/*
Now PROJECTION
*/

begin
  for carol in
    (select num, cube from x)
  loop
    dbms_output.put_line(carol.num ||
                         ' CUBE IS: ' || carol.cube);
  end loop;
end;
/
/

/*
begin
  for counter in
    (select * from x)
  loop
    dbms_output.put_line(counter.num ||
                           ' ' || counter.cube);
  end loop;
end;
/
/
If we have a * we can use all the columns in the loop body.
P21

/*
 * Now selecting a few rows:
 * A SELECT in the sense of Relational Algebra
 */

begin
  for counter in (select * from x where sqr > 4) loop
    dbms_output.put_line(counter.num || ' ' || counter.sqr || ' ' || counter.cube);
  end loop;
end;
/

P21b

/*
 * Now selecting with a PL/SQL variable in the select statement.
 */

declare
  anumber number;

begin
  anumber := 1+1;
  for counter in (select * from x where num > anumber) loop
    dbms_output.put_line(counter.num || ' ' || counter.sqr || ' ' || counter.cube);
  end loop;
end;
/
P21c
/
Nesting cursor loop in normal loop is okay (of course!).
/>

declare
anumber number;
begin
for i in 1..12
loop
  anumber := i + 1;
dbms_output.put_line();
  for counter in (select * from x where num > anumber)
  loop
    dbms_output.put_line(counter.num || ' ' || counter.sqr || ' ' || counter.cube);
  end loop;
end loop;
end:
/

P21d
/
What if I use a local variable with the same name as a column?
In this case, there is no syntax error.
However, the second num in the WHERE appears to be the column name ALSO.
We can show this by changing the > sign to >=.
num > num is never true.
um >= num is always true.
The column name "shadows" the variable name.
*/

declare
num number;
begin
for i in 1..12
loop
  num := i + 1;
dbms_output.put_line();
  for counter in (select * from x where num > num) -- This is never true
  loop
    dbms_output.put_line(counter.num || ' ' || counter.sqr || ' ' || counter.cube);
  end loop;
end loop;
end;
/*
Now it's time to learn how to insert data.
The insert syntax is the same as in SQL.

Insert is simple and does not require something like a cursor because the program itself does not have to handle a table of data.

The difficulty of processing a whole table of data inside of a program is one version of the "impedance mismatch problem."

*/

begin
  insert into x values (5, 25, 125);
end;
/

P22b

/*
More inserts.
This also shows a local function.
*/
declare
  anu number; asq number; acu number;
function mysql(x number)
  return number is
    begin
      return x * x;
    end;
function mycu(x number)
  return number is
    begin
      return x * x * x;
    end;
begin
  anu := 7;
  asq := mysql(7);
  acu := mycu(7);
  insert into x values (anu, asq, acu);
end;
/
/*
Insert in a loop.
*/

declare
    anu number;
    asq number;
    acu number;
    function mysq(y number)
        return number is
        begin
            return y * y;
        end;
    function mycu(y number)
        return number is
        begin
            return y * y * y;
        end;
begin
    for i in 5..2000
        loop
            anu := i;
            asq := mysq(i);
            acu := mycu(i);
            insert into x values (anu, asq, acu);
        end loop;
end;
/

P23
/*
Can we delete? Yes we can.
*/

declare
    abc number;
begin
    abc := 6;
    delete from x where num = 5;
    delete from x where num = abc;
end;
/
/*
Can we update? Yes we can!
*/

begin
  update x
  set num = 5
  where num = 4;
end;
/

P24b

/*
Can we update? Yes we can!
*/

declare
  mywhat number;
  mywhere number;

begin
  mywhere := 5;
  mywhat := 25;

  update x
  set sqx = mywhat
  where num = mywhere;
end;
/

P24d

// This program seems to work.
// But it DOES NOT CHANGE THE DATABASE

begin
  for j in (select * from x)
    loop
      num := 0;
      dbms_output.put_line(j, num);
    end loop;
end;

P24f

-- Do you always need a cursor?
-- Well, this DOES NOT work.

declare
  n number;
begin
  n := select count(*) from x;
  dbms_output.put_line(n);
end;
P24g

/* However, THESE WORK: */

declare
  n number;
begin
  select count(*) into n from x;
  dbms_output.put_line(n);
end;

declare
  a number;
  b number := 0;
begin
  select num, sar into a, b from x where num=2;
  dbms_output.put_line(a || '' || b);
end;

P25

/*
Can we alter? NO, we cannot!!!
(Without advanced packages.)
*/

begin
  create table temp (num number); -- Does NOT work.
  alter table x add (sum number); -- Does NOT work.
end;
/

This works:

begin
  execute immediate 'create table temp (num number)';
  execute immediate 'alter table x add (sum number)'; -- Does work.
end;

P25b
/*
Can we create from inside of
a PL/SQL program? NO, we cannot!!
(Unless we execute immediate.)
*/

begin
    create table ztop (
        a number,
        b number,
        c number
    );

end;
/

/*