEAR = 2007;



UPDATE SALARY

SET SALVALUE = 100

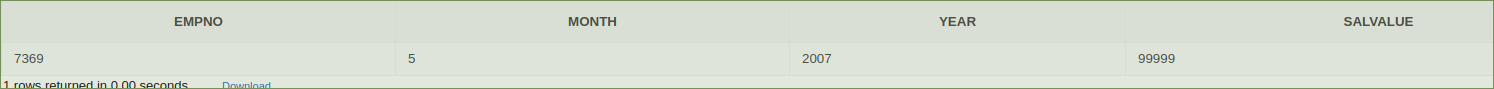
WHERE EMPNO = 7369 AND MONTH = 5 AND YEAR = 2007;



UPDATE SALARY

SET SALVALUE = 99999

WHERE EMPNO = 7369 AND MONTH = 5 AND YEAR = 2007;



**--8) Создайте триггер, действующий при удалении записи из таблицы CAREER. Если в удаляемой строке поле ENDDATE содержит NULL, то запись не удаляется, в противном случае удаляется.**

CREATE OR REPLACE TRIGGER HANDLE\_DELETE\_FOR\_WORKERS

BEFORE DELETE ON CAREER

FOR EACH ROW

WHEN (OLD.ENDDATE IS NULL)

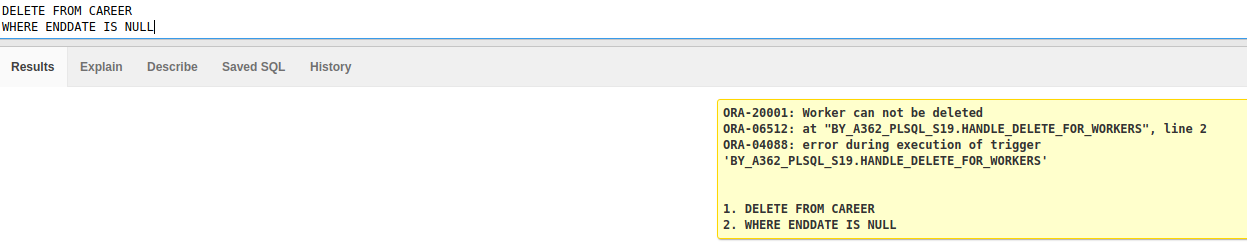
BEGIN

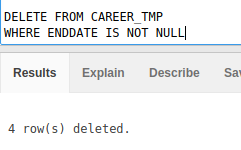
RAISE\_APPLICATION\_ERROR(-20001,'Worker can not be deleted');

END HANDLE\_DELETE\_FOR\_WORKERS;

DELETE FROM CAREER

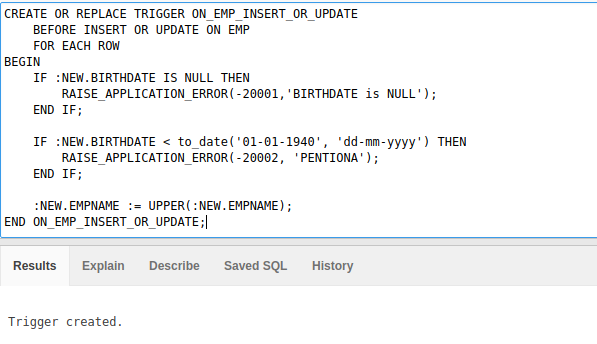
WHERE ENDDATE IS NULL





**9. Создайте триггер, действующий на добавление или изменение данных в таблице**

**EMP. Если во вставляемой или изменяемой строке поле BIRTHDATE содержит NULL, то после вставки или изменения должно быть выдано сообщение ‘BERTHDATE is NULL’. Если во вставляемой или изменяемой строке поле BIRTHDATE содержит дату ранее ‘01- 01-1940’, то должно быть выдано сообщение ‘PENTIONA’. Во вновь вставляемой строке имя служащего должно быть приведено к заглавным буквам.**



CREATE OR REPLACE TRIGGER ON\_EMP\_INSERT\_OR\_UPDATE

BEFORE INSERT OR UPDATE ON EMP

FOR EACH ROW

BEGIN

IF :NEW.BIRTHDATE IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001,'BIRTHDATE is NULL');

END IF;

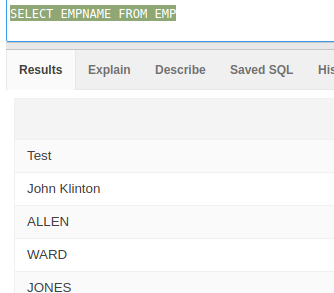
IF :NEW.BIRTHDATE < to\_date('01-01-1940', 'dd-mm-yyyy') THEN

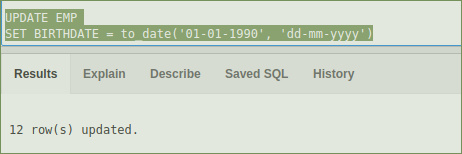
RAISE\_APPLICATION\_ERROR(-20002, 'PENTIONA');

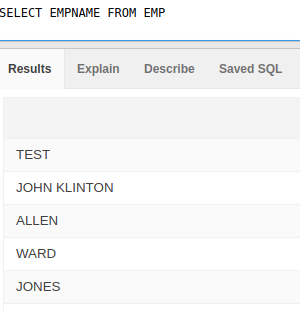
END IF;

:NEW.EMPNAME := UPPER(:NEW.EMPNAME);

END ON\_EMP\_INSERT\_OR\_UPDATE;



****

****

**10. Создайте программу изменения типа заданной переменной из символьного типа**

**(VARCHAR2) в числовой тип (NUMBER). Программа должна содержать раздел**

**обработки исключений. Обработка должна заключаться в выдаче сообщения ‘ERROR:**

**argument is not a number’ . Исключительная ситуация возникает при задании строки в виде числа с запятой, разделяющей дробную и целую части.**

CREATE OR REPLACE FUNCTION VARCHAR2\_TO\_NUMBER (str in VARCHAR2) return NUMBER IS

BEGIN

RETURN CAST(str AS NUMBER);

EXCEPTION

WHEN VALUE\_ERROR THEN

RAISE\_APPLICATION\_ERROR(-20108, 'ERROR: argument is not a number: ' || str);

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20103, 'Unexpected error');

END VARCHAR2\_TO\_NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(VARCHAR2\_TO\_NUMBER( '1234123' ));

DBMS\_OUTPUT.PUT\_LINE(VARCHAR2\_TO\_NUMBER( '5.1' ));

DBMS\_OUTPUT.PUT\_LINE(VARCHAR2\_TO\_NUMBER('4,123'));

END;

**Задание 2**

1. Процедура распределения урожая. Входные параметры – название культуры, количество удобрений, режим полива и вегетационный период. Действие процедуры – выделение подходящего участка под культуру, объём воды и удобрений. Для новой культуры добавить соответствующие записи в базу данных. Выходной параметр – номер участка.

CREATE OR REPLACE PROCEDURE PROC\_PLANT (UDOBR NUMBER, WATER NUMBER, VEGET NUMBER, UCH\_ID OUT INTEGER) AS

PLANT\_ID INTEGER;

POSEV INTEGER;

POLIV INTEGER;

BEGIN

PLANT\_ID := SEQ\_PLANT.NEXTVAL;

POSEV := 2;

POLIV := 2;

SELECT U.ID INTO UCH\_ID FROM УЧАСТОК U INNER JOIN КУЛЬТУРА К ON U.КУЛЬТУРА\_ID=К.ID WHERE (U.КУЛЬТУРА\_ID IS NULL AND U.ПЛОЩАДЬ >= WATER \* POLIV AND U.ПЛОЩАДЬ >= UDOBR \* POSEV);

INSERT INTO КУЛЬТУРА VALUES(PLANT\_ID, UDOBR, POSEV, '2', VEGET, '12-AUG-2022', 300, WATER);

UPDATE УЧАСТОК SET КУЛЬТУРА\_ID = PLANT\_ID WHERE ID = UCH\_ID;

END PROC\_PLANT;

1. Функция для определения необходимого количества воды на участок. Входной параметр – номер участка. Действие функции – определение суммарного объема подачи воды на указанный участок с учётом потребностей высаженных на участке культур. Выходной параметр – суммарный объем подачи воды в литрах.

CREATE OR REPLACE FUNCTION FUNC\_WATER(NUM\_OF\_UCH NUMBER) RETURN NUMBER AS

RESULT NUMBER;

BEGIN

SELECT SUM(K.ВОДА\_ПОЛИВ \* TO\_NUMBER(K.РЕЖИМ\_ПОЛИВА) \* 12) INTO RESULT

FROM КУЛЬТУРА K INNER JOIN УЧАСТОК U ON U.КУЛЬТУРА\_ID = K.ID

WHERE U.ID = NUM\_OF\_UCH;

RETURN RESULT;

END FUNC\_WATER;

1. Триггер для ограничения суммы затрат на хозяйственный сектор.

CREATE OR REPLACE TRIGGER TRIG\_EXPENCES

BEFORE INSERT OR UPDATE ON ИНВЕНТАРЬ FOR EACH ROW

BEGIN

IF :NEW.СТОИМОСТЬ > 10000 THEN

RAISE\_APPLICATION\_ERROR ( -20000, 'VERY EXPENSIVE ITEM');

END IF;

END