**My notes:**

**Print syntax:**

DBMS\_OUTPUT.PUT\_LINE('Hours of last day: '|| (ROUND((trunc(p\_work\_end))\*24,1))

**call procedure syntax:**

BEGIN

TASK1.Q2;

END;

**call function syntax:**

SELECT task1.Q3(42,98) FROM DUAL;

* **QUESTIONS**
* **HOW TO EXECUTE SINGLE LINE**

**TASK 1. Simple Tasks**

* 1. Write PL-SQL program, which prints multipliers of number 9 i.e., 9,18,27, …, up to 30\*9
  2. Prepare PL/SQL procedure which does the same as the task 1. Run that procedure.
  3. Write a function which returns the greatest common divisor (highest common factor) of two numbers.
  4. Prepare a function which displays n-th number in a Fibonacci sequence (1,1,2,3,5,8,13...) use recursion, then do it without using recursion.
  5. Create a table with the name »TEST« with columns »NAME« and »SURNAME« and write procedure, that inserts data to it. Parameters should be p\_name and p\_surname. Call procedure and check if data was added to the table.
  6. Change procedure from task 5. in a way that would not allow to insert the same person multiple times
  7. Add unique key to the table, so it will not be possible to insert one person more than one time
  8. Study which types are supported in oracle database. From now on use only the following:

NUMBER for all numeric columns,

VARCHAR2(size) for all character columns and

DATE for all date columns.

It is much easier if we don't over-complicate our lives.

**TASK 2. Meaningful names of tables and columns**

2.1 Use constants in task 1.1

2.2 Create table with the same columns as »TEST« but change a name to make it more meaningful. Add column id, which should be used as primary key, create sequence that you will insert when adding the data. Adapt procedure from task 1.6 in a way that it will insert name, surname, and sequence value. The data should be inserted in new table, of course.

2.3 Create table with the information of the presence in the job. Table should have meaningful name, the columns should include the data about: Date, Number of hours, Comment. The table from task 2.2 should be connected using the field id. Define foreign key between those two tables.

2.4 Add to the table from task 2.3 a column which hold information about regular hours/overtime hours. Create new table with this information (hour types) and create foreign key between them.

2.5 Create procedure, which inserts data to table from task 2.3. Parameters should be id of a person, number of hours and hour type.

* 1. Create procedure, which inserts data to table from task 2.3. Parameters should be id of a person, work start date, work end date (date columns in oracle include data about hours, minutes and seconds also). Your procedure should calculate number of hours from provided dates. If start date and end date are different (not in the same day) insert one record for each day (calculate and insert the appropriate number of hours). Use procedure from task 2.5 to insert one record. A person can for each day have 8 regular hours only, the rest should be overtime hours.
  2. Prepare database schema (Create tables, primary keys, foreign keys) to store information of employees and cars of a company. Include data about car manufacturer, car model, car manufacturing year, colour, equipment package, registration number, add a column for comment. Use master/detail as you find meaningful. It should be possible for one employee to have multiple cars. Schema should also include data about history of car registrations.

**TASK 3. Exceptions handling, indexes, execution time measurements**

* 1. Write a procedure, which inserts new person to table from task 2.7. Procedure should have parameters: p\_name, p\_surname and output parameters p\_status and p\_msg. Add exception handler to BEGIN-END block which executes ROLLBACK in case of error and returns p\_status='ERR' and p\_msg=…error message (use SQLERRM built in variable). If procedure finishes successfully execute COMMIT and return p\_status='OK'. Call this procedure from anonymous PL/SQL block and write returned status and message to Simple Tasks.
  2. Write a procedure that generates person names, surnames from sequence numbers (id=10, name=10, surname=10,…) – just to fill something to the table. Use sequence for id (using: Simple Tasks () ). Insert 1000 such persons.
  3. Add column status to persons table. Create procedure, which updates that table and set column status='A'. Use cursor, which loops through all persons in the table and update each of them. Calculate time to do that with the help of function DBMS\_UTILITY.GET\_TIME.
  4. Prepare procedure that does the same job as the procedure from task 3.3 but it updates all records in one SQL statement. Measure time needed to do that. Compare execution times of procedures from tasks 3.3 and 3.4.

1. Add additional 1000000 with the help of procedure from task 3.2. Add column person\_group to the persons table. Update column with the numbers form 1 till 10 using folowing rules: person with id=1 person\_group=1, id=2 person\_group=2... id=10 person\_group=10, id=11 person\_group=1, id=12 person\_group=2,...
2. Update column status='A' for persons with person\_group=1. Measure time.
3. Create index on table persons on the column person\_group.
4. Repeat task 3.6, measure time and compare it with the one from task 3.6

**ANSWERS:**

**Task 1.**

* 1. **--** code that prints multiples of 9

DECLARE

i NUMBER := 1;

BEGIN

LOOP

EXIT WHEN i> 30 ;

DBMS\_OUTPUT.PUT\_LINE(9 \* i);

i := i+1;

END LOOP;

END;

* 1. **--** procedure that prints multiples of 9

CREATE OR REPLACE PROCEDURE PRINT\_MULTIPLES\_OF\_9 AS

i NUMBER := 1;

BEGIN

LOOP

EXIT WHEN i> 30 ;

DBMS\_OUTPUT.PUT\_LINE(9 \* i);

i := i+1;

END LOOP;

END PRINT\_MULTIPLES\_OF\_9;

* 1. **–** function that returns Highest Common Factor given two numbers

CREATE OR REPLACE FUNCTION HCF

(

p\_num1 IN number, p\_num2 IN number

) RETURN NUMBER AS

num1 NUMBER:=p\_num1;

num2 NUMBER:=p\_num2;

temp NUMBER;

BEGIN

WHILE MOD(num2, num1) != 0 LOOP

temp := MOD(num2, num1);

num2 := num1;

num1 := temp;

END LOOP;

RETURN num1;

END HCF;

**1.4** -- function that returns nth Fibonacci number

**Recursive**

create or replace FUNCTION FIB\_RECURSIVE

(

K IN NUMBER

) RETURN NUMBER AS

BEGIN

IF k= 1 THEN

RETURN 0;

ELSIF k = 2 THEN

RETURN 1;

ELSE

RETURN FIB\_RECURSIVE(K - 2) + FIB\_RECURSIVE(K - 1);

END IF;

END FIB\_RECURSIVE;

**Non-recursive**

CREATE OR REPLACE FUNCTION FIB\_NON\_RECURSIVE

(

p\_k IN NUMBER

) RETURN NUMBER AS

num1 NUMBER:= 0;

num2 NUMBER:= 1;

k NUMBER := p\_k;

temp NUMBER := 0;

nsum NUMBER := 0;

BEGIN

WHILE k > temp LOOP

nsum := nsum + num1;

num1 := num2;

num2 := nsum;

temp := temp+1;

END LOOP;

RETURN nsum;

END FIB\_NON\_RECURSIVE;

**1.5** -- create table test

CREATE TABLE test

(

NAME VARCHAR2(20),

SURNAME VARCHAR2(40)

);

-- procedure of adding records without any checks

create or replace PROCEDURE ADD\_DATA\_TO\_TABLE\_TEST

(

P\_NAME IN TEST.NAME%TYPE

, P\_SURNAME IN TEST.SURNAME%TYPE

) AS

BEGIN

INSERT INTO TEST ("NAME", "SURNAME")

VALUES (P\_NAME, P\_SURNAME);

COMMIT;

END ADD\_DATA\_TO\_TABLE\_TEST;

**1.6** -- procedure of adding records if they don’t exist

CREATE OR REPLACE PROCEDURE ADD\_TO\_TEST\_WITH\_CHECK

(

P\_NAME IN TEST.NAME%TYPE

, P\_SURNAME IN TEST.SURNAME%TYPE

) AS

BEGIN

INSERT INTO TEST ("NAME", "SURNAME")

SELECT p\_name, p\_surname

FROM TEST

WHERE NOT EXISTS

(

SELECT 1 FROM TEST

WHERE "NAME" = p\_name AND "SURNAME" = p\_surname

);

COMMIT;

END ADD\_TO\_TEST\_WITH\_CHECK;

**1.7 --** Alter table test by adding primary key

ALTER TABLE TEST

ADD UNIQUE\_KEY NUMBER NOT NULL;

ALTER TABLE TEST

ADD CONSTRAINT pk\_1 PRIMARY KEY (UNIQUE\_KEY);

**2.1** -- task 1.1 with the use of constant

DECLARE

i NUMBER := 1;

multiplier NUMBER := 9;

BEGIN

LOOP

EXIT WHEN i> 30 ;

DBMS\_OUTPUT.PUT\_LINE(multiplier \* i);

i := i+1;

END LOOP;

END;

**2.2** -- creating ID\_Generator

CREATE SEQUENCE Id\_Generator\_1

START WITH 1

INCREMENT BY 1;

-- creating “Add employee” procedure with “if doesn’t exist” check

create or replace procedure ADD\_EMPLOYEE

(

p\_name IN EMPLOYEES.NAME%TYPE,

p\_surname IN EMPLOYEES.SURNAME%TYPE

) AS

BEGIN

INSERT INTO EMPLOYEES ("ID","NAME","SURNAME" )

select Id\_Generator.NEXTVAL, p\_name,p\_surname from dual

where not exists

(

select NULL

from EMPLOYEES

where "NAME" = p\_name AND "SURNAME" = p\_surname

);

END ADD\_EMPLOYEE;

-- creating table “employees”

DROP TABLE EMPLOYEES;

CREATE TABLE EMPLOYEES -- Create table

(

ID NUMBER NOT NULL,

NAME VARCHAR2(20),

SURNAME VARCHAR2(40),

CONSTRAINT EMPLOYEE\_ID PRIMARY KEY (ID)

);

-- adding data using “Add employee” procedure

BEGIN

ADD\_EMPLOYEE('Alice', 'AAA');

END;

**2.3** -- creating table “Job\_Presence” with foreign key reference to table “Employees”

CREATE TABLE JOB\_PRESENCE

(

EMPLOYEE\_ID NUMBER NOT NULL,

"DATE" DATE,

HOURS\_OF\_WORK NUMBER,

"COMMENT" VARCHAR2(255),

CONSTRAINT FK\_ID\_EMPLOYEE

FOREIGN KEY (EMPLOYEE\_ID)

REFERENCES EMPLOYEES(ID)

);

**2.4** -- creating table “hours\_type”

CREATE TABLE HOURS\_TYPE

(

HOURS\_TYPE\_ID NUMBER NOT NULL,

HOURS\_TYPE VARCHAR2(30),

CONSTRAINT PK\_HOURS\_TYPE

PRIMARY KEY (HOURS\_TYPE\_ID)

);

-- adding a column to table “Job\_Presence” with hours type

-- adding foreign key reference to create a link between two tables

ALTER TABLE JOB\_PRESENCE

ADD

(

"HOURS\_TYPE\_ID" NUMBER,

CONSTRAINT FK\_HOURS\_TYPE\_ID\_HOURS\_TYPE

FOREIGN KEY (HOURS\_TYPE\_ID)

REFERENCES HOURS\_TYPE(HOURS\_TYPE\_ID)

);

**2.5** --inserting data to selected columns of a table

create or replace PROCEDURE TASK\_2\_5

(

p\_employee\_id IN JOB\_PRESENCE.EMPLOYEE\_ID%TYPE,

p\_hours\_of\_work IN JOB\_PRESENCE.HOURS\_OF\_WORK%TYPE,

p\_hours\_type\_id IN JOB\_PRESENCE.HOURS\_TYPE\_ID%TYPE

) AS

BEGIN

INSERT INTO JOB\_PRESENCE

("EMPLOYEE\_ID", "HOURS\_OF\_WORK", "HOURS\_TYPE\_ID")

VALUES (p\_employee\_id, p\_hours\_of\_work, p\_hours\_type\_id);

COMMIT;

END TASK\_2\_5;

**2.6**

DECLARE

employee\_id NUMBER := 2;

hours\_type\_id NUMBER := 1;

p\_work\_start DATE:= '07/01/2020 07:20:10';

p\_work\_end DATE:= '09/01/2020 20:50:50';

p\_work\_start\_day NUMBER := EXTRACT(DAY FROM p\_work\_start);

p\_work\_end\_day NUMBER := EXTRACT(DAY FROM p\_work\_end);

p\_day\_hours NUMBER;

temp\_date DATE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Starting day: '||p\_work\_start\_day);

DBMS\_OUTPUT.PUT\_LINE('Last day: '||p\_work\_end\_day);

p\_day\_hours := ROUND((p\_work\_end - p\_work\_start)\*24,1);

DBMS\_OUTPUT.PUT\_LINE('Hours of work: '||p\_day\_hours);

FOR number\_of\_days IN 0..(p\_work\_end\_day - p\_work\_start\_day) LOOP

DBMS\_OUTPUT.PUT\_LINE('Day no: '|| (number\_of\_days + 1));

IF number\_of\_days = 0

THEN

if (24 - ROUND((p\_work\_start-trunc(p\_work\_start))\*24,1)) < 8

then

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,p\_work\_start,24 - ROUND((p\_work\_start-trunc(p\_work\_start))\*24,1),'first day of work',1);

DBMS\_OUTPUT.PUT\_LINE('Hours of day'|| (number\_of\_days +1)|| ': '|| (24 - ROUND((p\_work\_start-trunc(p\_work\_start))\*24,1)));

else

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,p\_work\_start,8,'first day of work',1);

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,p\_work\_start + (8/24),(24 - ROUND((p\_work\_start-trunc(p\_work\_start))\*24,1))-8,'first day of work',2);

DBMS\_OUTPUT.PUT\_LINE('Hours of day'|| (number\_of\_days +1)|| ': '|| (24 - ROUND((p\_work\_start-trunc(p\_work\_start))\*24,1)));

end if;

ELSIF (number\_of\_days = (p\_work\_end\_day - p\_work\_start\_day))

THEN

select to\_char(cast(p\_work\_end as date),'DD-MM-YYYY') INTO temp\_date FROM dual;

if (ROUND((p\_work\_end - trunc(p\_work\_end))\*24,1)) < 8

then

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,temp\_date,ROUND((p\_work\_end - trunc(p\_work\_end))\*24,1),'last day of work',1);

DBMS\_OUTPUT.PUT\_LINE('Hours of day'|| (number\_of\_days +1)|| ': '|| (ROUND((p\_work\_end - trunc(p\_work\_end))\*24,1)));

else

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,temp\_date,8,'last day of work',1);

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,p\_work\_end - (8/24),ROUND((p\_work\_end - trunc(p\_work\_end))\*24,1)-8,'last day of work',2);

DBMS\_OUTPUT.PUT\_LINE('Hours of day'|| (number\_of\_days +1)|| ': '|| (ROUND((p\_work\_end - trunc(p\_work\_end))\*24,1)));

end if;

ELSE

select to\_char(cast(p\_work\_start as date),'DD-MM-YYYY') INTO temp\_date FROM dual;

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,temp\_date + number\_of\_days,8,'in between',1);

INSERT INTO JOB\_PRESENCE("EMPLOYEE\_ID","DATE","HOURS\_OF\_WORK","COMMENT","HOURS\_TYPE\_ID")

VALUES(employee\_id,temp\_date + (8/24) + number\_of\_days,16,'in between',2);

DBMS\_OUTPUT.PUT\_LINE('Hours of day'|| (number\_of\_days +1)|| ': '|| 24);

END IF;

END LOOP;

END;

2.7

CREATE SCHEMA AUTHORIZATION smithj

CREATE TABLE products

( product\_id number(10) not null,

product\_name varchar2(50) not null,

category varchar2(50),

CONSTRAINT products\_pk PRIMARY KEY (product\_id)

)

CREATE TABLE suppliers

( supplier\_id number(10) not null,

supplier\_name varchar2(50) not null,

city varchar2(25),

CONSTRAINT suppliers\_pk PRIMARY KEY (supplier\_id)

);

-- task 2.7

-- task 2.7

CREATE TABLE CARS\_REG\_PLATES

(

REG\_PLATE VARCHAR2(10) NOT NULL,

REGISTRATION\_INFO VARCHAR2(50),

CONSTRAINT PK\_REG\_PLATE PRIMARY KEY (REG\_PLATE)

);

CREATE TABLE VEHICLE\_EQUIPMENT

(

PACKAGE\_ID NUMBER(10) NOT NULL,

PACKAGE\_NAME VARCHAR2(50),

PACKAGE\_DETAILS VARCHAR2(50),

CONSTRAINT PK\_PACKAGE PRIMARY KEY (PACKAGE\_ID)

);

CREATE TABLE EMPLOYEES

(

EMPLOYEE\_ID NUMBER(10) NOT NULL,

"NAME" VARCHAR2(50),

SURNAME VARCHAR2(50),

CAR\_REG\_PLATE VARCHAR2(10),

CONSTRAINT PK\_EMPLOYEE\_ID PRIMARY KEY (EMPLOYEE\_ID),

CONSTRAINT FK\_CAR\_REG\_PLATE

FOREIGN KEY (CAR\_REG\_PLATE)

REFERENCES CARS\_REG\_PLATES(REG\_PLATE)

);

CREATE TABLE CARS

(

REG\_PLATE VARCHAR2(10) NOT NULL,

CAR\_NAME VARCHAR2(50) NOT NULL,

CAR\_MANUFACTURER VARCHAR2(50),

CAR\_MODEL VARCHAR2(50),

CAR\_MANUFACTURING\_YEAR DATE,

CAR\_COLOUR VARCHAR2(50),

EQUIPMENT\_PACKAGE\_ID NUMBER(10),

CONSTRAINT PK\_REG\_PLATE\_CARS PRIMARY KEY (REG\_PLATE),

CONSTRAINT FK\_REG\_PLATE\_CARS

FOREIGN KEY (REG\_PLATE)

REFERENCES CARS\_REG\_PLATES(REG\_PLATE),

FOREIGN KEY (EQUIPMENT\_PACKAGE\_ID)

REFERENCES VEHICLE\_EQUIPMENT(PACKAGE\_ID)

);3.1

CREATE OR REPLACE PROCEDURE PROCEDURE\_3\_1

(

P\_NAME IN EMPLOYEES.NAME%TYPE,

P\_SURNAME IN EMPLOYEES.SURNAME%TYPE,

P\_STATUS OUT VARCHAR2,

P\_MSG OUT VARCHAR2

) AS

--exception\_n EXCEPTION;

BEGIN

INSERT INTO EMPLOYEES ("EMPLOYEE\_ID","NAME","SURNAME" )

SELECT ID\_GENERATOR2.NEXTVAL, p\_name,p\_surname FROM dual

WHERE NOT EXISTS

(

SELECT NULL

FROM EMPLOYEES

WHERE "NAME" = P\_NAME AND "SURNAME" = P\_SURNAME

);

EXCEPTION

WHEN OTHERS THEN

P\_STATUS := 'ERR';

P\_MSG := SQLERRM;

DBMS\_OUTPUT.PUT\_LINE(P\_STATUS ||' - '||P\_MSG);

ROLLBACK;

IF P\_STATUS <> 'ERR'

THEN

P\_STATUS := 'OK';

COMMIT;

END IF;

END PROCEDURE\_3\_1;

--------------------------------------------------------------

create or replace PACKAGE TASK1 AS

procedure Q2;

function Q3 (p\_num1 IN number, p\_num2 IN number)return number;

function Q4recursion(k IN number) return number;

function Q4nonrecursion(p\_kth IN number) return number;

procedure Q5 (p\_name IN TEST.NAME%TYPE, p\_surname IN TEST.SURNAME%TYPE);

procedure Q6 (p\_name IN TEST.NAME%TYPE, p\_surname IN TEST.SURNAME%TYPE);

END TASK1;