**Procedures**

----------------function 1----------

create or replace function get\_count

(v\_branch in supermarket.branch%type)

return number is

v\_person number;

begin

select count(invoice\_id) into v\_person from supermarket

where branch = v\_branch;

return v\_person;

end;

begin

dbms\_output.put\_line(get\_count('A'));

end;

---------------procedure-----------------

create or replace procedure count

(product in sales.product\_line%type,

tax out sales.tax%type) is

begin

select max(tax) into tax from sales

where product\_line = product;

end;

-------------------procedure----------------------

create or replace procedure info\_about(v\_invoice\_id sales.invoice\_id%type, v\_city sales.city%type, v\_customer\_type sales.customer\_type%type, v\_gender sales.gender%type)

is

begin

insert into sales(invoice\_id, city, customer\_type, gender) values (v\_invoice\_id, v\_city, v\_customer\_type, v\_gender);

end info\_about;

-------------procedure----------

CREATE OR REPLACE PROCEDURE get\_from\_branch(v\_branch in sales.branch%type, v\_product\_line out sales.product\_line%type) is

BEGIN

SELECT product\_line, branch into v\_product\_line, v\_branch

FROM sales

WHERE branch = v\_branch;

END;

**Collections**

----------1---------------

create or replace procedure get\_total(inv\_id sales.invoice\_id%type) is

type t\_sup is table of sales%rowtype index by binary\_integer;

v\_super t\_sup;

v\_total sales.total%type;

begin

select \* bulk collect into v\_super from sales

where invoice\_id =inv\_id;

for i in v\_super.first..v\_super.last loop

exit when i>v\_super.last;

dbms\_output.put\_line(v\_super(i).total);

end loop;

end;

**Cursors**

-------1 for ex we want to watch which product lines consists in branch A-----------

DECLARE

CURSOR cur\_product\_line IS

SELECT product\_line FROM sales WHERE branch = 'A'

group by product\_line;

v\_product\_line sales.product\_line%TYPE;

BEGIN

OPEN cur\_product\_line;

LOOP

FETCH cur\_product\_line INTO v\_product\_line;

EXIT WHEN cur\_product\_line%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_product\_line);

END LOOP;

CLOSE cur\_product\_line;

END;

-------------------------2for ex we want to count how many normal and member in branch A---

set SERVEROUTPUT ON;

DECLARE

CURSOR cur\_customer IS

SELECT customer\_type, Count(\*) as count\_type FROM sales WHERE branch = 'A'

group by customer\_type;

v\_customer\_type cur\_customer%ROWTYPE;

BEGIN

OPEN cur\_customer;

LOOP

FETCH cur\_customer INTO v\_customer\_type;

EXIT WHEN cur\_customer%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_customer\_type.customer\_type||' '|| v\_customer\_type.count\_type);

END LOOP;

CLOSE cur\_customer;

END;

-------------------------3for ex we want to find out how many male/female in all rows---

set SERVEROUTPUT ON;

DECLARE

CURSOR cur\_gender IS

SELECT gender, Count(\*) as count\_type FROM sales

group by gender;

v\_gender cur\_gender%ROWTYPE;

BEGIN

OPEN cur\_gender;

LOOP

FETCH cur\_gender INTO v\_gender;

EXIT WHEN cur\_gender%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_gender.gender||' '|| v\_gender.count\_type);

END LOOP;

CLOSE cur\_gender;

END;

---------------4for ex we want to see statistics about payment----

set SERVEROUTPUT ON;

DECLARE

CURSOR cur\_pay IS

SELECT payment, Count(\*) as count\_type FROM sales

group by payment;

v\_payment cur\_pay%ROWTYPE;

BEGIN

OPEN cur\_pay;

LOOP

FETCH cur\_pay INTO v\_payment;

EXIT WHEN cur\_pay%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_payment.payment||' '|| v\_payment.count\_type);

END LOOP;

CLOSE cur\_pay;

END;

**Packages**

**Triggers**

CREATE OR REPLACE TRIGGER

prevent\_drop\_trigg

BEFORE DROP ON SCHEMA

BEGIN

RAISE\_APPLICATION\_ERROR

(-20203, 'Attempted drop – failed');

END;

CREATE OR REPLACE TRIGGER del\_sales\_values

AFTER DELETE ON sales FOR EACH ROW

BEGIN

INSERT INTO sales\_log(USER\_NAME, DATE\_OF\_CHANGES, TYPE\_OF\_CHANGES)

VALUES (USER, SYSDATE, 'DELETION');

END;

CREATE OR REPLACE TRIGGER ins\_sales\_values

AFTER INSERT ON sales FOR EACH ROW

BEGIN

INSERT INTO sales\_log(USER\_NAME, DATE\_OF\_CHANGES, TYPE\_OF\_CHANGES)

VALUES (USER, SYSDATE, 'INSERTING');

END;

CREATE OR REPLACE TRIGGER upd\_sales\_values

AFTER UPDATE ON sales FOR EACH ROW

BEGIN

INSERT INTO sales\_log(USER\_NAME, DATE\_OF\_CHANGES, TYPE\_OF\_CHANGES)

VALUES (USER, SYSDATE, 'UPDATING');

END;

**Dynamic SQL**

-------------1--------------

CREATE PROCEDURE

drop\_any\_column(p\_column\_name VARCHAR2)IS

BEGIN

EXECUTE IMMEDIATE 'DROP COLUMN' || p\_COLUMN\_name;

END;

-------------------2---------------

CREATE or replace PROCEDURE add\_info(p\_table\_name VARCHAR2, v\_invoice\_id sales.invoice\_id%type, v\_city sales.city%type, v\_customer\_type sales.customer\_type%type, v\_gender sales.gender%type) IS

BEGIN

EXECUTE IMMEDIATE 'INSERT INTO ' || p\_table\_name|| '(invoice\_id, city, customer\_type, gender) (VALUES(' || v\_invoice\_id|| ', ''' || v\_city||' , ''' || v\_customer\_type|| ',''' || v\_gender|| ''')';

END;

-------------------3---------------

create or replace procedure get\_total(inv\_id sales.invoice\_id%type) is

type t\_sup is table of sales%rowtype index by binary\_integer;

v\_super t\_sup;

v\_total sales.total%type;

begin

select \* bulk collect into v\_super from sales

where invoice\_id =inv\_id;

for i in v\_super.first..v\_super.last loop

exit when i>v\_super.last;

dbms\_output.put\_line(v\_super(i).total);

end loop;

end;

CREATE OR REPLACE PROCEDURE p\_print (p\_percent NUMBER) IS

TYPE numlist\_type IS TABLE OF supermarket.invoice\_id%type

INDEX BY BINARY\_INTEGER;

v\_id numlist\_type; -- collection

BEGIN

v\_id(1) := '750-67-8428';

v\_id(2) := '226-31-3081';

v\_id(3) := '631-41-3108';

v\_id(4) := '123-19-1176';

-- bulk-bind the associative array

FORALL i IN v\_id.FIRST .. v\_id.LAST

UPDATE supermarket

SET tax = (1 + p\_percent / 100) \* tax

WHERE invoice\_id =v\_id(i);

END;

SELECT tax

FROM supermarket

where

invoice\_id = '750-67-8428' OR invoice\_id = '226-31-3081'

OR invoice\_id = '631-41-3108' OR invoice\_id = '123-19-1176';

BEGIN

p\_print(50);

END;