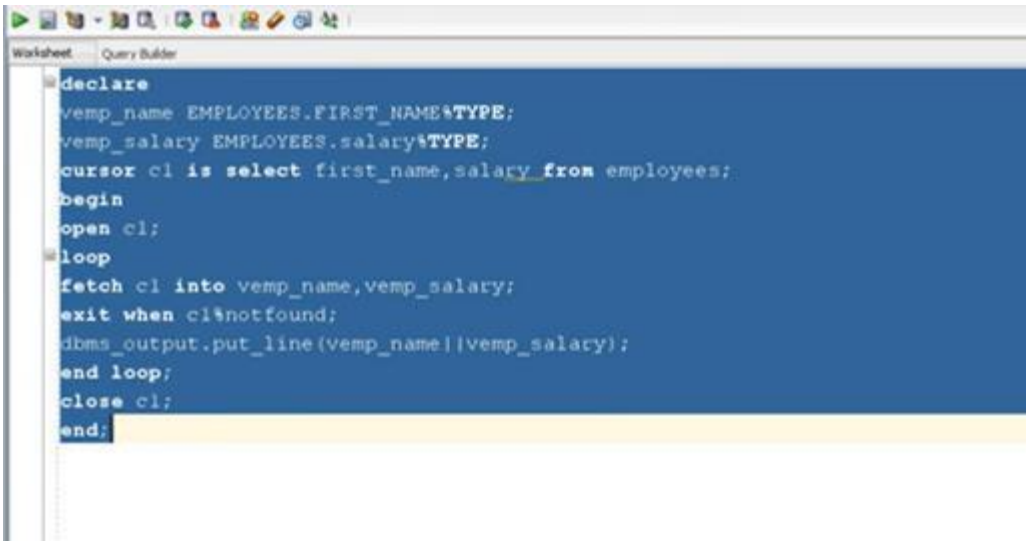
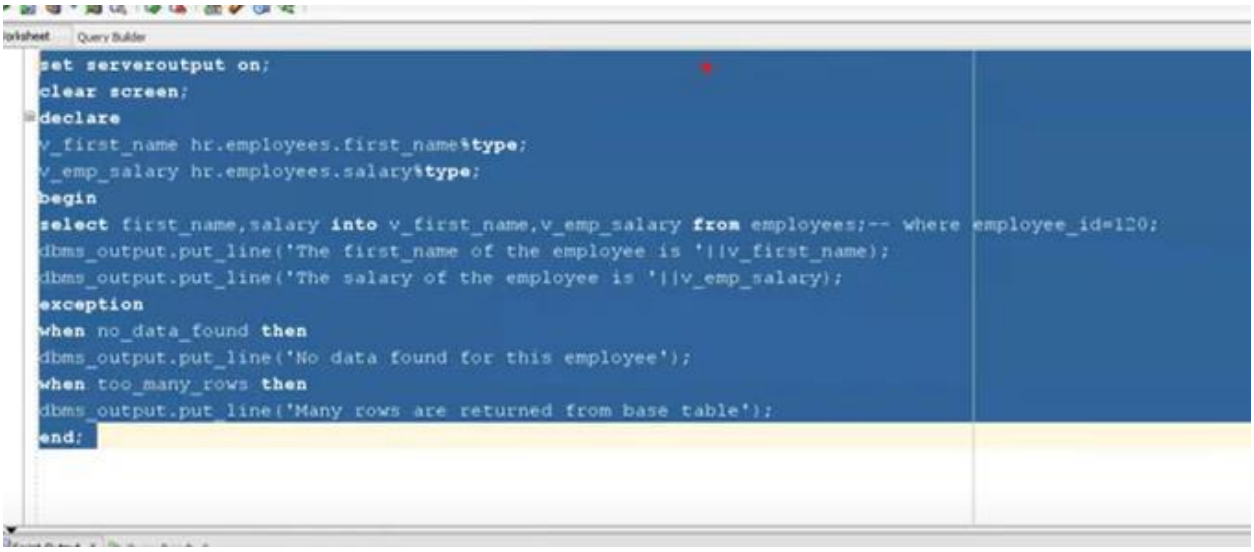
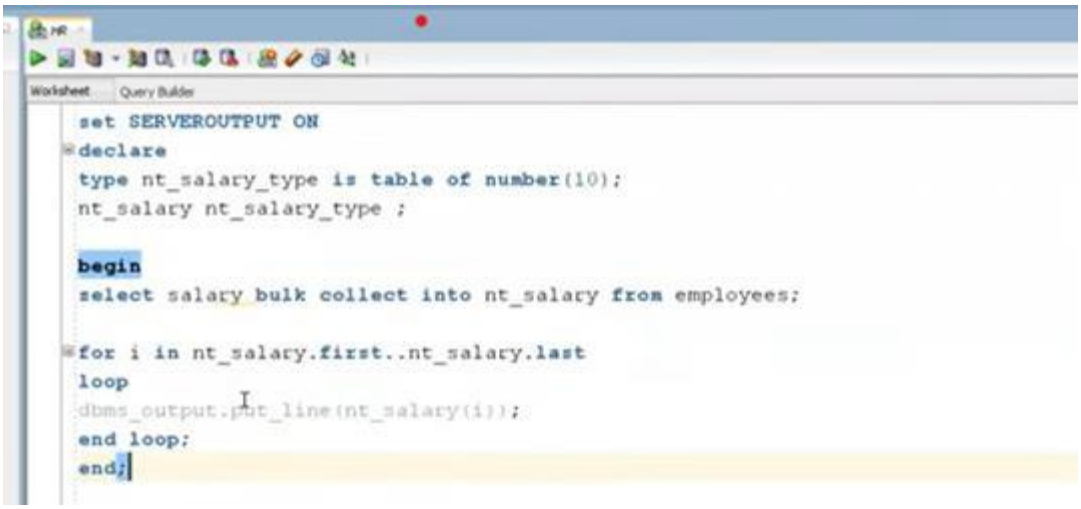


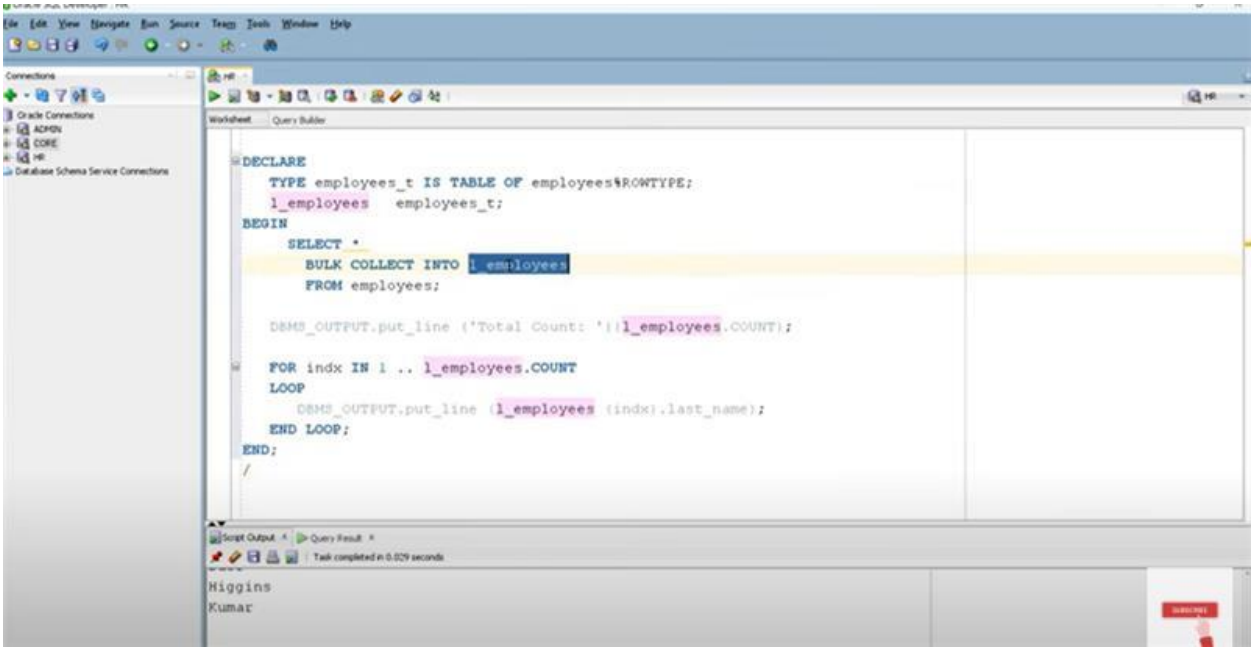
```
declare
vemp_name EMPLOYEES.FIRST_NAME%TYPE;
vemp_salary EMPLOYEES.salary%TYPE;
cursor c1 is select first_name,salary from employees;
begin
open c1;
loop
fetch c1 into vemp_name,vemp_salary;
exit when c1%notfound;
dbms_output.put_line(vemp_name||vemp_salary);
end loop;
close c1;
end;
```



```
set serveroutput on;
clear screen;
declare
v_first_name hr.employees.first_name%type;
v_emp_salary hr.employees.salary%type;
begin
select first_name,salary into v_first_name,v_emp_salary from employees;-- where employee_id=120;
dbms_output.put_line('The first_name of the employee is '||v_first_name);
dbms_output.put_line('The salary of the employee is '||v_emp_salary);
exception
when no_data_found then
dbms_output.put_line('No data found for this employee');
when too_many_rows then
dbms_output.put_line('Many rows are returned from base table');
end;
```



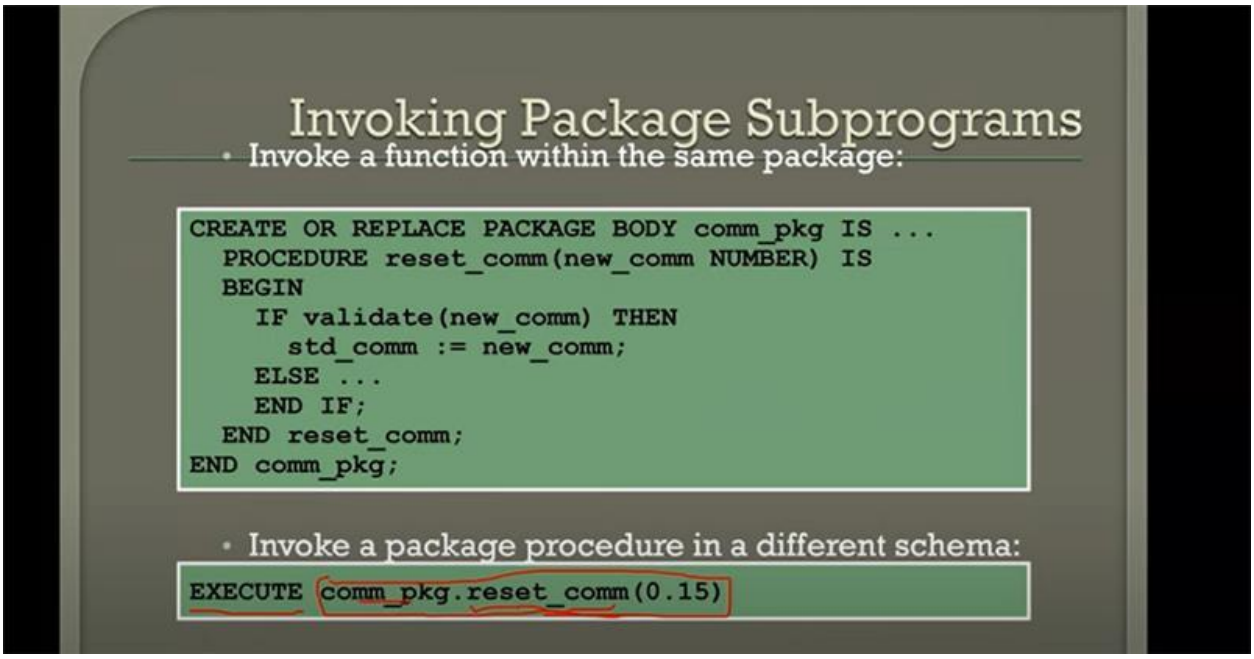
```
set SERVEROUTPUT ON
declare
type nt_salary_type is table of number(10);
nt_salary nt_salary_type ;

begin
select salary bulk collect into nt_salary from employees;

for i in nt_salary.first..nt_salary.last
loop
dbms_output.put_line(nt_salary(i));
end loop;
end;
```

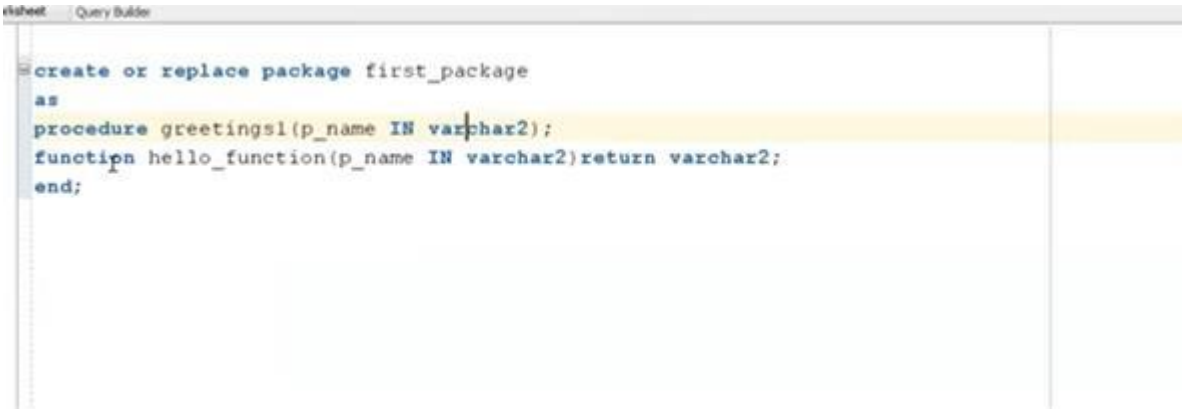


```
DECLARE
   TYPE employees_t IS TABLE OF employees%ROWTYPE;
   l_employees   employees_t;
BEGIN
    SELECT *
      BULK COLLECT INTO l_employees
      FROM employees;

   DBMS_OUTPUT.put_line ('Total Count: '||l_employees.COUNT);

   FOR indx IN 1 .. l_employees.COUNT
   LOOP
      DBMS_OUTPUT.put_line (l_employees (indx).last_name);
   END LOOP;
END;
/
```

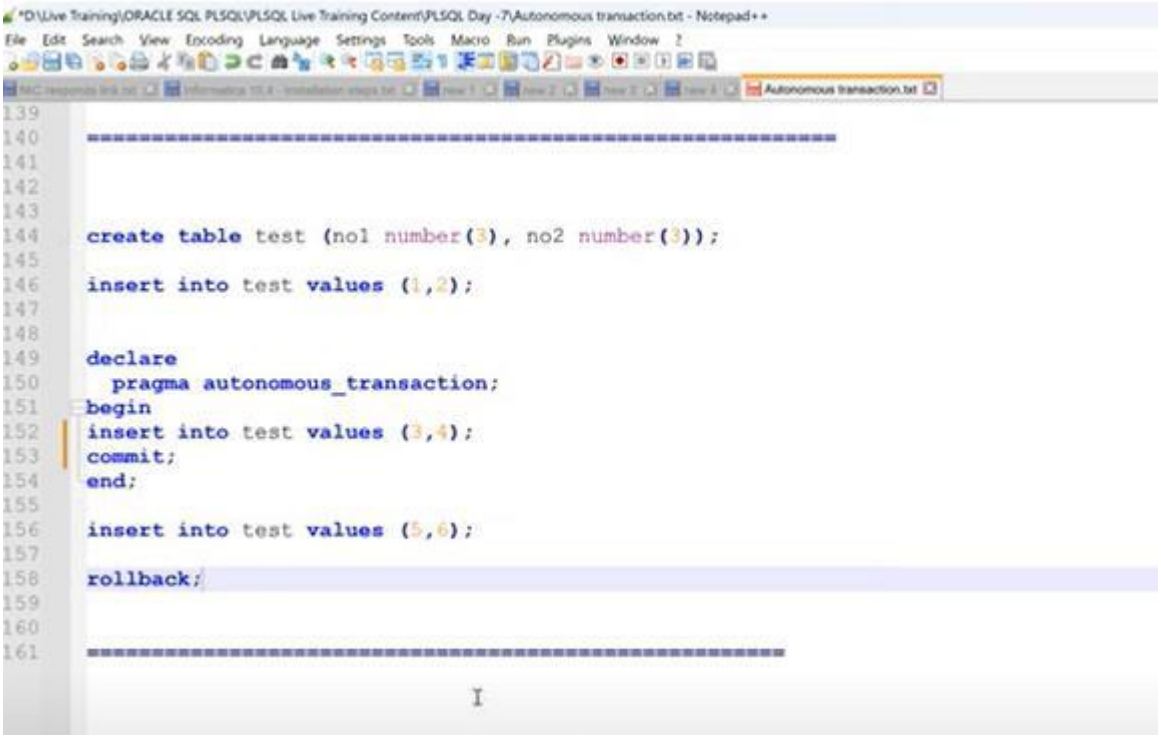
```
Higgins
Kumar
```

```
declare
v_mobile_no hr.customer.mobile_no%type;
begin
select mobile_no into v_mobile_no from customer where cust_id=1002;
dbms_output.put_line('The mobile number is '||v_mobile_no);
end;

declare
v_customer hr.customer%rowtype;
begin
select * into v_customer from customer where cust_id=1002;
dbms_output.put_line('The customer name is '||v_customer.cust_name);
dbms_output.put_line('The customer dob is '||v_customer.dob);
end;
```

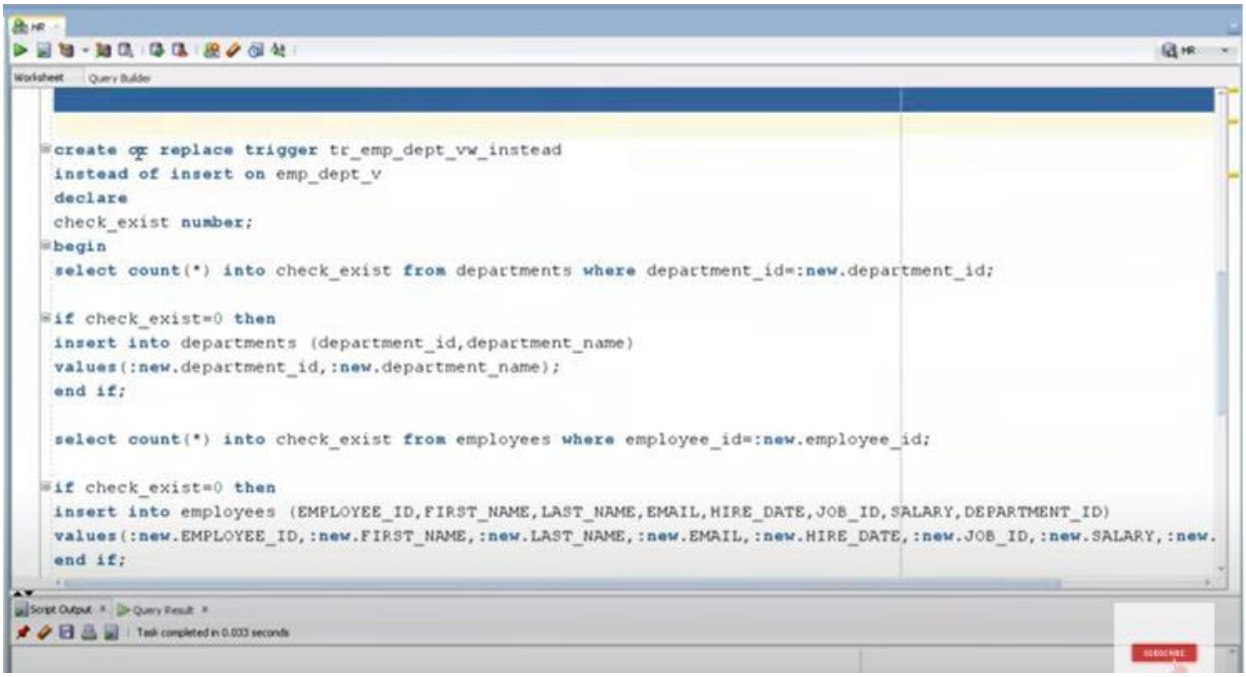


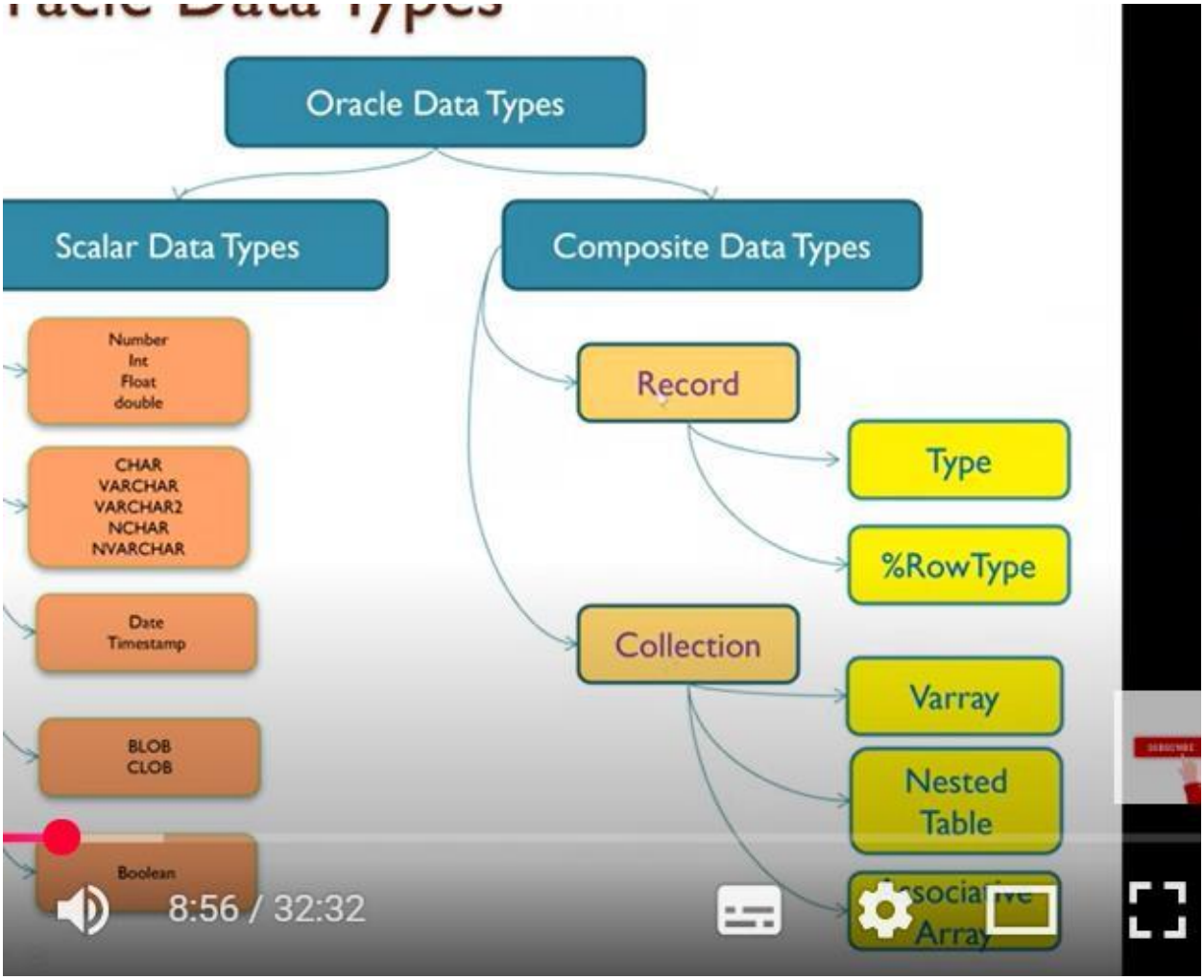


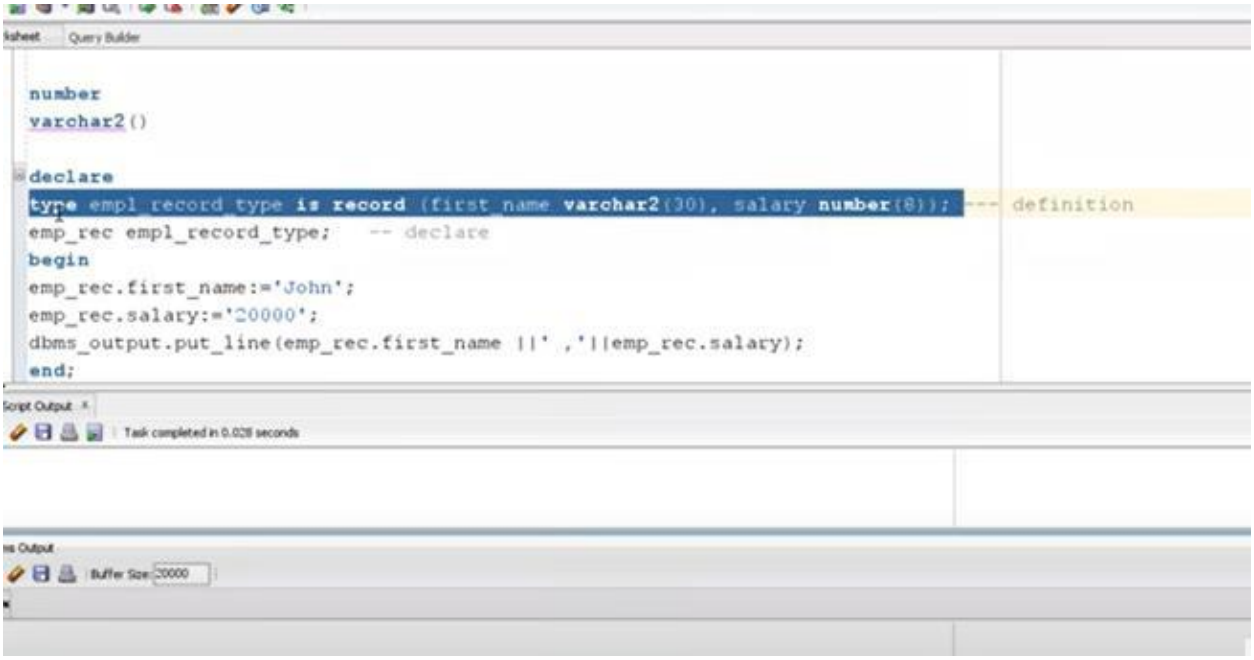
```
create or replace package first_package
as
procedure greetings1(p_name IN varchar2);
function hello_function(p_name IN varchar2)return varchar2;
end;
```



```
==========================================================



create table test (no1 number(3), no2 number(3));

insert into test values (1,2);


declare
  pragma autonomous_transaction;
begin
insert into test values (3,4);
commit;
end;

insert into test values (5,6);

rollback;


==========================================================
```


HR
Worksheet
Query Builder
create or replace trigger tr_emp_dept_vw_instead
instead of insert on emp_dept_v
declare
check_exist number;
begin
select count(*) into check_exist from departments where department_id=:new.department_id;
if check_exist=0 then
insert into departments (department_id,department_name)
values(:new.department_id,:new.department_name);
end if;
select count(*) into check_exist from employees where employee_id=:new.employee_id;
if check_exist=0 then
insert into employees (EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,HIRE_DATE,JOB_ID,SALARY,DEPARTMENT_ID)
values(:new.EMPLOYEE_ID,:new.FIRST_NAME,:new.LAST_NAME,:new.EMAIL,:new.HIRE_DATE,:new.JOB_ID,:new.SALARY,:new.
end if;
Script Output
Query Result
Task completed in 0.033 seconds


Oracle Data Types
Scalar Data Types
Composite Data Types
Number
Int
Float
double
CHAR
VARCHAR
VARCHAR2
NCHAR
NVARCHAR
Date
Timestamp
BLOB
CLOB
Boolean
Record
Type
%RowType
Collection
Varray
Nested
Table
Array
8:56 / 32:32



```
number
varchar2()

declare
type empl_record_type is record (first_name varchar2(30), salary number(8)); --- definition
emp_rec empl_record_type;    -- declare
begin
emp_rec.first_name:='John';
emp_rec.salary:='20000';
dbms_output.put_line(emp_rec.first_name ||' ,'||emp_rec.salary);
end;
```

Script Output — Task completed in 0.028 seconds

DBMS Output — Buffer Size: 20000

Collections:
==========

Collections are used in some of the most important performance optimization features of PL/SQL, such as

BULK COLLECT. SELECT statements that retrieve multiple rows with a single fetch, increasing the speed of data retrieval.

FORALL. Inserts, updates, and deletes that use collections to change multiple rows of data very quickly.

Table functions. PL/SQL functions that return collections and can be called in the FROM clause of a SELECT statement.

A Collection is an ordered group of logically related elements.

Main Purpose of using collection is to improve performance.

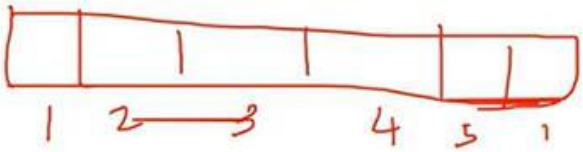
Using collection we can load all records once from database into local memory and then we can perform operation on it and save it back to database. It reduces calls to database.

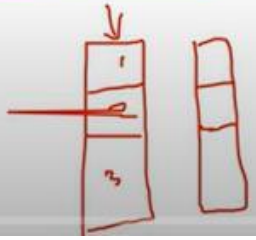
Oracle provided 3 types of collection
==========

VARRAY - variable array
Nested Table
Associative Array

Collection

- Varray
- Nested Table
- Index by table or Associative array





Associative Array

1. VARRAYS: variable size array

    arrays
    pre-defined size
    index starts with 1
    cannot delete elements .

2. Nested Tables:

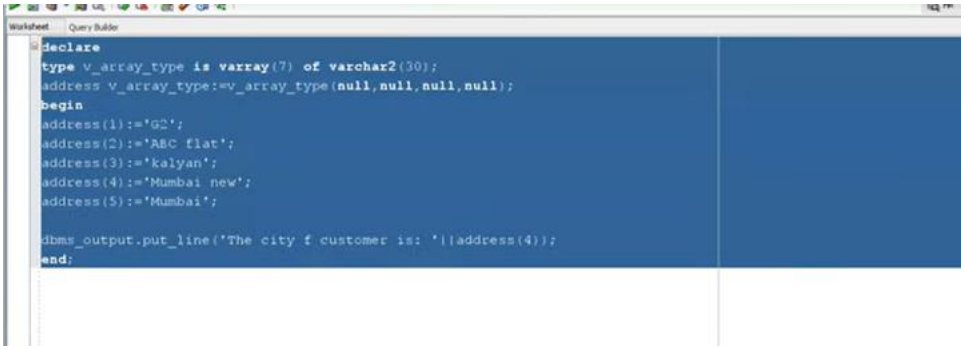
    -- List
    -- Variable Size
    -- Index starts with 1
    -- can delete the element with in the array
    -- will not define maximum limit

3. Associative Arrays - Index by tables:

    -- Indexing can be done with strings
    -- Map
    -- Key:value

Steps in Collections:

1. Define
2. Declare
3. Intialize
4. Assign
5. Access



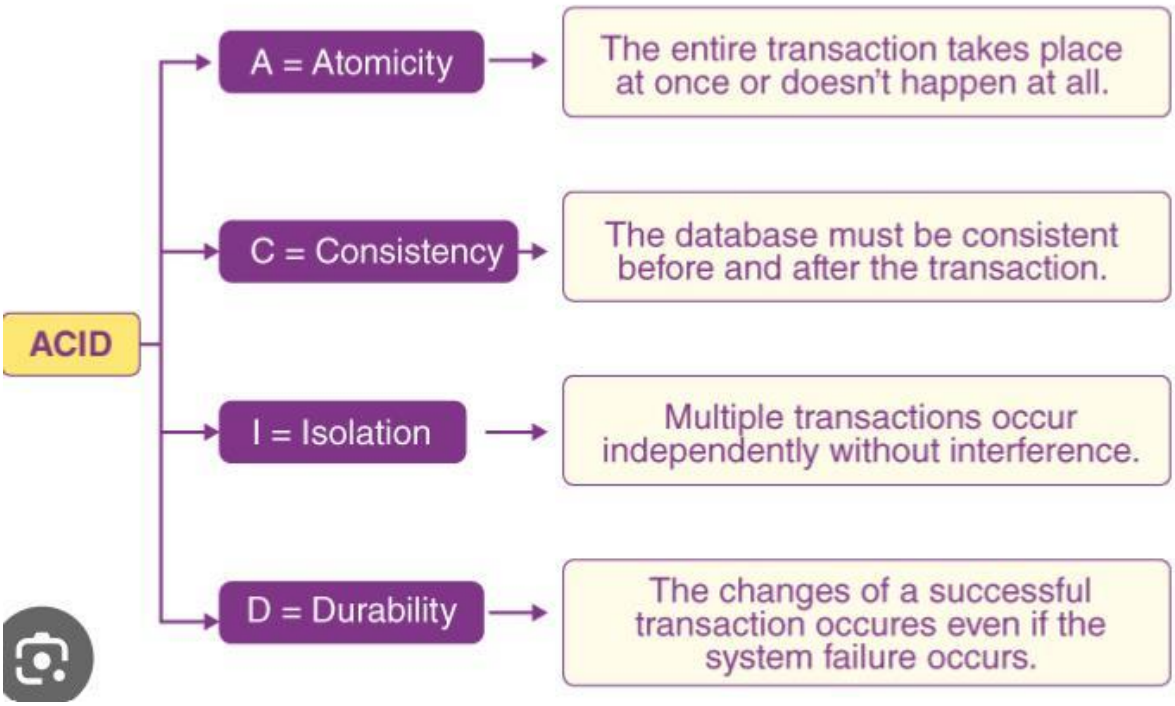
```
declare
type v_array_type is varray(7) of varchar2(30);
address v_array_type:=v_array_type(null,null,null,null);
begin
address(1):='G2';
address(2):='ABC flat';
address(3):='kalyan';
address(4):='Mumbai new';
address(5):='Mumbai';

dbms_output.put_line('The city f customer is: '||address(4));
end;
```

```
Declare
Type v_nested_table_type is table of varchar2(40);

v_color v_nested_table_type:=v_nested_table_type(null,null,null);

begin
v_color(1) := 'Red';
v_color(2) := 'Black';
v_color(3) := 'Blue';
v_color.extend(4);
v_color(4) := 'Green';
v_color.trim(2);
v_color.delete(2);
dbms_output.put_line('v_color(4) '||v_color(4));

dbms_output.put_line('v_limit '||v_color.limit);
dbms_output.put_line('v_count '||v_color.count);
dbms_output.put_line('v_first_index '||v_color.first); -- index of first element
dbms_output.put_line('v_last_index '||v_color.last);
if v_color.exists(1) then
dbms_output.put_line('v_color(2) '||v_color(2));
else
dbms_output.put_line('Elements is not available');
end if;
end;
```

```
Declare
 Type v_array_type is table of varchar2(40) index by varchar2(10);

 v_color v_array_type;

begin
 v_color('color1') := 'Red';
 v_color('color2') := 'Black';
 v_color('color3') := 'Blue';
 v_color('color4') := 'Green';

 dbms_output.put_line('v_color(3) '||v_color('color3'));

 end;


**********************************************************************
```

BYJU'S The Learning App

ACID Properties in DBMS



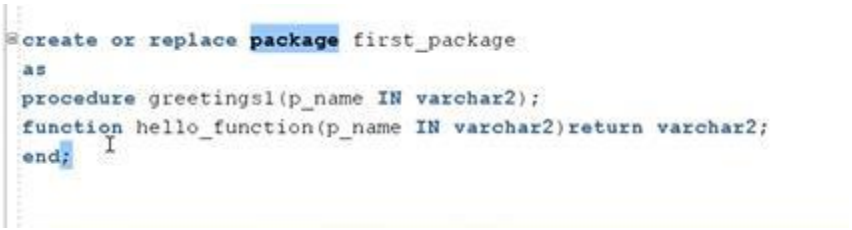
66. What are the differences between OLTP and OLAP systems?

1. **OLTP (Online Transaction Processing)**

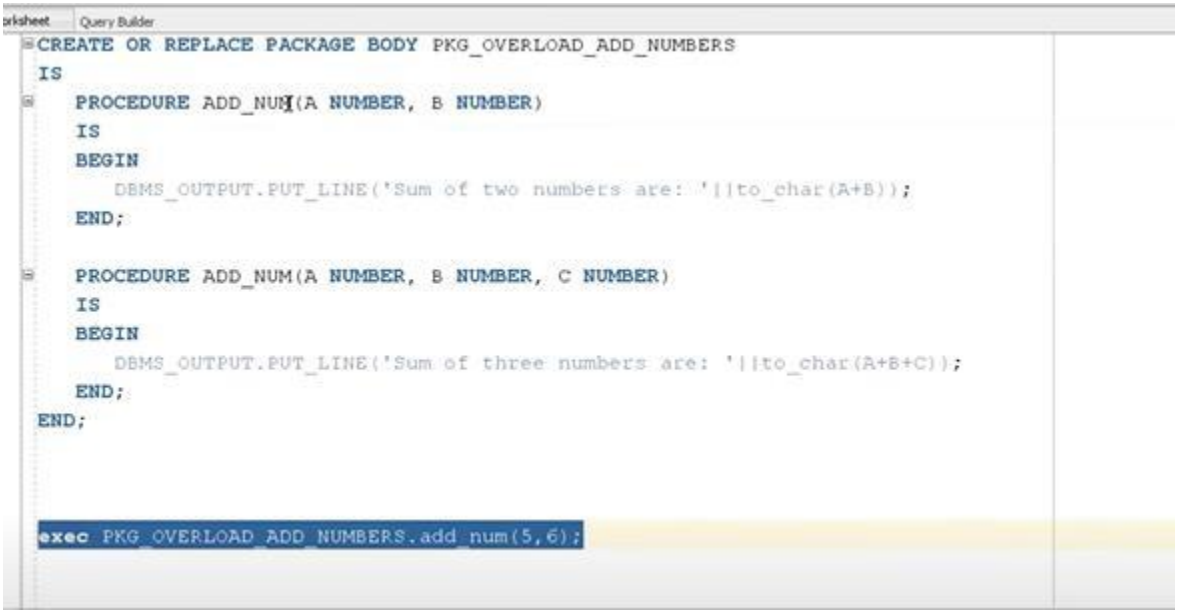
- Handles large volumes of simple transactions (e.g., order entry, inventory updates).
- Optimized for fast, frequent reads and writes.
- Normalized schema to ensure data integrity and consistency.
- Examples: e-commerce sites, banking systems.

2. **OLAP (Online Analytical Processing)**

- Handles complex queries and analysis on large datasets.
- Optimized for read-heavy workloads and data aggregation.
- Denormalized schema (e.g., star or snowflake schemas) to support faster querying.
- Examples: Business intelligence reporting, data warehousing.



```
create or replace package body first_package as
PROCEDURE greetings1(p_name IN varchar2)
AS
BEGIN
   dbms_output.put_line('Hello '||p_name);
END greetings1;
function hello_function(p_name IN varchar2)
return varchar2
as
v_result varchar2(100);
begin
v_result:='Hello '||p_name;
return v_result;
end hello_function;
end;
```



```
create table test (no1 number(3), no2 number(3));

insert into test values (1,2);


declare
  pragma autonomous_transaction;
begin
insert into test values (3,4);
commit;
end;

insert into test values (5,6);

rollback;
```

```
create or replace trigger tr_emp_dept_vw_instead
instead of insert on emp_dept_v
declare
check_exist number;
begin
select count(*) into check_exist from departments where department_id=:new.department_id;

if check_exist=0 then
insert into departments (department_id,department_name)
values(:new.department_id,:new.department_name);
end if;

select count(*) into check_exist from employees where employee_id=:new.employee_id;

if check_exist=0 then
insert into employees (EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,HIRE_DATE,JOB_ID,SALARY,DEPARTMENT_ID)
values(:new.EMPLOYEE_ID,:new.FIRST_NAME,:new.LAST_NAME,:new.EMAIL,:new.HIRE_DATE,:new.JOB_ID,:new.SALARY,:new.
end if;
end;
```

```
number
varchar2()

declare
type empl_record_type is record (first_name varchar2(30), salary number(8)); --- definition
emp_rec empl_record_type;   -- declare
begin
emp_rec.first_name:='John';
emp_rec.salary:='20000';
dbms_output.put_line(emp_rec.first_name ||' ,'||emp_rec.salary);
end;
```

```
declare
type emp_rec_type is record (first_name varchar2(20),salary number(8));
emp_rec emp_rec_type;
begin
select first_name, salary into emp_rec from employees where employee_id=120;
dbms_output.put_line('emp_rec.first_name is :='||emp_rec.first_name);
dbms_output.put_line('emp_rec.salary is :='||emp_rec.salary);
end;
```