```
REM   Script: 18c Echo Polymorphic Table Function
REM   This PTF returns all the columns in the input table tab, and adds to it the columns listed in cols but with the
column names changed with the additional prefix of "ECHO_".

-- The echo_package package specification defines the DESCRIBE and FETCH_ROWS methods.
-- The parameters are : tab (The input table) and cols (The list of columns).
CREATE OR REPLACE PACKAGE echo_package AS
  FUNCTION describe(tab   IN OUT dbms_tf.table_t,
                    cols  IN     dbms_tf.columns_t)
           RETURN dbms_tf.describe_t;

  PROCEDURE open;

  PROCEDURE fetch_rows;

  PROCEDURE close;
END echo_package;
/

-- The echo_package package body contains the PTF implementation.
CREATE OR REPLACE PACKAGE BODY echo_package AS
  col_name_prefix CONSTANT dbms_id := '"ECHO_';
  col_name_suffix CONSTANT dbms_id := '"';

  FUNCTION describe(tab   IN OUT dbms_tf.table_t,
                    cols  IN     dbms_tf.columns_t)
           RETURN dbms_tf.describe_t
  AS
    new_cols dbms_tf.columns_new_t;
    col_id   PLS_INTEGER := 1;
  BEGIN

    DBMS_TF.Trace('Describe()');
    DBMS_TF.Trace('Looping over '||  tab.column.count || ' columns....');

    FOR i IN 1 .. tab.column.count LOOP
      continue WHEN NOT dbms_tf.supported_type(tab.column(i).description.TYPE);

      DBMS_TF.TRACE('Processing...'||tab.column(i).description.name);

      FOR j IN 1 .. cols.count LOOP

        DBMS_TF.TRACE('     Looking for...'||cols(j));

        IF (tab.column(i).description.name = cols(j)) THEN
          tab.column(i).for_read:= true;

          DBMS_TF.TRACE('          Working on column...'||tab.column(i).description.name);

          new_cols(col_id)      := tab.column(i).description;
          new_cols(col_id).name := col_name_prefix
                                || regexp_replace(new_cols(col_id).name, '^"|"$')
                                || col_name_suffix;


          EXIT;
        END IF;
          col_id                := col_id + 1;
      END LOOP;
    END LOOP;

    RETURN dbms_tf.describe_t(new_columns => new_cols);
  END;

 PROCEDURE Open
as
   env DBMS_TF.env_t := DBMS_TF.Get_Env();

begin
  DBMS_TF.Trace('Open()');
  DBMS_TF.Trace('Get_Col.Count = '||env.get_columns.count, prefix => '....');
  DBMS_TF.Trace('Put_Col.Count = '||env.put_columns.count, prefix => '....');
end;


  PROCEDURE fetch_rows
  AS
    rowset dbms_tf.row_set_t;

  BEGIN

    DBMS_TF.Trace('Fetch_Rows()');


    dbms_tf.get_row_set(rowset);
    dbms_tf.put_row_set(rowset);
  END;


PROCEDURE Close
as
   env DBMS_TF.env_t := DBMS_TF.Get_Env();

begin
  DBMS_TF.Trace('Close()');
--  DBMS_TF.Trace('Passed in '||env.get_columns.count|| ' columns', prefix => '....');
--  DBMS_TF.Trace('Processed = '||env.put_columns.count|| ' columns', prefix => '....');
end;

END echo_package;
/

-- The PTF echo references the implementation package echo_package.
CREATE OR REPLACE FUNCTION echo(tab TABLE, cols columns) RETURN TABLE
       PIPELINED ROW POLYMORPHIC USING echo_package;
/
set SERVEROUTPUT on;

SELECT *
FROM   echo(emp, COLUMNS(first_name, salary, hire_date))
WHERE  department_id = 20;


set SERVEROUTPUT on;

SELECT *
FROM   echo(dept, COLUMNS(department_name, location_id));
```