Labo 4 BDR

**-- Exercice 01**

drop trigger if exists payment\_date;

DROP TRIGGER IF EXISTS trigEx1;

DELIMITER $$

create trigger trigEx1

before insert on payment

for each row

begin

set

NEW.amount = NEW.amount \* 1.08,

NEW.last\_update = NOW();

end; $$

DELIMITER ;

**-- END Exercice 01**

**-- Exercice 01\_insert**

insert into payment(customer\_id, staff\_id, rental\_id, amount, payment\_date)

values (1, 1, 1, 1, now());

**-- END Exercice 01\_insert**

**-- Exercice 01\_affichage**

select

customer\_id,

staff\_id,

rental\_id,

amount,

payment\_date

from payment

where rental\_id = 1;

**-- END Exercice 01\_affichage**





**-- Exercice 02\_a**

CREATE TABLE staff\_creation\_log (

username varchar(16) NOT NULL,

when\_created TIMESTAMP NOT NULL

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- END Exercice 02\_a**

**-- Exercice 02\_b**

DROP TRIGGER IF EXISTS trigEx2;

DELIMITER $$

create trigger trigEx2

after insert on staff

for each row

begin

insert into staff\_creation\_log (username, when\_created)

values (

NEW.username,

NEW.last\_update

);

end; $$

DELIMITER ;

**-- END Exercice 02\_b**

**-- Exercice 02\_insert**

insert into staff(first\_name, last\_name, address\_id, email, store\_id, active, username, last\_update)

values ("Jean", "Dupond", 5, "jean.dupond@gmail.com", 3, 1, "JDup", NOW());

**-- END Exercice 02\_insert**

**-- Exercice 02\_affichage**

select \*

from staff\_creation\_log

where username = "JDup";

**-- END Exercice 02\_affichage**



**-- Exercice 03**

DROP TRIGGER IF EXISTS trigEx3;

DELIMITER $$

create trigger trigEx3

before insert on staff

for each row

begin

set

NEW.email = concat(NEW.first\_name, '.', New.last\_name, '@sakilastaff.com');

end; $$

DELIMITER ;

**-- END Exercice 03**

**-- Exercice 03\_insert**

insert into staff(first\_name, last\_name, address\_id, email, store\_id, active, username, last\_update)

values ("Jean", "Dupond", 5, "jean.dupond@gmail.com", 3, 1, "JDup", NOW());

**-- END Exercice 03\_insert**

**-- Exercice 03\_affichage**

select

first\_name,

last\_name,

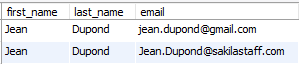
email

from staff

where username = "JDUP";

**-- END Exercice 03\_affichage**





**-- Exercice 04\_a**

DROP TABLE IF EXISTS customer\_store\_log;

create table customer\_store\_log(

customer\_id int,

last\_store\_id int,

register\_date DATE,

unregister\_date DATE

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- END Exercice 04\_a**

**-- Exercice 04\_b**

DROP TRIGGER IF EXISTS trigEx4;

DELIMITER $$

create trigger trigEx4

before update on customer

for each row

begin

if OLD.store\_id <> NEW.store\_id

then

insert into customer\_store\_log(customer\_id, last\_store\_id, register\_date, unregister\_date)

values(OLD.customer\_id, OLD.store\_id, OLD.last\_update, now());

end if;

end;$$

DELIMITER ;

**-- END Exercice 04\_b**

**-- Exercice 04\_update**

update customer set store\_id = 4 where customer\_id = 1;

**-- END Exercice 04\_update**

**-- Exercice 04\_affichage**

select \*

from customer\_store\_log

where customer\_id = 1;

**-- END Exercice 04\_affichage**



**-- Exercice 05**

set global event\_scheduler = ON;

drop event if exists delete\_row;

create event delete\_row on schedule every 1 minute

on completion preserve

do

delete from customer\_store\_log

where customer\_id in (

select

customer\_id

from (

select

customer\_id

from customer\_store\_log

group by customer\_id

having

count(customer\_id) > 1

and

max(unregister\_date) < now() - interval 1 year

) as tab1

);

**-- END Exercice 05**

**-- Exercice 06\_a**

drop view if exists vue\_ex6;

create view vue\_ex6

as

select

phone,

address

from staff

inner join address

on staff.address\_id = address.address\_id;

**-- END Exercice 06\_a**

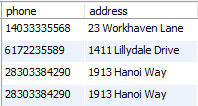
**-- Exercice 06\_a\_affichage**

select \*

from vue\_ex6

limit 20;

**-- END Exercice 06\_a\_affichage**



**-- Exercice 06\_b**

Oui Franklin pourra mettre à jour la base de donnée à travers cette vue car elle ne contient aucune des catégories bloquant les vues updatables :

pas de fonction d'aggregation, ni de distinct, ni de group by, ni de having, union, sous-requête dans un select, sous-requête dans un where, etc..

update vue\_ex6 set phone = '4' where phone = '14033335568';

**-- END Exercice 06\_b**

**-- Exercice 06\_b\_affichage**

select

address,

phone

from address

where address\_id = 3;

**-- END Exercice 06\_b\_affichage**





**-- Exercice 07**

drop view if exists vue\_ex7;

create view vue\_ex7

as

select

email,

film.title as 'titre',

datediff(NOW(), rental.rental\_date + interval film.rental\_duration day) as 'jours\_retard'

from rental

inner join customer

on rental.customer\_id = customer.customer\_id

inner join inventory

on rental.inventory\_id = inventory.inventory\_id

inner join film

on inventory.film\_id = film.film\_id

where rental.return\_date is null

having jours\_retard > 0;

**-- END Exercice 07**

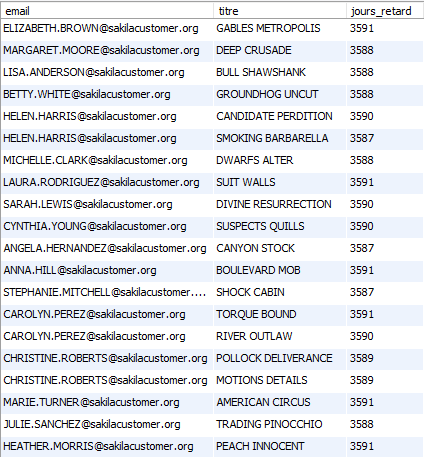
**-- Exercice 07\_affichage**

select \*

from vue\_ex7

limit 20;

**-- END Exercice 07\_affichage**



**-- Exercice 08**

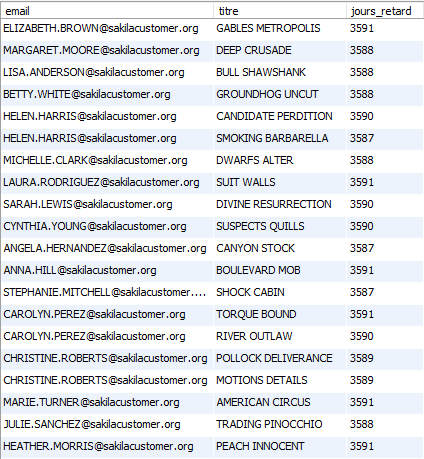
select \*

from vue\_ex7

where jours\_retard > 3

limit 20;

**-- END Exercice 08**



**-- Exercice 09\_a**

drop view if exists vue\_ex9;

create view vue\_ex9

as

select

customer.first\_name as 'prénom',

customer.last\_name as 'nom',

customer.customer\_id as 'id',

count(\*) as 'nombre\_locations'

from rental

inner join customer

on rental.customer\_id = customer.customer\_id

group by customer.customer\_id;

**-- END Exercice 09\_a**

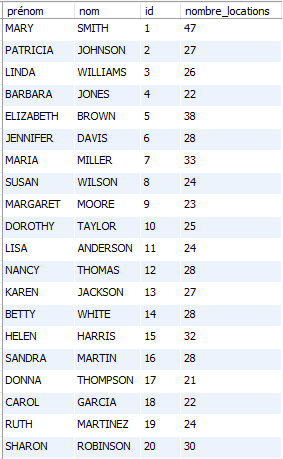
**-- Exercice 09\_a\_affichage**

select \*

from vue\_ex9

limit 20;

**-- END Exercice 09\_a\_affichage**



**-- Exercice 09\_b**

select

prénom,

nom,

id,

max(nombre\_locations) as nombre\_locations

from vue\_ex9

limit 20;

**-- END Exercice 09\_b**



**-- Exercice 10\_a**

drop view if exists vue\_ex10;

create view vue\_ex10

as

select

DATE(rental\_date) as jour\_location,

count(\*) as nombre

from rental

group by jour\_location;

**-- END Exercice 10\_a**

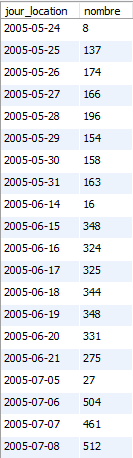
**-- Exercice 10\_a\_affichage**

select \*

from vue\_ex10

limit 20;

**-- END Exercice 10\_a\_affichage**



**-- Exercice 10\_b**

select \*

from vue\_ex10

where jour\_location = "2005-08-01";

**-- END Exercice 10\_b**



**-- Exercice 11\_a**

drop function if exists func\_ex11;

delimiter //

create function func\_ex11 (id\_store int)

returns int

begin

declare nombre int;

select

count(\*)

into nombre

from (

select distinct

film\_id

from

inventory

where

store\_id = id\_store

) as a;

return nombre;

end; //

delimiter ;

**-- END Exercice 11\_a**

**-- Exercice 11\_b**

select

func\_ex11(1) as nombre\_films\_store\_1,

func\_ex11(2) as nombre\_films\_store\_2;

**-- END Exercice 11\_b**



**-- Exercice 12\_a**

drop function if exists func\_ex12;

delimiter //

create function func\_ex12 (id\_store int)

returns int

begin

declare nombre int;

select

count(\*)

into nombre

from

customer

where

store\_id = id\_store;

return nombre;

end; //

delimiter ;

**-- END Exercice 12\_a**

**-- Exercice 12\_b**

select

func\_ex12(1) as nombre\_clients\_store\_1,

func\_ex12(2) as nombre\_clients\_store\_2;

**-- END Exercice 12\_b**



**-- Exercice 13\_a**

drop function if exists func\_ex13;

delimiter //

create function func\_ex13 (id\_store int)

returns double

begin

declare montant double;

select

sum(amount)

into montant

from

payment

inner join

staff

on

payment.staff\_id = staff.staff\_id

where

staff.store\_id = id\_store;

return montant;

end; //

delimiter ;

**-- END Exercice 13\_a**

**-- Exercice 13\_b**

select

func\_ex13(1) as revenu\_total\_store\_1,

func\_ex13(2) as revenu\_total\_store\_2;

**-- END Exercice 13\_b**



**-- Exercice 14\_a**

drop procedure if exists proc\_ex14;

delimiter //

create procedure proc\_ex14 ()

begin

update film

set last\_update = NOW();

end; //

delimiter ;

**-- END Exercice 14\_a**

**-- Exercice 14\_b**

call proc\_ex14();

select

film\_id,

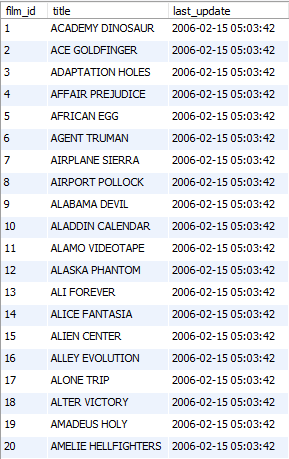
title,

last\_update

from film

limit 20;

**-- END Exercice 14\_b**

****

**-- Exercice 15\_a**

drop procedure if exists proc\_ex15;

delimiter //

create procedure proc\_ex15 (

in id\_store int,

out nombre\_locations int,

out nombre\_exemplaires int)

begin

select

count(rental\_id)

into nombre\_locations

from

rental

inner join

inventory

on

rental.inventory\_id = inventory.inventory\_id

where

store\_id = id\_store;

select

count(inventory\_id)

into nombre\_exemplaires

from

inventory

where

store\_id = id\_store;

end; //

delimiter ;

**-- END Exercice 15\_a**

**-- Exercice 15\_b**

call proc\_ex15(1,@nombre\_locations, @nombre\_exemplaires);

select

@nombre\_locations,

@nombre\_exemplaires;

**-- END Exercice 15\_b**

****