```
create table CUSTOMER

(

meter_num varchar(4),

meter_type char(1),

pre_reading int(5),

cur_reading int(5),

cust_type char(1),

last_bill_pay char(1),

rate float

);


create table Bill

(

cust char(1),

meter_num varchar(4),

unit_used int,

rate float,

amount float,

surcharge float,

excise float,

net float

);


insert into CUSTOMER values

('A1','T',1000,3000,'A','Y',1),

('B2','T',1800,3400,'I','N',1.25),

('C3','S',1200,2500,'C','N',1.50),

('D4','T',1000,2000,'R','Y',1.30);


SELECT * FROM CUSTOMER;
```

```
delimiter //

create function unit_use(cr int, pr int)

returns int

deterministic

begin

	return (cr-pr);

end; //

delimiter ;


delimiter //

create function amt(r float, u int)

returns float

deterministic

begin

	return (r*u);

end; //

delimiter ;


delimiter //

create function excise(a float, s float)

returns float

deterministic

begin

	return 0.03*(a+s);

end; //

delimiter ;


delimiter //

create function net(am float, sr float, ex float)
```

```
returns float
deterministic
begin
            return am+sr+ex;
end; //
delimiter ;

delimiter //
create procedure cal()
begin
   declare mn varchar(4);
   declare mt char(1);
   declare pr int;
   declare cr int;
   declare cu char(1);
   declare b char(1);
   declare rate float;
   declare unit_use int;
   declare amount float;
   declare sur float;
   declare exc float;
   declare nt float;
   declare y int default 0;
   declare c1 cursor for select * from CUSTOMER;
   declare continue handler for not found set y = 1;
   open c1;
   c1_cursor_loop:loop
                        fetch c1 into mn,mt,pr,cr,cu,b,rate;
      if y=1 then
                                    leave c1_cursor_loop;
```

```
                    end if;
        set unit_use = unit_use(cr,pr);
        set amount = amt(rate,unit_use(cr,pr));
        if mt = 'S' then
                              set sur = 0.05;
                    else
                              set sur = 0.10;
                    end if;
        set exc = excise(amount,sur);
        set nt = net(amount,sur,exc);
        insert into Bill values (cu,mn,unit_use,rate,amount,sur,exc,nt);
            end loop c1_cursor_loop;
      close c1;
end; //
delimiter ;

drop procedure cal;

call cal();
select * from Bill;

create table Total_Bill
(
total_amt int,
total_sur float,
total_exc float,
total_net float
);

delimiter //
```

```
create procedure total()
begin
            declare ta int;
   declare ts float;
   declare te float;
   declare tn float;
   set ta = (select sum(amount) from Bill);
   set ts = (select sum(surcharge) from Bill);
   set te = (select sum(excise) from Bill);
   set tn = (select sum(net) from Bill);
   insert into Total_Bill values (ta,ts,te,tn);
end; //
delimiter ;

call total();
select * from Total_Bill;
```

```
/*Write a stored function to take three parameters, the sides of a triangle. The sides of
the triangle should be accepted from the user. The function should return a Boolean
value:- true if the triangle is valid, false otherwise. A triangle is valid if the length
of each side is less than the sum of the lengths of the other two sides. Check if the
dimensions entered can form a valid triangle.*/

create table tempp
(
Side1 int,
Side2 int,
Side3 int,
Check_triangle varchar(25)
);
```

```
delimiter //
create function tri(a int, b int, c int)
returns boolean
deterministic
begin
	if a<(b+c) and b<(a+c) and c<(a+b) then
		return true;
	else
		return false;
	end if;
end; //
delimiter ;

drop function tri;

delimiter //
create procedure call_tri(a int, b int, c int)
begin
	if tri(a,b,c) then
		insert into tempp values (a,b,c,'Valid Triangle');
	else
		insert into tempp values (a,b,c,'Invalid Triangle');
	end if;
end; //
delimiter ;

call call_tri(30,20,30);
select * from tempp;
```

```
/*Write a function that generates a random number between 1 and 10. Use any logic
```

```
of your choice to achieve this.*/

delimiter //
create function ran()
returns int
deterministic
begin
            declare x int;
   set x = (select rand()*(10-0)+0);
            return x;
end; //
delimiter ;
drop function ran;
select ran() from dual;
```

```
/*Write a stored procedure by the name of Comp_intr to calculate the amount of
interest on a bank account that compounds interest yearly. The formula is:- I
= p (1+ r)^y – p
where:-
 I is the total interest earned.
 p is the principal.
 r is the rate of interest as a decimal less than 1, and
 y is the number of years the money is earning interest.

Your stored procedure should accept the values of p, r and y as parameters and insert
the Interest and Total amount into tempp table. */

create table temp
(
Interest float,
Amount float
);

delimiter //
create procedure ci(p int, r float, y int)
begin
        declare interest float;
   declare amount float;
   set amount = p*pow((1+r),y);
   set interest = amount-p;
   insert into temp values (interest,amount);
end; //
delimiter ;

call ci(1000,0.10,4);
select * from temp;
```

```
/*Create a stored function by the name of Age_calc. Your stored function should
accept the date of birth of a person as a parameter. The stored function should
calculate the age of the person in years. The stored function should return the age
in years. */

delimiter //
create function age_calc(x date)
returns int
deterministic
begin
        declare age int;
   set age = (select year(sysdate())-year(x) from dual);
   return age;
end; //
delimiter ;

select age_calc('1995-10-01') "AGE in yrs" from dual;
```

```
create table Ord_mst
(
ord_no int,
cust_cd varchar(5),
status varchar(5)
);

insert into Ord_mst values (1,'C1','P');

create table Ord_dtl
(
ord_no int,
prod_cd varchar(5),
qty int
);

insert into Ord_dtl values
(1,'P1',100),
(1,'P2',200);

create table Prod_mst
(
prod_cd varchar(5),
prod_name varchar(10),
qty_in_stock int,
booked_qty int
);

insert into Prod_mst values
('P1','Floppies',10000,1000),
('P2','Printers',5000,600),
('P3','Modems',3000,200);

/*1. Write a Before Insert trigger on Ord_dtl. Anytime a row is inserted in Ord_dtl, the
Booked_qty in Prod_mst should be increased accordingly*/

delimiter //
create trigger befins
before insert
on Ord_dtl for each row
begin
		update Prod_mst
   set Booked_qty = Booked_qty+new.qty
   where Prod_mst.Prod_cd = new.Prod_cd;
end; //
delimiter ;
drop trigger befins;
insert into Ord_dtl values (2,'P1',50);
select * from Prod_mst;
select * from Ord_dtl;

/*2.Write a Before Delete trigger on Ord_dtl. Anytime a row is deleted from Ord_dtl, the
Booked_qty in Prod_mst should be decreased accordingly.*/
```

```
delimiter //
create trigger befdel
before delete
on Ord_dtl for each row
begin
		update Prod_mst
   set Booked_qty = Booked_qty-old.qty
   where Prod_mst.Prod_cd = old.Prod_cd;
end; //
delimiter ;

delete from Ord_dtl
where qty = 50;

select * from Prod_mst;
select * from Ord_dtl;

/*3. Write a Before Update of Prod_cd, Qty trigger on Ord_dtl. Anytime the Prod_cd or
Qty is updated, the Booked_qty in Prod_mst should be increased/decreased
accordingly*/

delimiter //
create trigger befup
before update
on Ord_dtl for each row
begin
		update Prod_mst
		set Booked_qty = Booked_qty-old.qty+new.qty
		where new.prod_cd = Prod_mst.Prod_cd;
```

```
end; //
delimiter ;

update Ord_dtl
set qty = 500
where Prod_cd = 'P1';

select * from Prod_mst;
```