Изисквания :

1/ обща Oracle база за проекта, ще се предостави на всеки екип

2/ ПроС програмите са разработват на линукс.

Alter table customers add external_reference varchar2(60);

update customers set external_reference = 'EXT' || customer_id where external_reference is null ;

//Да се добави unique not null constraint към полето customers.external_reference.

//За стари съществуващи данни да е във формат EXT3427.

Alter table payments add original_payment_id varchar2(60); -> key kym payment_id

Alter table invoices add original_invoice_id varchar2(60); -> key kym inv_id

// original_invoice_id и original_payment_id трябва да съдържат съответно само стойности от полетата invoice_id и payment_id или стойност NULL.
Да се създаде такава релация в базата от данни.

Още една таблица :

```
create table over_payments (
       invoice_no      varchar2(50),
       payment_id      varchar2(60),
       payment_dt      date,
       amount          number(10,2),
       payment_method  varchar2(10),
       currency        varchar2(3),
       fg_processed    varchar2(1),
       iban            varchar2(60),
       bank_name       varchar2(100),
       created_by      varchar2(50),
       free_text       varchar2(1000),
       original_payment_id varchar2(60),
       constraint op_paym_no_pk primary key (payment_id) ,
       CONSTRAINT op_fk_inv FOREIGN KEY (invoice_no) REFERENCES
invoices(inv_no)
);
```

```
alter table over_payments
   add constraint fk_over_pay_id_orig_paym_id
   foreign key (original_payment_id) references payments(payment_id)
   deferrable initially deferred;
```

Задание

1. Създайте входен интерфейс(програма, която чете данни от файлове, обработва данните и ги слага в базата). Интерфейсът чете файлове, които се намират в предварително дефинирана (от вас) директория и имат име във формат IMP_CUSTOMER_DATA_YYYYMMDDHH24MISS.txt, където YYYYMMDDHH24MISS е датата с часове, минути и секунди, например 20230214183755. Ако в директорията има повече от

един файл се обработват по ред като се започва от най-стария спрямо датата в името на файла. Всеки файл се чете линия по линия. Отварят се два файла в друга директория със същото име и разширение .OK и .KO . Всяка успешно обработена линия се записва във файла с разширение .OK, а всяка грешна се записва във файла .KO. Веднъж обработен, файлът се мести от основната директория в друга директория с разширение .back
Във всеки файл може да има 3 вида линии – клиент, фактура, плащане. Първото поле от линията определя нейния тип (01 – клиент, 02 - фактура, 03 - плащане). Всяко поле е разделено със символ **;** (символът **;** не може да се съдържа в данните).
Примерен файл:
01;;EXT3427;VALENTIN;MILANOV;m;SOFIA;BULGARIA;998;v-milanov@jjj.gg
01;79;EXT1473;IVELINA;SIMOVA;f;;BULGARIA;5000;ivvvv45@jjj.gg
02;EXT3427;15022023;60.00;BGN
02;EXT3427;17022023;20.00;BGN
03;EXT3427;16022023;70;BankTransf;BGN;

Формат на линиите :

1	Тип на линията	2 символа, цифри, задължително поле	01 – клиент
2	Вътрешна референция на клиента	Макс. 20 символа, цифри,не задължително поле	Ex. 456
3	Външна референция на клиента	Макс. 60 символа, цифри и букви, задължително поле	Ex. EXT3427
4	Първо име	Макс. 60 символа, задължително поле	Ex. IVELINA
5	Фамилия	Макс. 60 символа, задължително поле	Ex. SIMOVA
6	Пол	1 символ, задължително поле	Стойност m или f
7	град	Макс. 50 символа, не задължително поле	Ex. SOFIA
8	Държава	Макс. 30 символа, задължително поле	Ex. BULGARIA
9	Лимит	Макс. 10 символа, цифри, задължително поле	Ex. 5000
10	Email	Макс. 50 символа, задължително поле	Ex. ivvvv45@jjj.gg

1	Тип на линията	2 символа, цифри, задължително поле	02 – фактура
2	Външна референция на клиента	Макс. 60 символа, цифри и букви, задължително поле	Ex. EXT3427
3	Дата на фактурата	8 символа, задължително поле, формат DDMMYYYY	Ex. 15022023
4	Сума на фактурата	До 10 цифри преди точката и до 2 след точката , задължително поле	Ex. 60.00
5	Валута на фактурата	3 символа, задължително поле	Ex. BGN

1	Тип на линията	2 символа, цифри, задължително поле	03 – плащане
2	Външна референция на клиента	Макс. 60 символа, цифри и букви, задължително поле	Ex. EXT3427
3	Дата на плащането	8 символа, задължително поле, формат DDMMYYYY	Ex. 16022023
4	Сума на плащането	До 10 цифри преди точката и до 2 след точката , задължително поле	Ex. 60.00
5	Начин на плащене	До 10 символа, задължително поле	Ex. BankTransf
6	Валута на плащането	3 символа, задължително поле	Ex. BGN

Обработка на линиите :
01 – Клиент
Проверява се дали има стойност в полето “Вътрешна референция на клиента”

- Ако да (попълнено е), тогава се търси линия в базата в таблицата CUSTOMERS със същото customer_id.
    - Ако не съществува клиент с това ID, отхвърляме линията
    - Ако съществува клинте със същото ID, тогава проверяваме дали някое от следващите полета в линия е различно от данните в базата. Ако да, ги променяме
- Ако не (полето “Вътрешна референция на клиента” е празно), тогава търсим в базата клиент (таблица CUSTOMERS), който external_reference = “Външна референция на клиента” (трето поле от линията).

- Ако не съществува, тогава инсъртаваме нов клиент с всички данни от линията (ред в таблицата CUSTOMERS)
- Ако съществува клинте със същия external_reference, тогава проверяваме дали някое от следващите полета в линия е различно от данните в базата. Ако да, ги променяме

02 – фактура
Търсим линия в CUSTOMERS, където customers.external_reference е равна на стойността в полето "Външна референция на клиента". Ако намерим, създаваме фактура за този клиент. Ако не намерим, отхвърляме линията с грешка "Фактура по несъществуващ клиент".

03 – плащане
Търсим линия в CUSTOMERS, където customers.external_reference е равна на стойността в полето "Външна референция на клиента".

- Ако не намерим, отхвърляме линията с грешка "Плащане от несъществуващ клиент".
- Ако намерим клиента, слагаме плащането по последната фактура на този клиент. Ако клиента няма нито една фактура, отхвърляме линията.

// като дойде плащане винаги ще отива по последната фактура – виж долу тригера трябва да го хване и оправи, дори фактурата да е платена, ще трябва да му смени сумата на 0 и да сложи друго вързано към това с цялата сума и NULL за Invoice_no

2. Напишете процедура – която да попълни полето invoice.due_date, за всички фактури където това поле е NULL по правилата : ако за същия клиент полето credit_limit е по-малко от 5000, тогава 10 дни след датата на издаване на фактурата; ако credit_limit е между 5000 и 8000, тогава 15 дни след датата на издаване на фактурата и ако е по-голямо от 8000, тогава 20 дни.

3. Напишете тригер, който при инсърт в таблицата PAYMENTS ~~или ъпдейт на полето PAYMENTS.invoice_no от NULL на стойност различна от NULL, тогава~~ ако сумата е по-голяма от дължимата сума по фактурата да направи сумата на плащането да е колкото се дължи по фактурата. С разликата от сумата се инсъртва ~~друго плащане~~ линия в таблицата over_payments, където original_payment_id = payment_id на главния запис и invoice_no is NULL. Ако се промени дължимата стойноста на фактура се променя и в базата.

4. Да се напише бач (ПроС програма, която обработва данни), който за всички ~~плащания, където invoice_no is NULL~~ надплатени суми (линии в таблицата over_payments) търси фактура, която ~~това плащане може да покрие~~ е надплатена за същия клиент и ако намери ~~променя invoice_no да е номера на фактурата~~ мести (копира данните и след това изтрива) линията от таблицата over_payments в таблицата payments със попълнен номер на фактура.

5. Да се напише бач, който да слагага лихви – линия в таблицата Invoices по правилото – за всички фактури, които не са напълно платени и са минали 5 дни от датата на която се дължат, се слага фактура за същия клиент, която е на стойност 5% от дължимата сума по фактурата. original_invoice_id на новата фактура е = invoice_id на главния запис. Да не слагаме лихва върху лихвата (където original_invoice_id е попълнено не се взимат тези линии)

//4. и 5. Да се дообави входен параметър customer на бачовете (т.е. customers.external_reference). Ако е попълнен, бачовете работят само по данни свързани с входния параметър customer, в противен случай обработват всички customers.

6. Да се напише репорт (ПроС програма, която извежда данни от базата по определени критерии и ги извежда във файл), който може да получи първи параметър client или period. Ако първия параметър е client, тогава следващия трябва да е ID на клиента (customer_id). При първи параметър period следват 2 параметъра с две дати date_from и date_to.
   Когато репортът е пуснат с параметър client се извеждат всички фактури, които има клиента с подаденото ИД, заедно с информация дали са платени или не.
   Когато репортът е пуснат с параметър period да се извеждат имената и двете референции на клиентите, които не са направили нито едно плащане в посочения период.

Още изисквания:

Всички ПроС програми – интерфейси, бачове, репорти да имат лог файлове(всяка програма свои), но да използват общи функции за принтиране на съобщение и на грешка log_msg и log_err

log_msg принтира първо датата с минути в секунди в скоби [] след това съобщението

log_err притрира във формат [ERROR][дата с минути и секунти] съобщение

Да се започне с файл “Project Assignments”, който да е в репото на проекта. Пример за такъв файл от друг проект: