#1

Delimiter //

create procedure usp\_get\_employees\_salary\_above\_35000 ()

begin

select first\_name, last\_name from employees

where salary > 35000

order by first\_name, last\_name, employee\_id;

end //

delimiter ;

#2

Delimiter //

create procedure usp\_get\_employees\_salary\_above (min\_salary decimal (19,4))

begin

select first\_name, last\_name from employees

where salary >= min\_salary

order by first\_name, last\_name, employee\_id;

end //

delimiter ;

#3

Delimiter //

create procedure usp\_get\_towns\_starting\_with (start\_str varchar(20))

begin

select `name` from towns

where `name` like concat(start\_str,'%')

order by `name`;

end //

delimiter ;

#4

Delimiter //

create function ufn\_get\_salary\_level (e\_salary decimal)

returns varchar (10)

deterministic

begin

return (case

when e\_salary < 30000 then 'Low'

when e\_salary between 30000 and 50000 then 'Average'

when e\_salary > 50000 then 'High'

end );

end //

delimiter ;

#5

Delimiter //

create function ufn\_get\_salary\_level (e\_salary decimal)

returns varchar (10)

deterministic

begin

return (case

when e\_salary < 30000 then 'Low'

when e\_salary between 30000 and 50000 then 'Average'

when e\_salary > 50000 then 'High'

end );

end;

create procedure usp\_get\_employees\_by\_salary\_level (s\_level varchar(20))

begin

select first\_name, last\_name from employees

where ufn\_get\_salary\_level(salary) = s\_level

order by first\_name desc, last\_name desc;

end //

delimiter ;

#6

Delimiter //

CREATE FUNCTION ufn\_is\_word\_comprised(set\_of\_letters varchar(50), word varchar(50))

RETURNS int

deterministic

BEGIN

DECLARE set\_index int;

DECLARE word\_indx int;

DECLARE my\_word\_CHR varchar(1);

set word\_indx =1;

WHILE word\_indx<=char\_length(word) DO

set my\_word\_CHR = substring(word,word\_indx,1);

set set\_index = locate (my\_word\_CHR,set\_of\_letters);

if set\_index = 0 then

return 0;

end if;

set word\_indx = word\_indx+1;

end while;

RETURN 1;

END //

Delimter ;

#7

Delimiter //

create procedure usp\_get\_holders\_full\_name ()

begin

select concat(first\_name,' ',last\_name) as full\_name

from account\_holders

order by full\_name, id;

end //

delimiter ;

#8

Delimiter //

create procedure usp\_get\_holders\_full\_name ()

begin

select concat(first\_name,' ',last\_name) as full\_name

from account\_holders

order by full\_name, id;

end //

delimiter ;

#9

Delimiter //

create procedure usp\_get\_holders\_with\_balance\_higher\_than(value int)

begin

select concat(first\_name,' ',last\_name) as full\_name

from account\_holders as a

join accounts as ac

on a.id = ac.account\_holder\_id

group by full\_name

having sum(ac.balance) >value;

end //

delimiter ;

#10

Delimiter //

create function ufn\_calculate\_future\_value (sum decimal(19,4), interest Double, years int)

returns decimal (19,4)

deterministic

begin

return sum \* pow(1+ interest, years);

end ;

delimiter ;

#11

Delimiter //

create function ufn\_calculate\_future\_value (sum decimal(19,4), interest Double, years int)

returns decimal (19,4)

deterministic

begin

return sum \* pow(1+ interest, years);

end;

create procedure usp\_calculate\_future\_value\_for\_account (acc\_id int, interest double)

begin

select a.id,ah.first\_name, ah.last\_name, a.balance,

ufn\_calculate\_future\_value(a.balance,interest,5)

from accounts as a

join account\_holders as ah

on a.account\_holder\_id = ah.id

where a.id = acc\_id;

end //

delimiter ;

#12

Delimiter //

create procedure usp\_deposit\_money(account\_id int, money\_amount decimal(19,4))

begin

start transaction;

if(select count(\*) from accounts where id = account\_id) = 0

or (money\_amount <=0)

then rollback;

else

update accounts

set balance = balance + money\_amount

where id = account\_id;

end if;

end //

delimiter ;

#13

Delimiter //

create procedure usp\_withdraw\_money(account\_id int, money\_amount decimal(19,4))

begin

start transaction;

if(select count(\*) from accounts where id = account\_id) = 0

or (money\_amount <=0)

or (select balance from account where id = account\_id) >= money\_amount

then rollback;

else

update accounts

set balance = balance - money\_amount

where id = account\_id;

end if;

end //

delimiter ;