

## Оглавление

Подготовка БД .....	2
Создание таблиц БД к экзамену .....	3
Задачки надыбанные с экзамена.....	10
Задачи из легального файла для подготовки (лёгкие (которыми дышат)) .....	18
Задачи из легального файла для подготовки (средние (для реальных пацанов)) .	26
Задачи из легального файла для подготовки (типа сложные (для додиков)).....	36

ТЫК по заголовку

## Подготовка БД

```
-- ALTER SESSION SET nls_date_format='yyyy-mm-dd';
create pluggable database EXAM_2023
  ADMIN USER DYV IDENTIFIED BY oracle
  DEFAULT TABLESPACE EXAM_1
  DATAFILE '/home/oracle/exam/exam1.dbf'
  SIZE 25m AUTOEXTEND ON
  NEXT 1m
  MAXSIZE 100m

FILE_NAME_CONVERT=('/u01/app/oracle/oradata/ORCLCDB/pdbseed/', '/home/oracle/exam/');

alter pluggable database EXAM_2023 open;

GRANT CREATE TABLE, CREATE PROCEDURE, CREATE TRIGGER, CREATE
SEQUENCE, CREATE VIEW, CREATE MATERIALIZED VIEW TO DYV;
GRANT CREATE USER, DROP USER TO DYV;
GRANT CREATE SESSION TO DYV WITH ADMIN OPTION;
GRANT CREATE SYNONYM, CREATE PUBLIC SYNONYM TO DYV;
GRANT ALL PRIVILEGES TO DYV;
ALTER USER DYV quota unlimited on EXAM_1;

alter pluggable database EXAM_2023 close;
drop pluggable database EXAM_2023;
```

## Создание таблиц БД к экзамену

---

--- CREATE TABLE for exam

---

```
DROP TABLE ORDERS CASCADE CONSTRAINTS;
DROP TABLE CUSTOMERS CASCADE CONSTRAINTS;
DROP TABLE SALESREPS CASCADE CONSTRAINTS;
DROP TABLE OFFICES CASCADE CONSTRAINTS;
DROP TABLE PRODUCTS CASCADE CONSTRAINTS;
```

```
CREATE TABLE PRODUCTS
  (MFR_ID CHAR(3) NOT NULL,
   PRODUCT_ID CHAR(5) NOT NULL,
   DESCRIPTION VARCHAR(20) NOT NULL,
   PRICE DECIMAL(9,2) NOT NULL,
   QTY_ON_HAND INTEGER NOT NULL,
   PRIMARY KEY (MFR_ID, PRODUCT_ID));
```

```
CREATE TABLE OFFICES
  (OFFICE INTEGER NOT NULL,
   CITY VARCHAR(15) NOT NULL,
   REGION VARCHAR(10) NOT NULL,
   MGR INTEGER,
   TARGET DECIMAL(9,2),
   SALES DECIMAL(9,2) NOT NULL,
   PRIMARY KEY (OFFICE));
```

```
CREATE TABLE SALESREPS
  (EMPL_NUM INTEGER NOT NULL,
   CHECK (EMPL_NUM BETWEEN 101 AND 199),
   NAME VARCHAR(15) NOT NULL,
   AGE INTEGER,
   REP_OFFICE INTEGER,
   TITLE VARCHAR(10),
   HIRE_DATE DATE NOT NULL,
   MANAGER INTEGER,
   QUOTA DECIMAL(9,2),
   SALES DECIMAL(9,2) NOT NULL,
   PRIMARY KEY (EMPL_NUM),
```

```
FOREIGN KEY (MANAGER)
REFERENCES SALESREPS(EMPL_NUM)
ON DELETE SET NULL,
CONSTRAINT WORKSIN FOREIGN KEY (REP_OFFICE)
REFERENCES OFFICES(OFFICE)
ON DELETE SET NULL);
```

```
CREATE TABLE CUSTOMERS
(CUST_NUM INTEGER NOT NULL,
COMPANY VARCHAR(20) NOT NULL,
CUST_REP INTEGER,
CREDIT_LIMIT DECIMAL(9,2),
PRIMARY KEY (CUST_NUM),
CONSTRAINT HASREP FOREIGN KEY (CUST_REP)
REFERENCES SALESREPS(EMPL_NUM)
ON DELETE SET NULL);
```

```
CREATE TABLE ORDERS
(ORDER_NUM INTEGER NOT NULL,
CHECK (ORDER_NUM > 100000),
ORDER_DATE DATE NOT NULL,
CUST INTEGER NOT NULL,
REP INTEGER,
MFR CHAR(3) NOT NULL,
PRODUCT CHAR(5) NOT NULL,
QTY INTEGER NOT NULL,
AMOUNT DECIMAL(9,2) NOT NULL,
PRIMARY KEY (ORDER_NUM),
CONSTRAINT PLACEDBY FOREIGN KEY (CUST)
REFERENCES CUSTOMERS(CUST_NUM)
ON DELETE CASCADE,
CONSTRAINT TAKENBY FOREIGN KEY (REP)
REFERENCES SALESREPS(EMPL_NUM)
ON DELETE SET NULL,
CONSTRAINT ISFOR FOREIGN KEY (MFR, PRODUCT)
REFERENCES PRODUCTS(MFR_ID, PRODUCT_ID));
```

```
ALTER TABLE OFFICES
ADD CONSTRAINT HASMGR
FOREIGN KEY (MGR) REFERENCES SALESREPS(EMPL_NUM)
```

```
ON DELETE SET NULL;
```

```
---
```

```
--- Inserts for sample schema, tailored for Oracle
```

```
---
```

```
alter session set NLS_DATE_FORMAT='YYYY-MM-DD';
```

```
set escape on
```

```
delete from orders;
```

```
delete from customers;
```

```
update offices set mgr=null;
```

```
delete from salesreps;
```

```
delete from offices;
```

```
delete from products;
```

```
commit;
```

```
---
```

```
--- PRODUCTS
```

```
---
```

```
INSERT INTO PRODUCTS VALUES('REI','2A45C','Ratchet Link',79.00,210);
```

```
INSERT INTO PRODUCTS VALUES('ACI','4100Y','Widget  
Remover',2750.00,25);
```

```
INSERT INTO PRODUCTS VALUES('QSA','XK47 ','Reducer',355.00,38);
```

```
INSERT INTO PRODUCTS VALUES('BIC','41627','Plate',180.00,0);
```

```
INSERT INTO PRODUCTS VALUES('IMM','779C ','900-LB Brace',1875.00,9);
```

```
INSERT INTO PRODUCTS VALUES('ACI','41003','Size 3  
Widget',107.00,207);
```

```
INSERT INTO PRODUCTS VALUES('ACI','41004','Size 4  
Widget',117.00,139);
```

```
INSERT INTO PRODUCTS VALUES('BIC','41003','Handle',652.00,3);
```

```
INSERT INTO PRODUCTS VALUES('IMM','887P ','Brace Pin',250.00,24);
```

```
INSERT INTO PRODUCTS VALUES('QSA','XK48 ','Reducer',134.00,203);
```

```
INSERT INTO PRODUCTS VALUES('REI','2A44L','Left Hinge',4500.00,12);
```

```
INSERT INTO PRODUCTS VALUES('FEA','112 ','Housing',148.00,115);
```

```
INSERT INTO PRODUCTS VALUES('IMM','887H ','Brace Holder',54.00,223);
```

```
INSERT INTO PRODUCTS VALUES('BIC','41089','Retainer',225.00,78);
```

```
INSERT INTO PRODUCTS VALUES('ACI','41001','Size 1 Wiget',55.00,277);
```

```
INSERT INTO PRODUCTS VALUES('IMM','775C ','500-lb Brace',1425.00,5);
```

```
INSERT INTO PRODUCTS VALUES('ACI','4100Z','Widget  
Installer',2500.00,28);
```

```
INSERT INTO PRODUCTS VALUES('QSA','XK48A','Reducer',177.00,37);
```

```
INSERT INTO PRODUCTS VALUES('ACI','41002','Size 2 Widget',76.00,167);
```

```

INSERT INTO PRODUCTS VALUES('REI','2A44R','Right Hinge',4500.00,12);
INSERT INTO PRODUCTS VALUES('IMM','773C ','300-lb Brace',975.00,28);
INSERT      INTO      PRODUCTS      VALUES('ACI','4100X','Widget
Adjuster',25.00,37);
INSERT INTO PRODUCTS VALUES('FEA','114  ','Motor Mount',243.00,15);
INSERT      INTO      PRODUCTS      VALUES('IMM','887X      ','Brace
Retainer',475.00,32);
INSERT INTO PRODUCTS VALUES('REI','2A44G','Hinge Pin',350.00,14);
commit;

```

```

---
---  OFFICES
---

```

```

INSERT                                INTO                                OFFICES
VALUES(22,'Denver','Western',null,300000.00,186042.00);
INSERT      INTO      OFFICES      VALUES(11,'New
York','Eastern',null,575000.00,692637.00);
INSERT                                INTO                                OFFICES
VALUES(12,'Chicago','Eastern',null,800000.00,735042.00);
INSERT                                INTO                                OFFICES
VALUES(13,'Atlanta','Eastern',null,350000.00,367911.00);
INSERT      INTO      OFFICES      VALUES(21,'Los
Angeles','Western',null,725000.00,835915.00);
commit;

```

```

---
---  SALESREPS
---

```

```

INSERT INTO SALESREPS VALUES (106,'Sam Clark',52,11,'VP Sales','2006-
06-14',null,275000.00,299912.00);
INSERT INTO SALESREPS VALUES (109,'Mary Jones',31,11,'Sales
Rep','2007-10-12',106,300000.00,392725.00);
INSERT INTO SALESREPS VALUES (104,'Bob Smith',33,12,'Sales
Mgr','2005-05-19',106,200000.00,142594.00);
INSERT INTO SALESREPS VALUES (108,'Larry Fitch',62,21,'Sales
Mgr','2007-10-12',106,350000.00,361865.00);
INSERT INTO SALESREPS VALUES (105,'Bill Adams',37,13,'Sales
Rep','2006-02-12',104,350000.00,367911.00);
INSERT INTO SALESREPS VALUES (102,'Sue Smith',48,21,'Sales
Rep','2004-12-10',108,350000.00,474050.00);
INSERT INTO SALESREPS VALUES (101,'Dan Roberts',45,12,'Sales
Rep','2004-10-20',104,300000.00,305673.00);

```

```
INSERT INTO SALESREPS VALUES (110,'Tom Snyder',41,null,'Sales Rep','2008-01-13',101,null,75985.00);
INSERT INTO SALESREPS VALUES (103,'Paul Cruz',29,12,'Sales Rep','2005-03-01',104,275000.00,286775.00);
INSERT INTO SALESREPS VALUES (107,'Nancy Angelli',49,22,'Sales Rep','2006-11-14',108,300000.00,186042.00);
commit;
```

---

--- OFFICE MANAGERS

---

```
UPDATE OFFICES SET MGR=108 WHERE OFFICE=22;
UPDATE OFFICES SET MGR=106 WHERE OFFICE=11;
UPDATE OFFICES SET MGR=104 WHERE OFFICE=12;
UPDATE OFFICES SET MGR=105 WHERE OFFICE=13;
UPDATE OFFICES SET MGR=108 WHERE OFFICE=21;
commit;
```

---

--- CUSTOMERS

---

```
INSERT INTO CUSTOMERS VALUES(2111,'JCP Inc.',103,50000.00);
INSERT INTO CUSTOMERS VALUES(2102,'First Corp.',101,65000.00);
INSERT INTO CUSTOMERS VALUES(2103,'Acme Mfg.',105,50000.00);
INSERT INTO CUSTOMERS VALUES(2123,'Carter \& Sons',102,40000.00);
INSERT INTO CUSTOMERS VALUES(2107,'Ace International',110,35000.00);
INSERT INTO CUSTOMERS VALUES(2115,'Smithson Corp.',101,20000.00);
INSERT INTO CUSTOMERS VALUES(2101,'Jones Mfg.',106,65000.00);
INSERT INTO CUSTOMERS VALUES(2112,'Zetacorp',108,50000.00);
INSERT INTO CUSTOMERS VALUES(2121,'QMA Assoc.',103,45000.00);
INSERT INTO CUSTOMERS VALUES(2114,'Orion Corp.',102,20000.00);
INSERT INTO CUSTOMERS VALUES(2124,'Peter Brothers',107,40000.00);
INSERT INTO CUSTOMERS VALUES(2108,'Holm \& Landis',109,55000.00);
INSERT INTO CUSTOMERS VALUES(2117,'J.P. Sinclair',106,35000.00);
INSERT INTO CUSTOMERS VALUES(2122,'Three Way Lines',105,30000.00);
INSERT INTO CUSTOMERS VALUES(2120,'Rico Enterprises',102,50000.00);
INSERT INTO CUSTOMERS VALUES(2106,'Fred Lewis Corp.',102,65000.00);
INSERT INTO CUSTOMERS VALUES(2119,'Solomon Inc.',109,25000.00);
INSERT INTO CUSTOMERS VALUES(2118,'Midwest Systems',108,60000.00);
INSERT INTO CUSTOMERS VALUES(2113,'Ian \& Schmidt',104,20000.00);
INSERT INTO CUSTOMERS VALUES(2109,'Chen Associates',103,25000.00);
INSERT INTO CUSTOMERS VALUES(2105,'AAA Investments',101,45000.00);
commit;
```

---

--- ORDERS

---

INSERT	INTO	ORDERS	VALUES	(112961, '2007-12-
17',	2117,106,	'REI', '2A44L',	7,31500.00);	
INSERT	INTO	ORDERS	VALUES	(113012, '2008-01-
11',	2111,105,	'ACI', '41003',	35,3745.00);	
INSERT	INTO	ORDERS	VALUES	(112989, '2008-01-
03',	2101,106,	'FEA', '114',	6,1458.00);	
INSERT	INTO	ORDERS	VALUES	(113051, '2008-02-
10',	2118,108,	'QSA', 'XK47',	4,1420.00);	
INSERT	INTO	ORDERS	VALUES	(112968, '2007-10-
12',	2102,101,	'ACI', '41004',	34,3978.00);	
INSERT	INTO	ORDERS	VALUES	(113036, '2008-01-
30',	2107,110,	'ACI', '4100Z',	9,22500.00);	
INSERT	INTO	ORDERS	VALUES	(113045, '2008-02-
02',	2112,108,	'REI', '2A44R',	10,45000.00);	
INSERT	INTO	ORDERS	VALUES	(112963, '2007-12-
17',	2103,105,	'ACI', '41004',	28,3276.00);	
INSERT	INTO	ORDERS	VALUES	(113013, '2008-01-
14',	2118,108,	'BIC', '41003',	1,652.00);	
INSERT	INTO	ORDERS	VALUES	(113058, '2008-02-
23',	2108,109,	'FEA', '112',	10,1480.00);	
INSERT	INTO	ORDERS	VALUES	(112997, '2008-01-
08',	2124,107,	'BIC', '41003',	1,652.00);	
INSERT	INTO	ORDERS	VALUES	(112983, '2007-12-
27',	2103,105,	'ACI', '41004',	6,702.00);	
INSERT	INTO	ORDERS	VALUES	(113024, '2008-01-
20',	2114,108,	'QSA', 'XK47',	20,7100.00);	
INSERT	INTO	ORDERS	VALUES	(113062, '2008-02-
24',	2124,107,	'FEA', '114',	10,2430.00);	
INSERT	INTO	ORDERS	VALUES	(112979, '2007-10-
12',	2114,102,	'ACI', '4100Z',	6,15000.00);	
INSERT	INTO	ORDERS	VALUES	(113027, '2008-01-
22',	2103,105,	'ACI', '41002',	54,4104.00);	
INSERT	INTO	ORDERS	VALUES	(113007, '2008-01-
08',	2112,108,	'IMM', '773C',	3,2925.00);	
INSERT	INTO	ORDERS	VALUES	(113069, '2008-03-
02',	2109,107,	'IMM', '775C',	22,31350.00);	
INSERT	INTO	ORDERS	VALUES	(113034, '2008-01-
29',	2107,110,	'REI', '2A45C',	8,632.00);	



INSERT	INTO	ORDERS	VALUES	(112992,'2007-11-
04',2118,108,'ACI','41002',10,760.00);				
INSERT	INTO	ORDERS	VALUES	(112975,'2007-10-
12',2111,103,'REI','2A44G',6,2100.00);				
INSERT	INTO	ORDERS	VALUES	(113055,'2008-02-
15',2108,101,'ACI','4100X',6,150.00);				
INSERT	INTO	ORDERS	VALUES	(113048,'2008-02-
10',2120,102,'IMM','779C',2,3750.00);				
INSERT	INTO	ORDERS	VALUES	(112993,'2007-01-
04',2106,102,'REI','2A45C',24,1896.00);				
INSERT	INTO	ORDERS	VALUES	(113065,'2008-02-
27',2106,102,'QSA','XK47',6,2130.00);				
INSERT	INTO	ORDERS	VALUES	(113003,'2008-01-
25',2108,109,'IMM','779C',3,5625.00);				
INSERT	INTO	ORDERS	VALUES	(113049,'2008-02-
10',2118,108,'QSA','XK47',2,776.00);				
INSERT	INTO	ORDERS	VALUES	(112987,'2007-12-
31',2103,105,'ACI','4100Y',11,27500.00);				
INSERT	INTO	ORDERS	VALUES	(113057,'2008-02-
18',2111,103,'ACI','4100X',24,600.00);				
INSERT	INTO	ORDERS	VALUES	(113042,'2008-02-
20',2113,101,'REI','2A44R',5,22500.00);				
commit;				

## Задачи наданные с экзамена

-- 6-7 -- Создать процедуру, которая выведет сотрудников, которые сделали заказ в определённый период и отсортировать по цене заказа

```
CREATE OR REPLACE PROCEDURE salesrepsWhoDidOrders(dateFrom date,
dateTo date)
```

```
as
```

```
    type c_type is ref cursor;
```

```
    c_list c_type;
```

```
    t_name SALESREPS.NAME%type;
```

```
begin
```

```
    open c_list for
```

```
    'SELECT DISTINCT nameAndAmount.NAME FROM
```

```
    (
```

```
    SELECT s.NAME, o.AMOUNT
```

```
        FROM SALESREPS s
```

```
        INNER JOIN ORDERS o
```

```
            ON o.REP = s.EMPL_NUM
```

```
        WHERE o.ORDER_DATE BETWEEN :f AND :t
```

```
        ORDER BY o.AMOUNT
```

```
    ) nameAndAmount'
```

```
    using dateFrom, dateTo;
```

```
    loop
```

```
        fetch c_list into t_name;
```

```
        exit when c_list%notfound;
```

```
        dbms_output.put_line(t_name);
```

```
    end loop;
```

```
end;
```

```
SELECT * FROM ORDERS;
```

```
begin
```

```
    salesrepsWhoDidOrders(dateFrom => to_date('2008-01-01'), dateTo
=> to_date('2008-01-08'));
```

```
end;
```

-- 6-7 -- Создать функцию, которая выведет сотрудников, у которых есть заказы в определенный период

```
CREATE OR REPLACE FUNCTION funcRepsWhoHadOrders(dateFrom date, dateTo date)
```

```
return nvarchar2 is
```

```
    t_names_answer VARCHAR2(2000);
```

```
    t_name SALESREPS.NAME%type;
```

```
    cursor c_names (df date, dt date) is
```

```
        SELECT s.NAME
```

```
            FROM ORDERS o
```

```
            INNER JOIN SALESREPS s
```

```
                ON o.REP = s.EMPL_NUM
```

```
            WHERE o.ORDER_DATE BETWEEN df AND dt;
```

```
begin
```

```
    open c_names(dateFrom, dateTo);
```

```
    fetch c_names into t_name;
```

```
    while (c_names%found)
```

```
    loop
```

```
        t_names_answer := concat(' ', t_names_answer);
```

```
        t_names_answer := concat(t_name, t_names_answer);
```

```
        fetch c_names into t_name;
```

```
    end loop;
```

```
    return t_names_answer;
```

```
end;
```

```
SELECT * FROM ORDERS;
```

```
begin
```

```
    dbms_output.put_line(funcRepsWhoHadOrders(dateFrom  
to_date('2008-01-01'), dateTo => to_date('2008-01-08')));  
end;
```

=>

-- 6-7 -- Написать функцию, которая считает количество заказов за определенный период

```
CREATE OR REPLACE FUNCTION funcHowMuchOrders(dateFrom date, dateTo date)
```

```
return number is
```

```
    n_count number(5,0);
```

```
begin
```

```
    SELECT COUNT(*) INTO n_count
```

```
        FROM ORDERS
```

```
        WHERE ORDER_DATE BETWEEN dateFrom AND dateTo;
```

```
    return n_count;
```

```
end;
```

```
SELECT COUNT(*) FROM ORDERS;
```

```
begin
```

```
    dbms_output.put_line('Count          of          orders:
```

```
' || funcHowMuchOrders(to_date('2008-01-01'), to_date('2008-01-11')));
```

```
end;
```

-- 6-7 -- Процедура увеличения стоимости товара на 10% по названию товара .  
Обработать ошибки

```
CREATE      OR      REPLACE      PROCEDURE      procUpPriceByTen(prName
PRODUCTS.PRODUCT_ID%type)
is
begin
    UPDATE PRODUCTS SET price = price * 1.1 WHERE product_id = prName;
    commit;
    dbms_output.put_line('UPDATE succsessfull!');
exception
    when others
        then
            rollback;
            dbms_output.put_line('procUpPriceByTen          ERROR:
'||sqlerrm);
end;

SELECT * FROM PRODUCTS;

begin
    procUpPriceByTen('2A45C');
end;
```

-- 6-7 – Создайте процедуру, которая выводит N самых молодых сотрудников определенного офиса. Параметр – код офиса. Обработайте возможные ошибки.

```
set serveroutput on;
create or replace procedure taskk(kod offices.office%type, n
number) as
cursor porders is select salesreps.name, salesreps.age
from salesreps
join offices on offices.office = salesreps.rep_office
where offices.office = kod and rownum <= n
order by salesreps.age;
porder porders%rowtype;
exc exception;
found number := 0;
cursor check_cursor is select 1 from offices where offices.office =
kod;
begin
    open porders;
    loop
        fetch porders into porder;
        exit when porders%notfound;
        dbms_output.put_line(porder.name || porder.age);
    end loop;
    close porders;
    open check_cursor;
    fetch check_cursor into found;
    close check_cursor;
    if found = 0 then raise exc;
    end if;
    exception
        when exc
            then dbms_output.put_line('нет такого офиса');
            raise_application_error(-20001, 'exc');
        when others then
            dbms_output.put_line(sqlerrm);
            dbms_output.put_line(sqlcode);
end taskk;

begin
taskk(11,2);
taskk(1001, 2);
end;
```

-- 6-7 – Процедура, которая удаляет продукты, цена которых выше, чем цена, введенная в качестве параметра.

```
create or replace procedure printDelete(pprice products.price%type)
is
cursor c_cur is select o.product, p.price from orders o inner join
products p on o.product=p.product_id order by price desc;
wrongPrice exception;
begin
if pprice < 0 then raise wrongPrice;
end if;
for orde in c_cur loop
dbms_output.put_line('Product: ' || orde.product || ' Price: ' ||
orde.price);
end loop;
dbms_output.put_line('=====');
delete products where price > pprice;
for orde in c_cur loop
dbms_output.put_line('Product: ' ||orde.product || ' Price: '
||orde.price);
end loop;
exception
when wrongPrice then dbms_output.put_line('Wrong price');
when others then dbms_output.put_line(sqlerrm || ' ' ||sqlcode);
end;

begin
printdelete(4000);
end;
```

-- 9-10 -- Процедура, которая выдаёт привилегию пользователю. Привилегия и пользователь передаются в качестве параметров

```
CREATE OR REPLACE PROCEDURE procGivePrivToUser(priv varchar2, usr
varchar2)
```

```
is
    t_grant varchar2(300) := 'GRANT ';
begin
    t_grant := concat(t_grant, priv);
    t_grant := concat(t_grant, ' TO ');
    t_grant := concat(t_grant, usr);
    --t_grant := concat(t_grant, ';');
    execute immediate t_grant;
    commit;
    DBMS_OUTPUT.PUT_LINE(t_grant||' : success!');
exception
    when others
        then rollback;
        DBMS_OUTPUT.PUT_LINE(t_grant||' : ERROR: '||sqlerrm);
end;
```

```
create user JOJO_ref identified by oracle;
-- create connection - it wont work cause no priv!
begin
    procGivePrivToUser('CREATE SESSION','JOJO_ref');
end;
-- now connect as new user - it will be successful!
SELECT * FROM USER_SYS_PRIVS;
drop user JOJO_ref;
```



Вот моя была на экзамене, вкунятина, нормально так поела

Создать функцию, которая возвращает среднюю цену приобретенных товаров по части названия компании покупателя. Если такой компании нет, возвращается -1. Если таких компаний несколько, то возвращается -100. Предусмотреть выдачу сообщений при ошибках.

Еще что-то, может, уже решено в этом файле

Создайте процедуру, которая удалит продавцов, стоимость заказов которых меньше определенного значения, и выведет список оставшихся. Параметр – величина стоимости заказа. Обработайте возможные ошибки.

Создайте функцию, которая подсчитывает количество продавцов определенного товара. Параметры – наименование товара, производитель. Обработайте возможные ошибки.

## Задачи из легального файла для подготовки (лёгкие (которыми дышат))

---- 1. Определите размеры областей памяти SGA

```
SELECT component, current_size FROM v$sga_dynamic_components;
```

---- 2. Получите список всех параметров экземпляра

```
SELECT * FROM v$instance;
```

---- 3. Получите список управляющих файлов

```
SELECT name FROM v$controlfile;
```

---- 4. Сформируйте PFILE

```
-- To find path: SHOW PARAMETER SPFILE
```

```
CREATE PFILE = 'EXAM_2023_PFILE.ORA' FROM SPFILE;
```

---- 5. Создайте таблицу из двух столбцов, один из которых первичный ключ. Получите перечень всех сегментов. Вставьте данные в таблицу. Определите, сколько в сегменте таблицы экстенгов, их размер в блоках и байтах.

```
CREATE TABLE easy_5 (  
    x number(4,0) PRIMARY KEY,  
    y char(1000)  
);  
SELECT * FROM user_segments WHERE SEGMENT_NAME = 'EASY_5';  
INSERT INTO easy_5 values (1, '1');  
INSERT INTO easy_5 values (3, '3');  
INSERT INTO easy_5 values (5, '5');  
INSERT INTO easy_5 values (2, '1');  
INSERT INTO easy_5 values (4, '3');  
INSERT INTO easy_5 values (6, '5');  
INSERT INTO easy_5 values (11, '1');  
INSERT INTO easy_5 values (13, '3');  
INSERT INTO easy_5 values (15, '5');  
INSERT INTO easy_5 values (12, '1');  
INSERT INTO easy_5 values (14, '3');  
INSERT INTO easy_5 values (16, '5');  
INSERT INTO easy_5 values (21, '1');  
INSERT INTO easy_5 values (23, '3');  
INSERT INTO easy_5 values (25, '5');  
INSERT INTO easy_5 values (22, '1');  
INSERT INTO easy_5 values (24, '3');  
INSERT INTO easy_5 values (26, '5');  
INSERT INTO easy_5 values (211, '1');
```

```
INSERT INTO easy_5 values (213, '3');
INSERT INTO easy_5 values (215, '5');
INSERT INTO easy_5 values (212, '1');
INSERT INTO easy_5 values (214, '3');
INSERT INTO easy_5 values (216, '5');
INSERT INTO easy_5 values (31, '1');
INSERT INTO easy_5 values (33, '3');
INSERT INTO easy_5 values (35, '5');
INSERT INTO easy_5 values (32, '1');
INSERT INTO easy_5 values (34, '3');
INSERT INTO easy_5 values (36, '5');
INSERT INTO easy_5 values (311, '1');
INSERT INTO easy_5 values (313, '3');
INSERT INTO easy_5 values (315, '5');
INSERT INTO easy_5 values (312, '1');
INSERT INTO easy_5 values (314, '3');
INSERT INTO easy_5 values (316, '5');
INSERT INTO easy_5 values (321, '1');
INSERT INTO easy_5 values (323, '3');
INSERT INTO easy_5 values (325, '5');
INSERT INTO easy_5 values (322, '1');
INSERT INTO easy_5 values (324, '3');
INSERT INTO easy_5 values (326, '5');
INSERT INTO easy_5 values (3211, '1');
INSERT INTO easy_5 values (3213, '3');
INSERT INTO easy_5 values (3215, '5');
INSERT INTO easy_5 values (3212, '1');
INSERT INTO easy_5 values (3214, '3');
INSERT INTO easy_5 values (3216, '5');
INSERT INTO easy_5 values (41, '1');
INSERT INTO easy_5 values (43, '3');
INSERT INTO easy_5 values (45, '5');
INSERT INTO easy_5 values (42, '1');
INSERT INTO easy_5 values (44, '3');
INSERT INTO easy_5 values (46, '5');
INSERT INTO easy_5 values (411, '1');
INSERT INTO easy_5 values (413, '3');
INSERT INTO easy_5 values (415, '5');
INSERT INTO easy_5 values (412, '1');
INSERT INTO easy_5 values (414, '3');
INSERT INTO easy_5 values (416, '5');
INSERT INTO easy_5 values (421, '1');
INSERT INTO easy_5 values (423, '3');
```

```

INSERT INTO easy_5 values (425, '5');
INSERT INTO easy_5 values (422, '1');
INSERT INTO easy_5 values (424, '3');
INSERT INTO easy_5 values (426, '5');
INSERT INTO easy_5 values (4211, '1');
INSERT INTO easy_5 values (4213, '3');
INSERT INTO easy_5 values (4215, '5');
INSERT INTO easy_5 values (4212, '1');
INSERT INTO easy_5 values (4214, '3');
INSERT INTO easy_5 values (4216, '5');
INSERT INTO easy_5 values (431, '1');
INSERT INTO easy_5 values (433, '3');
INSERT INTO easy_5 values (435, '5');
INSERT INTO easy_5 values (432, '1');
INSERT INTO easy_5 values (434, '3');
INSERT INTO easy_5 values (436, '5');
INSERT INTO easy_5 values (4311, '1');
INSERT INTO easy_5 values (4313, '3');
INSERT INTO easy_5 values (4315, '5');
INSERT INTO easy_5 values (4312, '1');
INSERT INTO easy_5 values (4314, '3');
INSERT INTO easy_5 values (4316, '5');
INSERT INTO easy_5 values (4321, '1');
INSERT INTO easy_5 values (4323, '3');
INSERT INTO easy_5 values (4325, '5');
INSERT INTO easy_5 values (4322, '1');
INSERT INTO easy_5 values (4324, '3');
INSERT INTO easy_5 values (4326, '5');
INSERT INTO easy_5 values (4411, '1');
INSERT INTO easy_5 values (4413, '3');
INSERT INTO easy_5 values (4415, '5');
INSERT INTO easy_5 values (4412, '1');
INSERT INTO easy_5 values (4414, '3');
INSERT INTO easy_5 values (4416, '5');
SELECT SEGMENT_NAME, EXTENTS, BLOCKS/EXTENTS, BYTES/EXTENTS FROM
user_segments WHERE SEGMENT_NAME = 'EASY_5';
DELETE easy_5;
DROP TABLE easy_5;

```

---- 6. Получите перечень всех процессов СУБД Oracle. Для серверных процессов укажите режим подключения. Для фоновых укажите работающие в настоящий момент

```
--all server procs
SELECT * FROM v$process;
```

```
--server field is режим подключения
SELECT process, status, username, schemaname, type, server
  FROM v$session
  ORDER BY PROCESS;
--two ways to find active background processes. To find all
bgprocesses, use SELECT * FROM v$bgprocess
SELECT * FROM v$bgprocess WHERE hextraw('00') != paddr;
SELECT * FROM v$process u JOIN v$bgprocess d ON u.addr = d.paddr ORDER
BY ADDR;
```

---- 7. Получите перечень всех табличных пространств и их файлов

```
SELECT TABLESPACE_NAME, FILE_NAME FROM dba_data_files;
```

---- 8. Получите перечень всех ролей

```
SELECT * FROM DBA_ROLES; -- or SELECT ROLE FROM DBA_ROLES;
```

---- 9. Получите перечень привилегий для определенной роли

```
SELECT * FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'SYS';
```

----10. Получите перечень всех пользователей

```
SELECT * FROM DBA_USERS;
```

----11. Создайте роль

```
CREATE ROLE exam_2023_role;
GRANT CREATE SESSION TO exam_2023_role;
--DROP ROLE exam_2023_role;
```

----12. Создайте пользователя

```
CREATE USER exam_2023_user identified by password
  --DEFAULT TABLESPACE EXAM_1
  PROFILE exam_2023_profile
  ACCOUNT UNLOCK;
GRANT exam_2023_role TO exam_2023_user;
--DROP USER exam_2023_user;
```

----13. Получите перечень всех профилей безопасности

```
SELECT DISTINCT PROFILE FROM dba_profiles;
```

----14. Получите перечень всех параметров профиля безопасности

```
SELECT * FROM dba_profiles WHERE PROFILE = 'EXAM_2023_PROFILE';
```

----15. Создайте профиль безопасности

```
CREATE PROFILE exam_2023_profile LIMIT
```

```
PASSWORD_LIFE_TIME 365
SESSIONS_PER_USER 4
FAILED_LOGIN_ATTEMPTS 4
PASSWORD_LOCK_TIME 1
CONNECT_TIME 120
IDLE_TIME 30;
--DROP PROFILE exam_2023_profile;
```

----16. Создайте последовательность S1, со следующими характеристиками: начальное значение 1000; приращение 10; минимальное значение 0; максимальное значение 10000; циклическую; кэширующую 30 значений в памяти; гарантирующую хронологию значений. Создайте таблицу T1 с тремя столбцами и введите (INSERT) 10 строк, со значениями из S1.

```
CREATE SEQUENCE S1
  START WITH 1000
  INCREMENT BY 10
  MINVALUE 0
  MAXVALUE 10000
  CYCLE
  CACHE 30
  ORDER;
CREATE TABLE easy_16 (
  val number(5,0)
);
INSERT INTO easy_16 values (S1.nextval);
INSERT INTO easy_16 values (S1.nextval);
INSERT INTO easy_16 values (S1.nextval);
INSERT INTO easy_16 values (S1.nextval);
INSERT INTO easy_16 values (S1.nextval);
SELECT * FROM easy_16;
DELETE easy_16;
DROP TABLE easy_16;
DROP SEQUENCE S1;
```

----17. Создайте частный и публичный синоним для одной из таблиц и продемонстрируйте его область видимости. Найдите созданные синонимы в представлениях словаря Oracle

```
CREATE SYNONYM orc FOR DYV.ORDERS;
CREATE PUBLIC SYNONYM notorc FOR DYV.ORDERS;
SELECT * FROM orc;
SELECT * FROM notorc;
```

----18. Разработайте анонимный блок, демонстрирующий возникновение и обработку исключений WHEN TO\_MANY\_ROWS и NO\_DATA\_FOUND

```
declare
    o_order ORDERS%rowtype;
begin
    SELECT * INTO o_order FROM ORDERS WHERE ORDER_NUM = 0;--comment
this row to achieve TOO_MANY_ROWS
    SELECT * INTO o_order FROM ORDERS WHERE MFR = 'ACI';--comment
both rows to achieve no exceptions
    dbms_output.put_line('No exception');
exception
    when NO_DATA_FOUND
        then dbms_output.put_line('Exception NDF: '||sqlerrm);
    when TOO_MANY_ROWS
        then dbms_output.put_line('Exception TMR: '||sqlerrm);
end;
```

----19. Получите перечень всех файлов групп журналов повтора

```
select GROUP#, MEMBER from v$logfile;
```

----20. Определите текущую группу журналов повтора

```
select GROUP# from v$log WHERE STATUS = 'CURRENT';
```

----21. Получите перечень контрольных файлов

```
SELECT name FROM v$controlfile;
```

----22. Создайте таблицу и вставьте в нее 100 записей. Найдите таблицу и ее свойства в представлениях словаря

```
CREATE TABLE easy_22 ( x number(4,0) );
```

```
declare
```

```
    n_i number(4,0) := 0;
```

```
begin
```

```
    while (n_i < 100)
```

```
    loop
```

```
        INSERT INTO easy_22 VALUES (n_i);
```

```
        n_i := n_i + 1;
```

```
    end loop;
```

```
end;
```

```
SELECT * FROM dictionary WHERE table_name like '%TABLE%';
```

```
SELECT * FROM USER_TABLES WHERE TABLE_NAME = 'EASY_22';
```

----23. Получите список сегментов табличного пространства

```
SELECT * FROM dba_segments WHERE TABLESPACE_NAME = 'EXAM_1';
```

----24. Выведите список всех объектов, доступных пользователю

```
SELECT * FROM all_objects;
```

----25. Вычислите количество блоков, занятых таблицей

```
SELECT OWNER, SEGMENT_NAME, BLOCKS FROM dba_segments WHERE  
SEGMENT_NAME LIKE '%EASY%';
```

----26. Выведите список текущих сессий

```
SELECT * FROM v$session;
```

----27. Выведите, производится ли архивирование журналов повтора

```
SELECT name, log_mode FROM v$database;
```

```
SELECT instance_name, archiver, active_state FROM v$instance;
```

----28. Создайте представление с определенными параметрами

```
CREATE OR REPLACE VIEW exam_2023_view
```

```
as SELECT NAME FROM SALESREPS;
```

```
CREATE MATERIALIZED VIEW exam_2023_mv
```

```
BUILD IMMEDIATE
```

```
REFRESH FORCE
```

```
--ENABLE QUERY REWRITE --GRANT QUERY REWRITE TO DYV
```

```
as SELECT DISTINCT TITLE FROM SALESREPS;
```

```
SELECT * FROM exam_2023_view;
```

```
SELECT * FROM exam_2023_mv;
```

```
DROP VIEW exam_2023_view;
```

```
DROP MATERIALIZED VIEW exam_2023_mv;
```

----29. Создайте database link с определенными параметрами

```
CREATE DATABASE LINK exam_2023_link
```

```
CONNECT TO JOJO_ref
```

```
IDENTIFIED BY oracle
```

```
USING 'localhost:1521/EXAM_2023';
```

```
DROP DATABASE LINK exam_2023_link;
```

----30. Продемонстрируйте эскалацию исключения

```
begin
```

```
begin
```

```
RAISE_APPLICATION_ERROR(-20001, 'Rised error!');
```

```
exception
```



```
        when others
            then dbms_output.put_line(sqlerrm);
                RAISE;--if u whanna raise same error
                RAISE_APPLICATION_ERROR(-20002, 'Rised      rised
error!');--if u whanna raise another code/message
        end;
exception
    when others
        then dbms_output.put_line(sqlerrm);
end;
```

## Задачи из легального файла для подготовки (средние (для реальных пацанов))

---- 1. Создайте процедуру, которая выводит список заказов и их итоговую стоимость для определенного покупателя. Параметр – наименование покупателя. Обработайте возможные ошибки

```
CREATE OR REPLACE PROCEDURE getOrdersAndSumByCustomer(customer_id in
varchar2)
as
    cursor cus is
        SELECT *
        FROM ORDERS ord
        WHERE ord.CUST = customer_id;
    price_sum number := 0;
begin
    for q_order in cus
    loop
        dbms_output.put_line(q_order.ORDER_NUM || ' ' || q_order.ORDER_DATE || ' ' || q_order.AMOUNT);
        price_sum := price_sum + q_order.AMOUNT;
    end loop;
    dbms_output.put_line('price sum: ' || price_sum);
exception
    when NO_DATA_FOUND--wont ever be achieved
    then dbms_output.put_line('no data found');
    when others
    then dbms_output.put_line(sqlerrm);
end;

begin
    GetOrdersAndSumByCustomer(2108);
end;
```

---- 2. Создайте функцию, которая подсчитывает количество заказов покупателя за определенный период. Параметры – покупатель, дата начала периода, дата окончания периода

```
CREATE OR REPLACE FUNCTION funcHowMuchOrdersFromUser(custmr
CUSTOMERS.COMPANY%type,dateFrom date, dateTo date)
return number is
    n_count number(5,0);
begin
    SELECT COUNT(*) INTO n_count
        FROM ORDERS o
        INNER JOIN CUSTOMERS c
            ON o.CUST = c.CUST_NUM
        WHERE ORDER_DATE BETWEEN dateFrom AND dateTo AND COMPANY =
custmr;
    return n_count;
end;

begin
    dbms_output.put_line('Count of orders:
'||funcHowMuchOrdersFromUser('JCP Inc.', to_date('2007-12-01'),
to_date('2008-01-31')));
end;
```

---- 3. Создайте процедуру, которая выводит список всех товаров, приобретенных покупателем, с указанием суммы продаж по возрастанию. Параметр – наименование покупателя. Обработайте возможные ошибки.

```
CREATE OR REPLACE PROCEDURE
getProductsByCustomerOrderByAmount(customer_id in number)
as
    cursor cur is
        SELECT pr.description, sum(ord.amount) as sum
        FROM products pr
        JOIN orders ord
            on pr.product_id = ord.product
        GROUP BY pr.description, ord.cust
        HAVING ord.cust = customer_id
        ORDER BY sum(ord.amount) DESC;
begin
    for q_cur in cur
    loop
        dbms_output.put_line(q_cur.description || ' ' || q_cur.sum);
    end loop;
exception
    when others
        then dbms_output.put_line('GPBCOBA error: '||sqlerrm);
end;

begin
    getProductsByCustomerOrderByAmount(2111);
end;
```

---- 4. Создайте функцию, которая добавляет покупателя в таблицу Customers, и возвращает код добавленного покупателя или -1 в случае ошибки. Параметры соответствуют столбцам таблицы, кроме кода покупателя, который задается при помощи последовательности

```
CREATE SEQUENCE exam_sequence1
  START WITH 1000
  INCREMENT BY 1
  MAXVALUE 2000
  NOCYCLE
  NOCACHE
  ORDER;
```

```
CREATE OR REPLACE FUNCTION AddCustomer(company_name varchar2,
cust_rep_id number, credit_limit_num number)
return number
is
  cust_id number;
begin
  INSERT INTO CUSTOMERS(CUST_NUM, COMPANY, CUST_REP, CREDIT_LIMIT)
  VALUES (exam_sequence1.nextval, company_name, cust_rep_id,
credit_limit_num)
  RETURNING cust_num INTO cust_id;
  return cust_id;
exception
  when others
  then DBMS_OUTPUT.PUT_LINE('AC error: '||sqlerrm);
  return -1;
end;

declare
  retval number;
begin
  retval := AddCustomer('test', 110, 55000);
  dbms_output.put_line('New CUSTOMER: '||retval);
end;
```

---- 5. Создайте процедуру, которая выводит список покупателей, в порядке убывания общей стоимости заказов. Параметры – дата начала периода, дата окончания периода. Обработайте возможные ошибки

```
CREATE OR REPLACE PROCEDURE GetCust(start_date varchar2, end_date
varchar2)
as
    cursor cur is
        SELECT cust.COMPANY, sum(ord.AMOUNT) as price_sum
        FROM customers cust
        INNER JOIN orders ord
            ON ord.CUST = cust.CUST_NUM AND ord.order_date BETWEEN
to_date(start_date) AND to_date(end_date)
        GROUP BY ord.cust, cust.COMPANY
        ORDER BY sum(ord.AMOUNT) DESC;
begin
    for qcur in cur
    loop
        dbms_output.put_line(qcur.COMPANY || ' ' || qcur.price_sum);
    end loop;
exception
    when others
        then dbms_output.put_line('GC error: '||sqlerrm);
end;

begin
    GetCust('2007-04-01', '2008-03-02');
end;
```

---- 6. Создайте функцию, которая подсчитывает количество заказанных товаров за определенный период. Параметры – дата начала периода, дата окончания периода

```
CREATE OR REPLACE FUNCTION funcHowMuchProducts(dateFrom date, dateTo
date)
return number is
    n_count number(5,0);
begin
    SELECT SUM(QTY) INTO n_count
    FROM ORDERS
    WHERE ORDER_DATE BETWEEN dateFrom AND dateTo;
    return n_count;
exception
    when others
        then dbms_output.put_line('FHMP error: '||sqlerrm);
```

```
end;
```

```
SELECT SUM(QTY) FROM ORDERS;
```

```
begin
```

```
    dbms_output.put_line('Count          of          products:  
' || funcHowMuchProducts(to_date('2008-01-01'), to_date('2008-01-  
11')));  
end;
```

---- 7. Создайте процедуру, которая выводит список покупателей, у которых есть заказы в этом временном периоде. Параметры – дата начала периода, дата окончания периода. Обработайте возможные ошибки

```
CREATE OR REPLACE PROCEDURE customersWhoDidOrders(dateFrom date,
dateTo date)
as
    type c_type is ref cursor;
    c_list c_type;
    t_name SALESREPS.NAME%type;
begin
    open c_list for
    'SELECT DISTINCT nameAndAmount.COMPANY FROM
    (
    SELECT s.COMPANY, o.AMOUNT
      FROM CUSTOMERS s
     INNER JOIN ORDERS o
        ON o.CUST = s.CUST_NUM
     WHERE o.ORDER_DATE BETWEEN :f AND :t
     ORDER BY o.AMOUNT
    ) nameAndAmount'
    using dateFrom, dateTo;
    loop
        fetch c_list into t_name;
        exit when c_list%notfound;
        dbms_output.put_line(t_name);
    end loop;
exception
    when others
        then dbms_output.put_line('CWDO error: '||sqlerrm);
end;

begin
    customersWhoDidOrders(dateFrom => to_date('2008-01-01'), dateTo
=> to_date('2008-01-08'));
end;
```



---- 8. Создайте функцию, которая подсчитывает количество покупателей определенного товара. Параметры – наименование товара

```
CREATE OR REPLACE FUNCTION GetAmountCustByProduct(product_id in
varchar2) return number
as
    counter number := 0;
begin
    SELECT COUNT(distinct(ord.CUST)) INTO counter
    FROM orders ord
    WHERE ord.PRODUCT = product_id;
    return counter;
exception
    when others
    then dbms_output.put_line('CWD0 error: '||sqlerrm);
end;

select GetAmountCustByProduct('41004') from dual;
```

---- 9. Создайте процедуру, которая увеличивает на 10% стоимость определенного товара. Параметр – наименование товара. Обработайте возможные ошибки.

```
CREATE OR REPLACE PROCEDURE procUpPriceByTen(prName
PRODUCTS.PRODUCT_ID%type)
is
begin
    UPDATE PRODUCTS SET price = price * 1.1 WHERE product_id = prName;
    commit;
    dbms_output.put_line('UPDATE succsessfull!');
exception
    when others
        then
            rollback;
            dbms_output.put_line('procUpPriceByTen ERROR:
'||sqlerrm);
end;

SELECT * FROM PRODUCTS;

begin
    procUpPriceByTen('2A45C');
end;
```

----10. Создайте функцию, которая вычисляет количество заказов, выполненных в определенном году для определенного покупателя. Параметры – покупатель, год товара.

```
CREATE OR REPLACE FUNCTION GetAmountOrdersByYearAndCust(cust_id in
number, year in number) return number
as
    counter number := 0;
begin
    SELECT COUNT(DISTINCT(ord.CUST)) INTO counter
    FROM orders ord
    WHERE ord.cust = cust_id AND EXTRACT(year from ord.order_date) =
year;
    return counter;
exception
    when others
        then dbms_output.put_line('GAOBYAC error: '||sqlerrm);
end;

begin
    dbms_output.put_line('2114                in                2008:
'||GetAmountOrdersByYearAndCust(2114, 2008));
end;
```

## Задачи из легального файла для подготовки (типа сложные (для додиков))

---- 1. Создайте процедуру, которая добавляет заказ. Обработайте возможные ошибки. Создайте триггер, который контролирует целостность данных при добавлении заказа.

```
CREATE OR REPLACE PROCEDURE addOrder(orderId number, customerId
number, repId number, mfrId char, productId char, qty number, amount
number)
is
begin
    INSERT INTO ORDERS VALUES (orderId, SYSDATE, customerId, repId,
mfrId, productId, qty, amount);
exception
    when others
        then      dbms_output.put_line('Exception      in      addOrder:
'||sqlerrm);
end;
CREATE OR REPLACE TRIGGER addOrderTrigger
    before insert on ORDERS
    for each row
declare
    type c_type is ref cursor;
    c_nullable c_type;
    o_order ORDERS%rowtype;
    o_product PRODUCTS%rowtype;
    o_cust CUSTOMERS%rowtype;
    o_rep SALESREPS%rowtype;
begin
    if (:new.ORDER_NUM is null) or (:new.ORDER_DATE is null) or
(:new.CUST is null) or (:new.MFR is null) or (:new.PRODUCT is null)
or (:new.QTY is null) or (:new.AMOUNT is null)
        then RAISE_APPLICATION_ERROR(-20000, 'NOT NULL exception');
    elsif :new.ORDER_NUM <= 100000
        then  RAISE_APPLICATION_ERROR(-20001,  'ORDER_NUM  is  too
small');
    end if;
    dbms_output.put_line('OK1');
    open c_nullable for 'SELECT * FROM ORDERS WHERE ORDER_NUM = :uno'
using :new.ORDER_NUM;
    fetch c_nullable into o_order;
    if c_nullable%found
```

```

        then RAISE_APPLICATION_ERROR(-20002, 'ORDER_NUM is already
exist');
    end if;
    close c_nullable;
    dbms_output.put_line('OK2');
    open c_nullable for 'SELECT * FROM CUSTOMERS WHERE CUST_NUM =
:uno' using :new.CUST;
    fetch c_nullable into o_cust;
    if c_nullable%notfound
        then RAISE_APPLICATION_ERROR(-20003, 'No such CUSTOMER');
    end if;
    close c_nullable;
    dbms_output.put_line('OK3');
    open c_nullable for 'SELECT * FROM SALESREPS WHERE EMPL_NUM =
:uno' using :new.REP;
    fetch c_nullable into o_rep;
    if c_nullable%notfound
        then RAISE_APPLICATION_ERROR(-20004, 'No such SALESREP');
    end if;
    close c_nullable;
    dbms_output.put_line('OK4');
    open c_nullable for 'SELECT * FROM PRODUCTS WHERE MFR_ID = :uno
AND PRODUCT_ID = :duo' using :new.MFR, :new.PRODUCT;
    fetch c_nullable into o_product;
    if c_nullable%notfound
        then RAISE_APPLICATION_ERROR(-20005, 'No such Product');
    end if;
    close c_nullable;
    dbms_output.put_line('OK5');
    dbms_output.put_line('New order: ' || :new.ORDER_NUM);
end;
begin
    addOrder(888888,2111,110,'ACI','4100Z',9,22500.00);
end;
SELECT * FROM ORDERS WHERE ORDER_NUM = 888888;--check if anon block
succeeded correctly

```

---- 2. Создайте функцию, которая возвращает количество заказов покупателя ежемесячно за определенный период. Параметры – покупатель, дата начала периода, дата окончания периода. Обработайте возможные ошибки

```
CREATE OR REPLACE PACKAGE hatred
as
    type o_record is record ( qty integer, monthNumber number(2,0) );
    type o_table is table of o_record;
    FUNCTION funcNumOrdersByCustomer(cust integer, dateFrom date,
dateTo date) return o_table pipelined;
end;
```

```
CREATE OR REPLACE PACKAGE BODY hatred
is
    FUNCTION funcNumOrdersByCustomer(cust integer, dateFrom date,
dateTo date)
    return o_table pipelined is
        cursor c_table(cust integer, dateFrom date, dateTo date) is
            SELECT COUNT(*), extract(MONTH from ORDER_DATE)
                FROM ORDERS
                WHERE ORDER_DATE BETWEEN dateFrom AND dateTo AND CUST
= cust
                GROUP BY extract(MONTH from ORDER_DATE);
        o_answer o_record;

begin
    for o_answer in c_table(cust, dateFrom, dateTo)
    loop
        PIPE ROW (o_answer);
    end loop;
    return;
exception
    when others
        then dbms_output.put_line('FNOBC error: '||sqlerrm);
end;--func
end;--pack
```

```
declare
    cursor      c_kek      is      SELECT      *      FROM
table(hatred.funcNumOrdersByCustomer(2111,      to_date('2007-01-01'),
to_date('2007-12-31')));
    n_i integer;
    n_n number(2,0);
```

```
begin
  open c_kek;
  loop
    fetch c_kek into n_i, n_n;
    exit WHEN c_kek%notfound;
    dbms_output.put_line('In   '||n_n||'   month   was   '||n_i||'
orders. ');
  end loop;
  close c_kek;
end;
```

---- 3. Создайте процедуру, которая выводит в консоль список всех товаров, не приобретенных ни одним покупателем в определенном году по убыванию количества на складе. Параметр – год. Обработайте возможные ошибки

```
CREATE OR REPLACE PROCEDURE procListNotOrdered(dateYear number)
is
    cursor c_query (yr number) is
        SELECT DISTINCT p.product_id, p.description, p.mfr_id
        FROM products p
        INNER JOIN orders o
            ON p.product_id = o.product AND p.mfr_id = o.mfr
        WHERE extract(year from o.order_date) = yr;
    type o_product_type is record
    (
        product_id products.product_id%type,
        description products.description%type,
        mfr_id products.mfr_id%type
    );
    o_product o_product_type;
begin
    for o_product in c_query(dateYear)
    loop
        dbms_output.put_line(o_product.description||'
'||o_product.product_id||' ('||o_product.mfr_id||')');
    end loop;
exception
    when others
    then dbms_output.put_line('PLNO error: '||sqlerrm);
end;
```

```
begin
    procListNotOrdered(2007);
end;
```

---- 4. Создайте функцию, которая подсчитывает количество заказов покупателя за определенный год. Параметры – год, часть имени покупателя или код

```
CREATE OR REPLACE FUNCTION funcHowMuchOrders(yearAsChar char,
customer varchar2)
return number is
    n_counter number(5,0) := 0;
    t_pattern varchar2(30) := '%';
    n_pattern integer := 0;
begin
```



```

t_pattern := concat(t_pattern, customer);
t_pattern := concat(t_pattern, '%');
begin
    n_pattern := to_number(customer);
exception
    when others
        then n_pattern := 0;
end;
SELECT COUNT(*)
    INTO n_counter
    FROM ORDERS o
    INNER JOIN CUSTOMERS c
        ON o.CUST = c.CUST_NUM
    WHERE (yearAsChar = to_char(extract(year from o.ORDER_DATE)))
AND ((c.CUST_NUM = n_pattern) OR (UPPER(c.COMPANY) LIKE
UPPER(t_pattern)));
    return n_counter;
end;

begin
    dbms_output.put_line('2008,                                2111:
' || funcHowMuchOrders('2008','2111'));
    dbms_output.put_line('2007,                                Orion:
' || funcHowMuchOrders('2007','Orion'));
end;

```

---- 5. Создайте процедуру, которая сортирует таблицу по определенному столбцу  
Параметры – название столбца, порядок сортировки (ASC, DESC). Обработайте  
возможные ошибки.

```
CREATE TABLE hard_5 ( c1 char(2), c2 char(2));
INSERT INTO hard_5 VALUES ('Fi', 'If');
INSERT INTO hard_5 VALUES ('Dd', 'dD');
INSERT INTO hard_5 VALUES ('Qi', 'Iq');
INSERT INTO hard_5 VALUES ('Aq', 'Qa');
INSERT INTO hard_5 VALUES ('Gf', 'Fg');
INSERT INTO hard_5 VALUES ('Aa', 'aA');
INSERT INTO hard_5 VALUES ('aa', 'UU');
INSERT INTO hard_5 VALUES ('Uu', 'AA');
```

```
CREATE OR REPLACE PROCEDURE procOrderTableHard5( columnName char,
orderType varchar2)--COLUMN NAME => DYNAMIC CURSOR
is
    type c_type is ref cursor;
    c_mem c_type;
    o_memrow hard_5%rowtype;
    t_query varchar2(50) := 'SELECT * FROM hard_5 ORDER BY ';
begin
    if ((columnName != 'c1')AND(columnName != 'c2'))
        then RAISE_APPLICATION_ERROR(-20000, 'Incorrect column
name');
    elsif ((orderType != 'ASC')AND(orderType != 'DESC'))
        then RAISE_APPLICATION_ERROR(-20001, 'Incorrect order
type');
    end if;
    t_query := t_query||columnName||' '||orderType;
    open c_mem for t_query;
    DELETE hard_5;
    loop
        fetch c_mem into o_memrow;
        exit when c_mem%notfound;
        INSERT INTO hard_5 VALUES (o_memrow.c1, o_memrow.c2);
    end loop;
    commit;
exception
    when others
        then rollback;
        dbms_output.put_line('POTH5 error: '||sqlerrm);
end;
```

```
SELECT * FROM hard_5;
begin
    procOrderTableHard5('c2','ASC');
end;
SELECT * FROM hard_5;

DELETE hard_5;
DROP TABLE hard_5;
```