Әділжанов Есіл Мухамедғалиұлы

Для данного задания вам потребуется работающая БД oracle. Вы можете, например, воспользоваться Oracle Database Express Edition

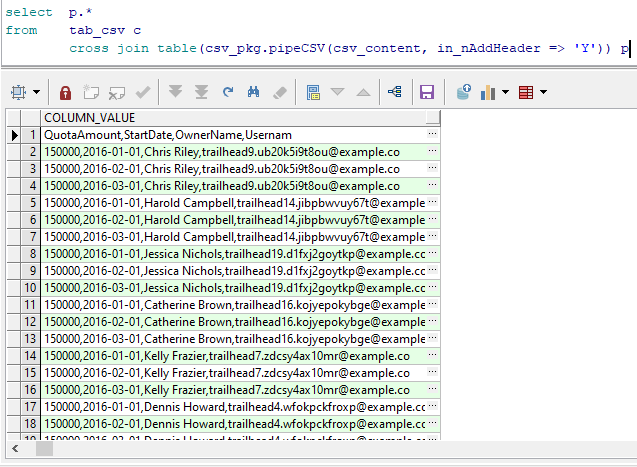
Вы можете пользоваться любыми ресурсами и использовать чужой код, но вам потребуется к каждой строчке кода написать достаточно детальный комментарий о том, что в ней происходит, дабы мы понимали, что вы представляете себе, как работают используемые функции и в целом приведенный вами код.

**Задание**

Написать парсер данных в формате CSV. Парсер должен быть оформлен как pipelined функция, которая на входе получает clob, и выдает строки с данными

Обязательно добавить параметр, который позволит включать или исключать заголовки CSV (первую строку)

Вы можете пойти двумя путями, либо их комбинацией:

1. Написать функцию, которая будет возвращать одну длинную строку. По сути вы разобьёте CSV на строки, а далее уже можно будет из этих строк брать нужные колонки  
   Вот пример использования такой функции:  
   
2. Написать функцию, которая будет так же делить CSV на строки, но помимо этого ещё разбивать эти строки на колонки и именно их возвращать (так например работает парсер в Oracle Apex, распределяя данные в колонки в apex\_collections). Так же не лишним было бы добавить в функцию параметр «Разделитель», потому что в CSV они (разделители) могут быть разные. Тем не менее, если это сложно, то данный параметр можно опустить.
3. Написать обе функции, при этом разбивку на строки отдать функции из п.1 и просто получать результат в функции из п.2 и делить его на колонки

После того как вы убедитесь, что ваш парсер работает, вам необходимо создать таблицу под данные CSV из примера с нужными типами (тип колонки вы можете понять исходя из данных). Таблицу и колонки можете назвать как угодно. Далее нужно будет возвращаемые функцией данные вставить в эту таблицу.

Мы ожидаем от вас результат в виде пакета (заголовок и тело) с pipelined функцией, ddl с таблицей под данные и dml, которым вы вставляете данные в эту таблицу

Схематично это бы выглядело так:

package ...

/

package body ...

/

create table ...

/

insert into ...

/

select \* from …

CSV для разбора находится в этом файле



* Пакет

*-- создаем пакет внутри которых объявлены 2 records(записи) ,2 составных типов TABLE, и 3 piplined functions(Табличные функции)*

CREATE OR REPLACE PACKAGE YESSIL\_csv\_pkg IS

type t\_csv is record

(

col\_value clob

);

type t\_csv\_cols is record

(

QuotaAmount clob,

StartDate clob,

DAY clob,

Username clob

);

type t\_csv\_table is table of t\_csv;

type t\_csv\_cols\_table is table of t\_csv\_cols;

function pipeCSV(clob\_in clob,head in char) return t\_csv\_table pipelined;

function pipeCSV\_toColumns(clob\_in clob) return t\_csv\_cols\_table pipelined;

function pipeCSV\_cols\_ver2(clob\_in clob) return t\_csv\_cols\_table pipelined;

END YESSIL\_csv\_pkg;

* Тело пакета

create or replace package body YESSIL\_csv\_pkg as

*-- Задание 1*

function pipeCSV(clob\_in clob,head in char) return t\_csv\_table pipelined as

begin

if(head in ('Y','y')) then *-- проверка параметра, который позволит включать или исключать заголовки*

for rec in

(

*/\* сперва заменяем chr(10) на ';' с помощью функции REGEXP\_REPLACE, так как я не смог сразу парсить через chr(10),*

*потом по шаблону '[^;]+' получаем нужные нам данные*

*[^;] -- получаем любые символы, за исключением ";"*

*'+' -- Соответствует одному или более вхождений.*

*и исползуем connect by level рекурсивную функцию для фиксации точки*

*\*/*

select regexp\_substr(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+', 1, rownum) col\_value

from dual

connect by level <= length(regexp\_replace(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+')) + 1

)

loop

pipe row (rec);

end loop;

else

for rec in

(

select col\_value from

( select rownum as n,regexp\_substr(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+', 1, rownum) col\_value

from dual

connect by level <= length(regexp\_replace(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+')) + 1

)

where n <> 1

)

loop

pipe row (rec);

end loop;

end if;

end;

*-- Задание 2*

function pipeCSV\_toColumns(clob\_in clob) return t\_csv\_cols\_table pipelined as

begin

for rec in

(

with tt as (select col\_value as txt from

( select rownum as n,regexp\_substr(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+', 1, rownum) col\_value

from dual

connect by level <= length(regexp\_replace(REGEXP\_REPLACE(clob\_in, chr(10), ';'), '[^;]+')) + 1

)

where n <> 1

)

select regexp\_substr(txt, '[^,]+', 1, 1) QuotaAmount,

regexp\_substr(txt, '[^,]+', 1, 2) StartDate,

regexp\_substr(txt, '[^,]+', 1, 3) DAY,

regexp\_substr(txt, '[^,]+', 1, 4) Username

from tt

)

loop

pipe row (rec);

end loop;

end;

*-- Задание 3*

function pipeCSV\_cols\_ver2(clob\_in clob) return t\_csv\_cols\_table pipelined as

begin

for rec in

(

with tt as (

select col\_value txt from table(YESSIL\_CSV\_PKG.pipeCSV(clob\_in,'n'))

)

select regexp\_substr(txt, '[^,]+', 1, 1) QuotaAmount,

regexp\_substr(txt, '[^,]+', 1, 2) StartDate,

regexp\_substr(txt, '[^,]+', 1, 3) DAY,

regexp\_substr(txt, '[^,]+', 1, 4) Username

from tt

)

loop

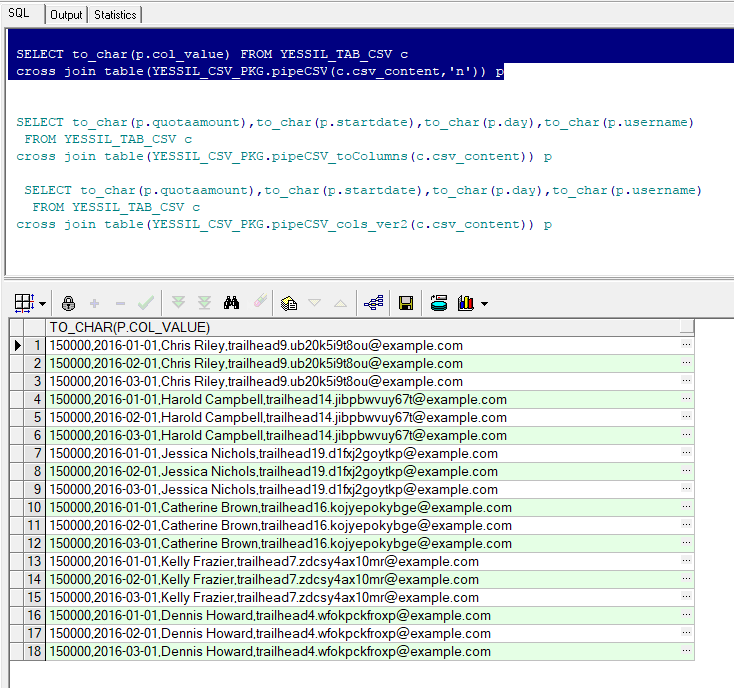
pipe row (rec);

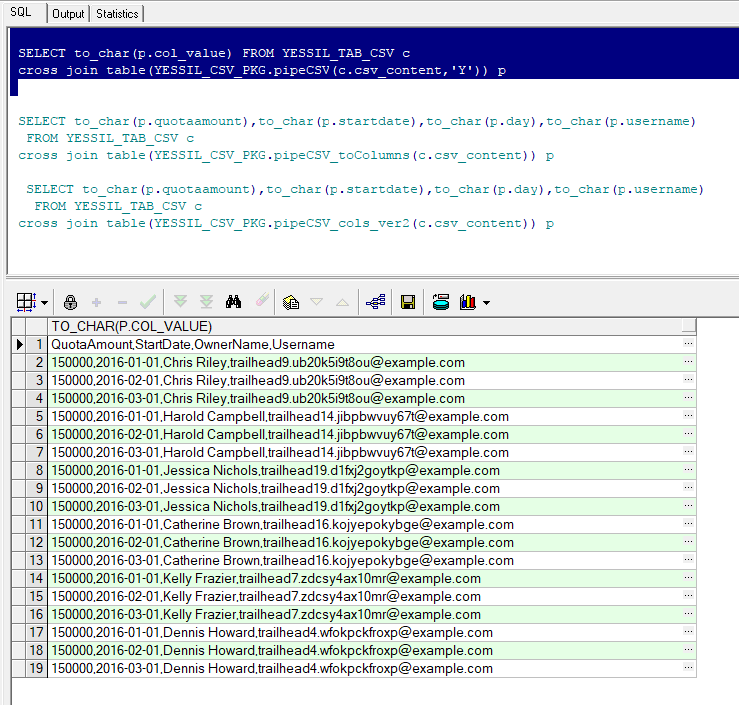
end loop;

end;

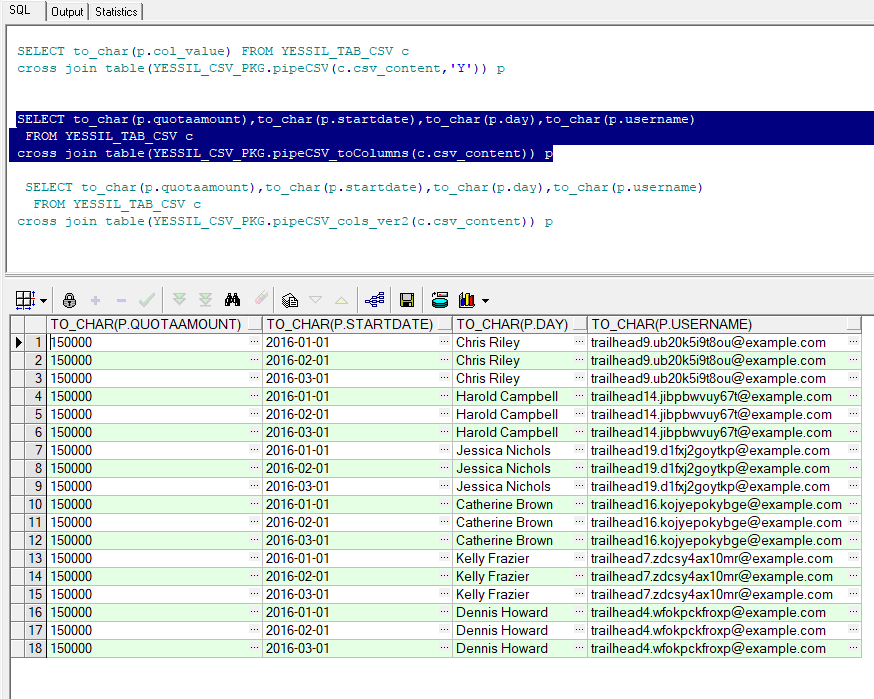
end;

* Проверка

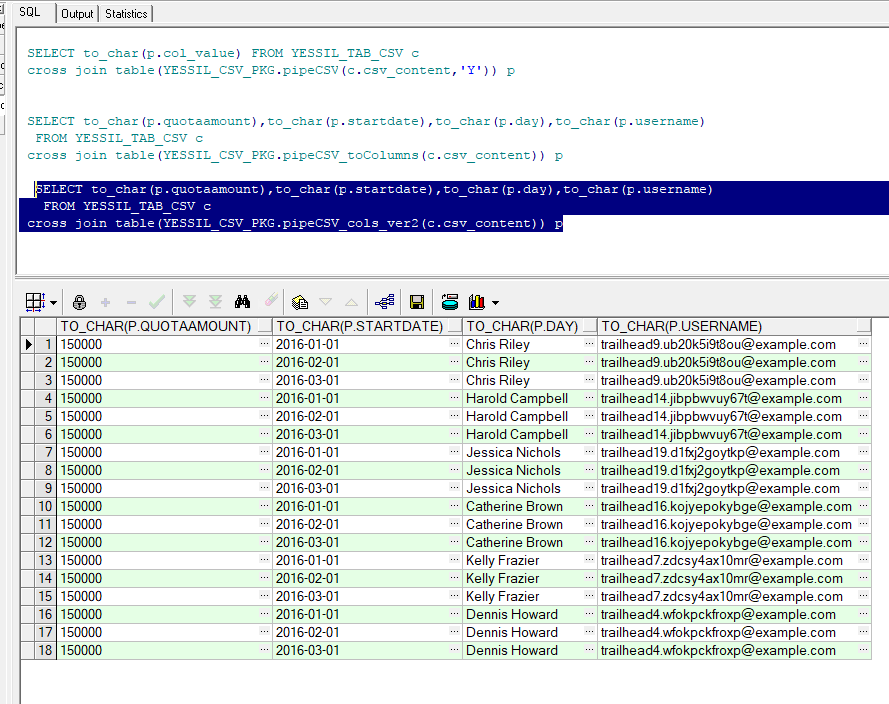
1)



2)



3)



Таблица

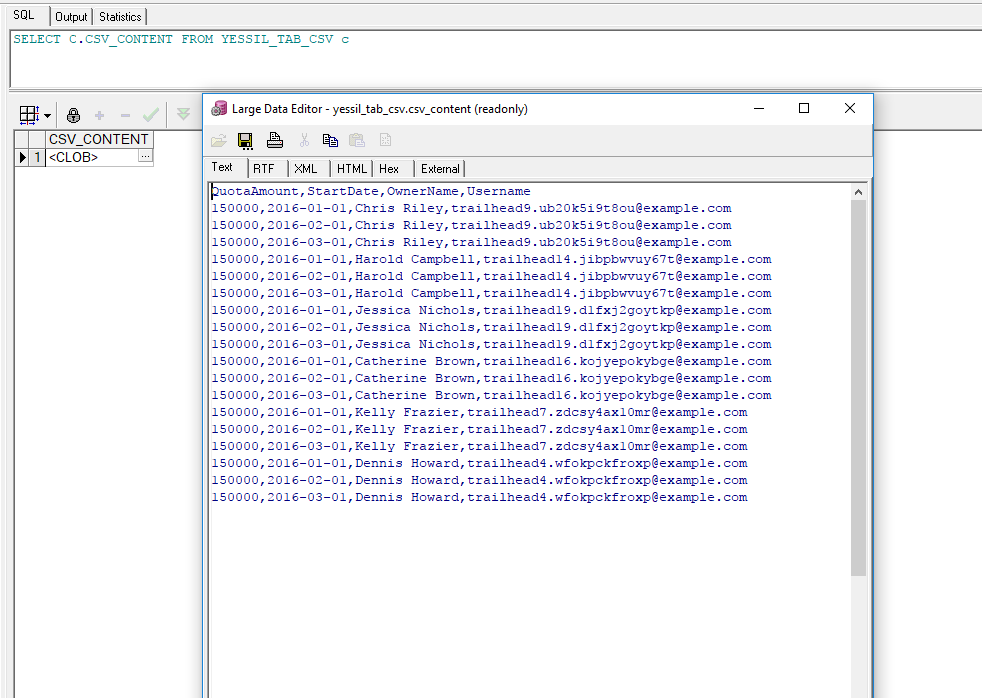


Таблица под данные csv

CREATE TABLE csv\_parsed\_table

( quotaamount clob,

startdate clob,

day clob,

username clob

);

insert into csv\_parsed\_table

SELECT p.quotaamount,p.startdate,p.day,p.username

FROM YESSIL\_TAB\_CSV c

cross join table(YESSIL\_CSV\_PKG.pipeCSV\_cols\_ver2(c.csv\_content)) p

